Certifikovaný tester

Učebná osnova pre základný stupeň

Verzia 2018 SK

Medzinárodný výbor pre kvalifikácie testovania softvéru

História zmien slovenskej verzie

Verzia	Dátum	Poznámky
ISTQB 2018 SK	28.2.2019	Certifikovaný tester Základný stupeň – slovenský preklad.
ISTQB 2011 SK Beta 2	9.10.2011	Certifikovaný tester Základný stupeň – slovenský preklad - verzia Beta 2.
ISTQB 2011 SK Beta 1	30.9.2011	Certifikovaný tester Základný stupeň – slovenský preklad - verzia Beta 1.
ISTQB 2007 SK Beta 2	10.3.2008	Certifikovaný tester Základný stupeň – slovenský preklad - verzia Beta 2.
ISTQB 2007 SK Beta 1	1.2.2008	Certifikovaný tester Základný stupeň – slovenský preklad - verzia Beta 1.

História zmien anglického originálu

Version	Date	Remarks
ISTQB 2018	27-April-2018	Candidate general release version
ISTQB 2018	12-February-2018	Candidate beta version
ISTQB 2018	19-January-2018	Cross-review internal version 3.0.
ISTQB 2018	15-January-2018	Pre-cross-review internal version 2.9, incorporating Core Team edits.
ISTQB 2018	9-December-2017	Alpha review 2.5 release – Technical edit of 2.0 release, no new content added
ISTQB 2018	22-November-2017	Alpha review 2.0 release – Certified Tester Foundation Level Syllabus Major Update 2018 – see Appendix C – Release Notes for details
ISTQB 2018	12-June-2017	Alpha review release - Certified Tester Foundation Level Syllabus Major Update 2018 – see Appendix C – Release Notes
ISTQB 2011	1-Apr-2011	Certified Tester Foundation Level Syllabus Maintenance Release – see Release Notes
ISTQB 2010	30-Mar-2010	Certified Tester Foundation Level Syllabus Maintenance Release – see Release Notes
ISTQB 2007	01-May-2007	Certified Tester Foundation Level Syllabus Maintenance Release
ISTQB 2005	01-July-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-Feb-1999	ISEB Software Testing Foundation Syllabus V2.0

Obsah

Poďakovanie

Tento dokument bol formálne vydaný inštitúciou General Assembly of the ISTQB (dátum) * *Vyplní sa po schválení* *

Vytvoril ho tím z Medzinárodný výbor pre kvalifikácie testovania softvéru: Klaus Olsen (predseda), Tauhida Parveen (podpredseda), Rex Black (projektový manažér), Debra Friedenberg, Judy McKay, Meile Posthuma, Hans Schaefer, Radoslaw Smilgin, Mike Smith, Steve Toms, Stephanie Ulrich, Marie Walsh a Eshraka Zakaria.

Tím ďakuje Rexovi Blackovi a Dorothy Grahamovej za ich technickú úpravu a revíznemu tímu, tímu krížovej revízie a členským výborom pre ich návrhy a vstupy

Nasledujúce osoby sa zúčastnili na revízii, komentovaní a hlasovaní o tejto osnove: 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 Majerník, 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.

Medzinárodný výbor pre kvalifikácie testovania softvéru (International Software Testing Qualifications Board) - Pracovná skupina ISTQB pre základný stupeň (edícia 2018): Klaus Olsen (predseda), Tauhida Parveen (popdredseda), 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. Hlavný tím ďakuje revíznemu tímu a všetkým členským výborom za ich návrhy.

Pracovná skupina “Základný stupeň” Medzinárodného výboru pre kvalifikácie testovania softvéru (International Software Testing Qualifications Board Working Group Foundation Level), edícia 2011:
Thomas Müller (predseda), Debra Friedenberg. Kľúčový tím ďakuje revíznemu tímu (Dan Almog, Armin Beer, Rex Black, Julie Gardiner, Judy McKay, Tuula Pääkkönen, Eric Riou du Cosquer, Hans Schaefer, Stephanie Ulrich, Erik van Veendendaal) a všetkým lokálnym výborom za návrhy k súčasnej učebnej osnove.

Pracovná skupina “Základný stupeň” Medzinárodného výboru pre kvalifikácie testovania softvéru (International Software Testing Qualifications Board Working Group Foundation Level), edícia 2010:
Thomas Müller (predseda), Rahul Verma, Martin Klonk a Armin Beer. Kľúčový tím ďakuje revíznemu tímu (Rex Black, Mette Bruhn-Pederson, Debra Friedenberg, Klaus Olsen, Judy McKay, Tuula Pääkkönen, Meile Posthuma, Hans Schaefer, Stephanie Ulrich, Pete Williams, Erik van Veendendaal) a všetkým lokálnym výborom za ich návrhy.

Pracovná skupina “Základný stupeň” Medzinárodného výboru pre kvalifikácie testovania softvéru (International Software Testing Qualifications Board Working Group Foundation Level), edícia 2007:
Thomas Müller (predseda), Dorothy Graham, Debra Friedenberg a Erik van Veendendal. Kľúčový tím ďakuje revíznemu tímu (Hans Schaefer, Stephanie Ulrich, Meile Posthuma, Anders Pettersson a Wonil Kwon) a všetkým lokálnym výborom za ich návrhy.

Pracovná skupina “Základný stupeň” Medzinárodného výboru pre kvalifikácie testovania softvéru (International Software Testing Qualifications Board Working Group Foundation Level), edícia 2005: Thomas Müller (predseda), Rex Black, Sigrid Eldh, Dorothy Graham, Klaus Olsen, Maaret Pyhäjärvi, Geoff Thompson a Erik van Veenedendaal. Kľúčový tím ďakuje revíznemu tímu a všetkým lokálnym výborom za ich návrhy.

Preklad do slovenského jazyka - Czech and Slovak Testing Board (CaSTB) - pracovná prekladová skupina do slovenského jazyka:
2018: Marek Majerník (predseda), Karol Frühauf, Roman Jurkech, Róbert Dankanin, Katarína Vavreková, Boris Maťko, Tamara Kadnárová
Revízia prekladu do slovenského jazyka: Michal Fecko, Marko Cagalinec, Vanda Hudáková, Milan Domonji, Ivana Svátková, Patrik Maček, Simona Höffner

Preklad do slovenského jazyka - Czech and Slovak Testing Board (CaSTB):
2011: Róbert Dankanin, Marek Majerník, Ľuboš Práznovský.

Revízia prekladu do slovenského jazyka: Daniela Čuvarská, Marcel Veselka.
2008: Daniela Čuvarská, Róbert Dankanin, Karol Frühauf, Marek Majerník, Ľuboš Práznovský.
Revízia prekladu do slovenského jazyka: Roman Jurkech.

0 Úvod

0.1 Účel tejto učebnej osnovy

Táto osnova tvorí základ pre medzinárodnú kvalifikáciu na testovanie softvéru na základnej úrovni. ISTQB poskytuje túto osnovu:

1. Členským radám na preloženie do miestneho jazyka a akreditovanie poskytovateľov školení. Členské rady môžu prispôsobiť osnovy svojim konkrétnym jazykovým potrebám a pridať odkazy s cieľom prispôsobenia sa miestnym publikáciám.
2. Certifikačným orgánom na vytvorenie skúšobných otázok v ich miestnom jazyku s prispôsobením sa študijným cieľom pre túto osnovu.
3. Poskytovateľom školení na prípravu výučby a určenie vhodných vyučovacích metód.
4. Uchádzačom o certifikáciu pre prípravu na certifikačnú skúšku (buď ako súčasť kurzu alebo samostatne).
5. Pre medzinárodnú komunitu softvérového a systémového inžinierstva, na podporu profesie testovania softvéru a systémov a ako základ pre knihy a články.

ISTQB môže umožniť používať túto osnovu aj iným subjektom pre iné účely za predpokladu, že si vyžiadajú a získajú predchádzajúce písomné povolenie od ISTQB.

0.2 Základná úroveň certifikovaného testera v softvérovom testovaní

Základná úroveň kvalifikácie je určená každému, kto sa zúčastňuje testovania softvéru. Patria sem aj ľudia ako sú testeri, analytici testovania, inžinieri testovania, konzultanti testovania, vedúci testovania, testeri vykonávajúci akceptačné testy a vývojári softvéru. Táto kvalifikácia základnej úrovne je tiež vhodná pre každého, kto potrebuje základné znalosti o testovaní softvéru, ako sú vlastníci produktov, projektoví manažéri, manažéri kvality, manažéri vývojového softvéru, biznis analytici, riaditelia IT a konzultanti pre riadenie. Držitelia základného certifikátu budú môcť pokračovať v kvalifikácii na vyššiu úroveň testovania softvéru.

Prehľad základnej úrovne ISTQB 2018 je samostatným dokumentom, ktorý obsahuje nasledujúce informácie:

- Biznis výsledky pre osnovu
- Matica ukazujúca sledovateľnosť medzi biznis výsledkami a cieľmi učenia
- Zhrnutie tejto osnovy

0.3 Skúmateľné študijné ciele a kognitívne úrovne znalostí

Študijné ciele podporujú biznis výsledky a používajú sa pri vytváraní skúšok pre základnú úroveň certifikovaných testerov.

Vo všeobecnosti je všetok obsah tejto osnovy skúšaný na úrovni K1, s výnimkou úvodu a príloh. Znamená to, že kandidát môže byť vyzvaný, aby rozpoznal, pamätal si alebo si spomenul na kľúčové slovo alebo koncept spomenutý v niektorej z uvádzaných šiestich kapitol. Úrovne znalostí špecifických študijných cieľov sú uvedené na začiatku každej kapitoly a klasifikované takto:

- K1: zapamätať
- K2: pochopiť
- K3: aplikovať

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

Je potrebné si pamätať definície všetkých výrazov uvedených v časti kľúčové slová, ktoré sú tesne pod nadpismi kapitol (K1), aj keď nie sú výslovne uvedené v študijných cieľoch.

0.4 Skúška pre certifikát základnej úrovne

Skúška pre certifikát základnej úrovne bude založená na tejto osnove. Odpovede na otázky pri skúške môžu vyžadovať použitie materiálu založeného na viac ako jednej časti tejto osnovy. Všetky časti osnovy môžu byť súčasťou skúšky s výnimkou úvodu a príloh. Normy, knihy a iné osnovy ISTQB sú zahrnuté ako odkazy, ale ich obsah nebude obsahom skúšky nad rámec toho, čo je zhrnuté v tejto osnove samotnej z takýchto noriem, kníh a iných osnov ISTQB

Formát skúšky je test s viacerými možnosťami, pozostávajúci zo 40 otázok. Pre absolvovanie skúšky je potrebné odpovedať správne na aspoň 65 % otázok (t. j. 26 otázok).

Skúšku je možné absolvovať ako súčasť akreditovaného školenia alebo samostatne (napr. v skúšobnom centre alebo pri verejnej skúške). Dokončenie akreditovaného vzdelávacieho kurzu nie je predpokladom absolvovania skúšky.

0.5 Akreditácia

Členská rada ISTQB môže akreditovať poskytovateľov školení, ktorých študijné materiály sa riadia touto osnovou. Poskytovatelia školení by mali získať akreditačné usmernenia od členskej rady alebo orgánu, ktorý vykonáva akreditáciu. Akreditovaný kurz je uznávaný ako vyhovujúci tejto učebnej osnove a môže vykonávať skúšku ISTQB ako súčasť kurzu.

0.6 Úroveň podrobností

Úroveň podrobností v tejto osnove umožňuje medzinárodne konzistentné kurzy a skúšky. Na dosiahnutie tohto cieľa tvoria osnovu:

- všeobecné vzdelávacie ciele popisujúce zámer základnej úrovne
- zoznam pojmov, ktoré musia študenti pochopiť
- vzdelávacie ciele pre každú oblasť vedomostí, ktoré opisujú výsledok kognitívneho učenia, ktorý sa má dosiahnuť
- opis kľúčových pojmov, vrátane odkazov na zdroje, akými sú napríklad akceptovaná literatúra alebo štandardy

Obsah učebných osnov nie je popisom celej znalostnej oblasti testovania softvéru. Odráža úroveň podrobností, ktoré sa majú pokryť v kurzoch na základnej úrovni. Zameriava sa na testovacie koncepty a techniky, ktoré sa môžu vzťahovať na všetky softvérové projekty, vrátane Agile projektov. Táto osnova neobsahuje žiadne špecifické študijné ciele súvisiace s konkrétnym životným cyklom alebo metódou vývoja softvéru, ale diskutuje o tom, ako sa tieto koncepty uplatňujú v projektoch Agile, iných typoch iteračných a inkrementálnych životných cyklov a v sekvenčných životných cykloch.

0.7 Ako je táto osnova organizovaná

Osnova pozostáva zo šiestich kapitol s preskúmateľným obsahom. Nadpis najvyššej úrovne pre každú kapitolu určuje čas pre danú kapitolu. Časovanie nie je uvedené pre úrovne pod kapitolou. Pre akreditované vzdelávacie kurzy vyžaduje celá osnova minimálne 16,75 hodín výučby s rozdelením do šiestich kapitol takto:

- Kapitola 1: 175 minút Základy testovania
- Kapitola 2: 100 minút Testovanie počas celého životného cyklu vývoja softvéru
- Kapitola 3: 135 minút Statické testovanie
- Kapitola 4: 330 minút Testovacie techniky
- Kapitola 5: 225 minút Manažment testovania
- Kapitola 6: 40 minút Podporné nástroje pre testovanie

1 Základy testovania — 175 minút

Kľúčové slová

pokrytie, ladenie, defekt, omyl, zlyhanie, kvalita, zabezpečenie kvality, prvotná príčina, analýza testovania, základ testovania, testovací prípad, dokončenie testovania, testovacia podmienka, riadenie testovania, testovacie dáta, návrh testu, vykonanie testovania, rozvrh vykonania testovania, implementácia testovania, monitorovanie testovania, testovaný objekt, cieľ testu, testovacia prognóza, plánovanie testovania, testovacia procedúra, testovací súbor, testovanie, testvér, sledovateľnosť, validácia, verifikácia

Študijné ciele pre základy testovania

1.1 Čo je testovanie?

FL-1.1.1 (K1) Identifikovať typické ciele testovania

FL-1.1.2 (K2) Rozlišovať testovanie a ladenie

1.2 Prečo je potrebné testovanie?

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

FL-1.2.2 (K2) Popísať vzťah medzi testovaním a zabezpečením kvality a na príkladoch uviesť, ako testovanie prispieva k vyššej kvalite

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

FL-1.2.4 (K2) Rozlišovať medzi prvotnou 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 vplyv kontextu na proces testovania

FL-1.4.2 (K2) Popísať aktivity testovania a príslušné úlohy v rámci procesu testovania

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

FL-1.4.4 (K2) Vysvetliť hodnotu zachovania sledovateľnosti medzi základom testovania a testovanými pracovnými produktmi

1.5 Psychológia testovania

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

FL-1.5.2 (K2) Vysvetliť rozdiel medzi postojom potrebným pre aktivity testovania a postojom požadovaným pre aktivity v oblasti vývoja

1.1 Čo je to testovanie?

Softvérové systémy sú neoddeliteľnou súčasťou života, od biznis aplikácií (napr. bankovníctvo) až po spotrebiteľské produkty (napr. vozidlá). Väčšina ľudí má skúsenosti so softvérom, ktorý nefungoval podľa očakávaní. Nesprávne fungujúci softvér môže spôsobiť mnohé problémy, vrátane straty peňazí, času, biznis reputácie alebo dokonca poranenie alebo smrť. Testovanie softvéru je spôsob ako zhodnotiť kvalitu softvéru a znížiť riziko zlyhania softvéru počas prevádzky.

Bežným omylom pri testovaní je to, že sa skladá len z bežiacich testov, teda spúšťa softvér a kontroluje výsledky. Ako je to popísané v časti 1.4, testovanie softvéru je proces, ktorý zahŕňa mnohé aktivity; vykonávanie testu (vrátane kontroly výsledkov) je len jednou z týchto aktivít. Proces testovania zahŕňa aj aktivity, ako sú plánovanie testovania, analýza, návrhy a implementácia testov, reportovanie pokroku a výsledkov a hodnotenie kvality testovaného objektu.

Niektoré testovanie zahŕňa vykonávanie testovaného komponentu alebo systému; takéto testovanie sa nazýva dynamické testovanie. Iné testovanie nezahŕňa vykonávanie testovaného komponentu alebo systému; takéto testovanie sa nazýva statické testovanie. Testovanie teda zahŕňa aj kontrolu pracovných produktov, ako sú požiadavky, používateľské príbehy a zdrojový kód.

Omylom v súvislosti s testovaním je to, že sa celkom zameriava na verifikáciu požiadaviek, používateľských príbehov alebo iných špecifikácií. Zatiaľ čo testovanie nezahŕňa iba kontrolu toho, či systém spĺňa uvedené požiadavky, ale zahŕňa aj validáciu, ktorá kontroluje, či systém splní potreby používateľa a kľúčovej osoby v prevádzkových prostrediach.

Aktivity testovania sú organizované a vykonávajú sa odlišne v rôznych životných cykloch (pozrite si časť 2.1).

1.1.1 Typické ciele testovania

Pri každom projekte môžu ciele testovania zahŕňať:

- hodnotenie pracovných produktov, ako sú požiadavky, používateľské príbehy, návrh a kód
- overenie toho, či boli všetky uvedené požiadavky splnené
- potvrdenie toho, či je testovaný objekt kompletný a či funguje tak, ako používatelia a ostatné kľúčové osoby očakávajú
- vybudovanie si dôvery v úroveň kvality testovaného objektu
- predchádzanie defektom
- hľadanie zlyhaní a defektov
- poskytnutie dostatočných informácií pre kľúčové osoby, aby mohli robiť informované (kvalifikované) rozhodnutia, najmä pokiaľ ide o úroveň kvality testovaného objektu
- zníženie úrovne rizika nevhodnej kvality softvéru (napr. predtým nedetekované zlyhania počas prevádzky)
- dodržanie zmluvných, právnych alebo normatívnych požiadaviek alebo noriem a/alebo overenie zhody predmetu testu s takýmito požiadavkami alebo normami

Ciele testovania sa môžu líšiť v závislosti od kontextu komponentu alebo systému, ktorý sa práve testuje, od úrovne testu a modelu životnosti vývoja softvéru. Tieto rozdiely môžu zahŕňať napríklad:

- Počas testovania komponentu sa môže u jedného cieľa vyskytnúť maximálny možný počet zlyhaní, aby sa identifikovali príslušné defekty a aby sa čoskoro opravili. Ďalším cieľom môže byť zvýšenie pokrytia kódu testov komponentov.
- Počas akceptačného testovania môže byť jedným z cieľov potvrdenie toho, že systém funguje podľa očakávaní a spĺňa požiadavky. Ďalším cieľom tohto testovania môže byť poskytnutie informácií kľúčovým osobám o riziku uvoľnenia systému v danom čase.

1.1.2 Testovanie a ladenie

Testovanie a ladenie sú odlišné činnosti. Vykonávanie testov môže ukázať zlyhania spôsobené defektami softvéru. Ladenie je vývojovou aktivitou, ktorá hľadá, analyzuje a opravuje takéto defekty. Ďalšie potvrdzujúce (konfirmačné) testovanie kontroluje, či opravy vyriešili zistené defekty. V niektorých prípadoch testeri zodpovedajú za prvý test a posledný potvrdzujúci test, zatiaľ čo vývojári vykonávajú ladenie a príslušné testovanie komponentov. Pri agilnom vývoji a v niektorých iných životných cykloch môžu byť testeri zapojení do ladenia a testovania komponentov.

Norma ISO (ISO/IEC/IEEE 29119-1) má ďalšie informácie o konceptoch testovania softvéru.

1.2 Prečo je potrebné testovanie?

Prísne testovanie komponentov a systémov a ich príslušná dokumentácia môžu pomôcť znížiť riziko zlyhania počas prevádzky. Keď sa defekty detegujú a potom sa opravia, zvýši sa kvalita komponentov alebo systémov. Okrem toho sa môže testovanie softvéru požadovať pre splnenie zmluvných alebo právnych požiadaviek alebo noriem špecifických pre dané odvetvie.

1.2.1 Ako testovanie prispieva k úspechu

Z minulosti vieme, že sa uvádzajú softvér a systémy do prevádzky napriek tomu, že neodhalené defekty spôsobujú zlyhania alebo iným spôsobom nespĺňajú potreby kľúčových osôb. Technikami uplatnenými s vhodnými znalosťami testovania na jednotlivých úrovniach testov a na vhodných miestach v životnom cykle vývoja sa dá znížiť frekvencia takýchto nežiaducich udalostí.

Medzi príklady patria:

- Testeri zapojení do revízií požiadaviek alebo zdokonaľovania používateľského príbehu by mohli detegovať defekty v týchto pracovných produktoch. Identifikácia a odstraňovanie defektov požiadaviek znižuje riziko vývoja nesprávnej alebo netestovateľnej funkcie.
- Testeri úzko spolupracujú s návrhármi systému, kým sa systém navrhuje, čo môže zvýšiť pochopenie návrhu a spôsobu jeho testovania u jednotlivých strán. Toto rozšírené pochopenie môže znížiť riziko defektov základného návrhu a umožňuje identifikáciu testov v rannom štádiu.
- Úzka spolupráca testerov s vývojármi počas vývoja kódu môže zvýšiť pochopenie kódu a spôsobu jeho testovania u jednotlivých strán. Toto zvýšené pochopenie dokáže znížiť riziko defektov v kóde a testoch.
- Testeri overujú a potvrdzujú softvér pred jeho zverejnením, čím sa môžu detegovať zlyhania, ktoré by sa inak nemuseli zachytiť a podporuje sa tým proces odstraňovania defektov, ktoré spôsobili zlyhania (teda ladenie). Tým sa zvyšuje pravdepodobnosť, že bude softvér plniť potreby kľúčovej osoby a spĺňať požiadavky.

Okrem týchto príkladov dosahovanie zadefinovaných cieľov testu (pozrite si časť 1.1.1) prispieva k celkovému úspechu vývoja a údržby systému.

1.2.2 Zabezpečenie kvality a testovanie

Zatiaľ čo ľudia často používajú výraz *zabezpečenie kvality* (alebo len *QA*), pokiaľ ide o testovanie, zabezpečenie kvality a testovanie nie je to isté, sú však prepojené. Spája ich širší koncept, riadenie kvality. Manažment kvality zahŕňa všetky činnosti, ktoré sa zameriavajú na riadenie organizácie, pokiaľ ide o kvalitu.

Spomedzi iných činností manažment kvality zahŕňa zabezpečenie kvality aj riadenie kvality. Zabezpečenie kvality sa typicky zameriava na dodržiavanie správnych procesov, aby sa zaistila dôvera, že sa dosiahnu vyžadované úrovne kvality. Keď sa budú procesy vykonávať správne, pracovné produkty vytvorené týmito procesmi budú mať vo všeobecnosti vyššiu kvalitu, čo prispieva k prevencii defektov. Okrem toho používanie analýzy prvotnej príčiny na detekciu a odstránenie príčin defektov, spolu so správnym uplatňovaním zistení zo schôdzok za účelom zlepšovania procesov, sú dôležité pre efektívne zabezpečenie kvality.

Riadenie kvality zahŕňa rôzne činnosti, vrátane aktivít testovania, ktoré podporujú dosahovanie vyžadovaných úrovní kvality. Aktivity testovania sú súčasťou celkového vývoja softvéru alebo jeho údržby. Keďže je zabezpečenie kvality spojené so správnym vykonávaním celého procesu, zabezpečenie kvality podporuje správne testovanie. Ako je to popísané v častiach 1.1.1 a 1.2.1, testovanie prispieva k dosahovaniu kvality mnohými spôsobmi.

1.2.3 Chyby, defekty a zlyhania

Človek môže urobiť chybu, ktorá môže viesť k defektu (chybe alebo poruche) v softvérovom kóde alebo v niektorom inom príslušnom pracovnom produkte. Chyba, ktorá vedie k defektu v jednom pracovnom produkte môže spustiť chybu, ktorá povedie k defektu v príslušnom pracovnom produkte. Napríklad chyba v špecifikácii požiadaviek môže viesť k defektu požiadavky, ktorý bude mať za následok chybu programovania, ktorá povedie k defektu v kóde.

Ak sa spustí defekt v kóde, môže to spôsobiť zlyhanie, nie však nevyhnutne za všetkých okolností. Napríklad niektoré defekty si môžu vyžadovať veľmi špecifické vstupy alebo podmienky, aby sa spôsobilo zlyhanie, ktoré sa môže vyskytnúť výnimočne alebo vôbec nie.

Chyby sú vytvorené z mnohých dôvodov, ako napríklad:

- pracovanie pod časovým tlakom
- ľudským zlyhaním
- neskúsenými alebo nedostatočne vyškolenými účastníkmi projektu
- nesprávnou komunikáciou medzi účastníkmi projektu, vrátane komunikácie o požiadavkách a návrhu
- komplexnosťou kódu, návrhom, architektúrou a problémom, ktorý sa má vyriešiť a/alebo použitými technológiami
- nepochopením interného systému alebo rozhraní medzi systémami, najmä vtedy, keď je takýchto interakcií interných systémov a medzi systémami veľké množstvo
- nové, neznáme technológie

Okrem zlyhaní spôsobených defektami kódu môžu byť zlyhania spôsobené aj podmienkami prostredia. Napríklad radiácia, elektromagnetické polia a znečistenie môžu spôsobiť defekty firmvéru alebo ovplyvniť spúšťanie softvéru zmenou hardvérových podmienok.

Nie všetky neočakávané výsledky testov predstavujú zlyhania. Môžu sa vyskytnúť klamné výsledky v dôsledku chýb v spôsobe vykonávania testov alebo v dôsledku defektov v testovacích dátach, prostredí testu alebo inom testvéri alebo z iných dôvodov. Môže vzniknúť aj opačná situácia, kedy podobné chyby alebo defekty vedú ku klamným výsledkom. Klamné výsledky sú testy, ktoré nedetegujú defekty, ktoré by mali detegovať; klamné výsledky sa označujú ako defekty, v skutočnosti to však nie sú defekty.

1.2.4 Defekty, prvotné príčiny a následky

Prvotné príčiny defektov sú prvé činnosti alebo podmienky, ktoré prispeli k vytvoreniu defektov. Defekty sa dajú analyzovať, aby sa identifikovali ich prvotné príčiny, aby sa tým znížil výskyt podobných defektov v budúcnosti. Zameraním sa na najvýznamnejšie prvotné príčiny môže ich analýza viesť k zlepšeniu procesu, čo zabráni výskytu veľkého množstva budúcich defektov.

Napríklad nesprávne platby úrokov, v dôsledku jedného riadku nesprávneho kódu, majú za následok sťažnosti zákazníkov. Chybný kód sa zapísal pri používateľskom príbehu, ktorý bol nejasný v dôsledku toho, že vlastník produktu nepochopil ako rátať úrok. Ak existuje veľké percento defektov vo výpočte úrokov a tieto defekty majú svoju prvotnú príčinu v podobných nepochopeniach, vlastníci produktu by mohli byť vyškolení v oblasti výpočtu úrokov, aby sa v budúcnosti počet takýchto defektov znížil.

V tomto príklade predstavujú sťažnosti zákazníkov javy. Nesprávne platby úrokov predstavujú zlyhania. Nesprávny výpočet v kóde predstavuje defekt a vyplýva z pôvodného defektu, nejasnosti v používateľskom príbehu. Prvotnou príčinou pôvodného defektu bol nedostatok znalostí zo strany vlastníka produktu, čo malo za následok to, že vlastník produktu urobil chybu počas zapisovania používateľského príbehu. Proces analýzy prvotných príčin je popísaný v ISTQB-ETM Expert Level Test Management Syllabus and ISTQB-EITP Expert Level Improving the Test Process Syllabus.

1.3 Sedem princípov testovania

Za posledných 50 rokov bolo navrhnutých niekoľko princípov testovania, ktoré ponúkajú spoločný rámec pre všetky testovania.

1. Testovanie ukazuje na prítomnosť defektov, nie na ich neprítomnosť

Testovanie môže ukázať, že sú defekty prítomné, ale nemôže preukázať, že sa žiadne defekty nevyskytujú. Testovanie znižuje pravdepodobnosť neobjavených defektov v softvéri, ale aj keď sa nenájdu žiadne defekty, testovanie nie je dôkazom jeho správnosti.

2. Vyčerpávajúce testovanie je nemožné

Otestovať všetko (všetky kombinácie vstupov a predbežných podmienok) nie je reálne, okrem triviálnych prípadov. Namiesto vyčerpávajúceho testovania analýzy rizika, testovacích techník a priority by ste sa mali zamerať na prácnosť testovania.

3. Včasné testovanie šetrí čas a peniaze

Ak chcete nájsť defekty včas, mali by ste čo najskôr začať vykonávať statické aj dynamické aktivity v životnom cykle vývoja softvéru. Včasné testovanie sa niekedy nazýva aj *shift left*. Testovanie včas počas cyklu životnosti vývoja softvéru pomáha znížiť počet zmien alebo odstrániť nákladné zmeny (pozrite si časť 3.1).

4. Zhlukovanie defektov

Malý počet modulov obyčajne obsahuje väčšinu defektov objavených počas testovania pred vydaním alebo sú zodpovedné za väčšinu zlyhaní v prevádzke. Predpokladané zhluky defektov a aktuálne pozorované zhluky defektov v teste alebo v prevádzke sú významným vstupom do analýzy rizík, ktorá je zameraná na prácnosť testovania (ako je to uvedené v princípe 2).

5. Dávajte si pozor na pesticídny paradox

Ak sa rovnaké testy opakujú stále dokola, napokon tieto testy už nenájdu žiadny nový defekt. Ak chcete detekovať nové defekty, bude potrebné zmeniť existujúce testy a testovacie dáta a možno bude potrebné navrhnúť nové testy. (Testy už nie sú efektívne pri hľadaní defektov, tak ako pesticídy nie sú po istom čase efektívne pri hubení hmyzu.) V niektorých prípadoch, ako je napríklad automatizované regresné testovanie, má pesticídny paradox pozitívny výsledok, ktorým je relatívne nízky počet defektov regresie.

6. Testovanie je závislé od kontextu

Testovanie sa vykonáva rôznym spôsobom v rôznych kontextoch. Napríklad priemyselný riadiaci softvér kriticky dôležitý pre bezpečnosť sa testuje odlišne ako mobilná aplikácia elektronického obchodu. Ako ďalší príklad môžeme uviesť testovanie v agilnom projekte, ktoré sa vykonáva odlišne od testovania v sekvenčnom životnom cykle projektu (pozrite si časť 2.1).

7. Neprítomnosť chýb je klam

Niektoré organizácie očakávajú, že testeri dokážu spustiť všetky uskutočniteľné testy a nájsť všetky možné defekty, ale princípy 2 a 1 nám hovoria, že to nie je možné. Bolo by klamlivé (teda mylné) očakávať, že nájdením a opravením veľkého počtu defektov zaistíme úspech systému. Napríklad dôkladné testovanie všetkých uvedených požiadaviek a opravenie všetkých nájdených defektov by mohlo vytvoriť systém, ktorý sa ťažko používa, ktorý nespĺňa potreby a očakávania používateľov alebo ktorý je menej výhodný v porovnaní s inými konkurenčnými systémami.

Pozrite si Myers 2011, Kaner 2002 a Weinberg 2008, kde nájdete príklady týchto a iných princípov testovania.

1.4 Proces testovania

Neexistuje univerzálny proces testovania softvéru, existujú však bežné sady aktivít testovania, bez ktorých testovanie pravdepodobne nedosiahne stanovené ciele. Tieto sady aktivít testovania predstavujú proces testovania. Vlastný, špecifický proces testovania softvéru v každej danej situácii závisí od mnohých faktorov. Ktoré aktivity testovania sú zapojené do tohto procesu testovania, ako sa tieto aktivity testovania implementujú a kedy sa tieto aktivity vyskytujú, to môže byť predmetom stratégie testovania.

1.4.1 Proces testovania v kontexte

Faktory kontextu, ktoré ovplyvňujú proces testovania pri organizácii, zahŕňajú, nie však výhradne:

- Model životného cyklu vývoja softvéru a použitú metodológiu projektu
- Úrovne testu a typy testu, ktoré sa berú do úvahy
- Projektové a produktové riziká
- Biznis oblasť
- Prevádzkové obmedzenia, vrátane, nie však výhradne:
  - Rozpočtov a zdrojov
  - Časových plánov
  - Komplexnosti
  - Zmluvných požiadaviek a nariadení
- Organizačných politík a postupov
- Požadovaných interných a externých noriem

Nasledujúce časti popisujú všeobecné aspekty procesov testovania organizácie v zmysle nasledujúcich bodov:

- Aktivity testovania a úlohy
- Testované pracovné produkty
- Sledovateľnosť medzi základom testovania a testovanými pracovnými produktmi

Je veľmi užitočné, ak má základ testovania (pri každej úrovni alebo type testovania, ktoré sa berú do úvahy) zadefinované merateľné kritériá pokrytia. Kritériá pokrytia môžu fungovať efektívne ako kľúčové indikátory výkonu (KPI) na podporu činností, ktoré preukážu dosiahnutie cieľov testovania softvéru (pozrite si časť 1.1.1).

Napríklad pri mobilnej aplikácii môže základ testovania obsahovať zoznam požiadaviek a zoznam podporovaných mobilných zariadení. Každá požiadavka predstavuje prvok základu testovania. Každé podporované zariadenie takisto predstavuje prvok základu testovania. Kritériá pokrytia si môžu vyžadovať minimálne jeden testovací prípad pre každý prvok základu testovania. Po spustení výsledky týchto testov hovoria kľúčovým osobám, či boli uvedené požiadavky splnené a či sa pozorovali zlyhania na podporovaných zariadeniach.

Norma ISO (ISO/IEC/IEEE 29119-2) obsahuje ďalšie informácie o procesoch testovania.

1.4.2 Aktivity testovania a úlohy

Proces testovania pozostáva z nasledujúcich hlavných skupín činností:

- Plánovanie testovania
- Monitorovanie testovania a riadenie
- Analýza testovania
- Návrhy testov
- Implementácia testovania
- Vykonávanie testovania
- Dokončenie testovania

Každá skupina činností sa skladá z jednotlivých aktivít, ktoré budú popísané v podčastiach nižšie. Každá aktivita v rámci každej skupiny aktivít môže pozostávať z viacerých individuálnych úloh, ktoré by sa pri jednotlivých projektoch alebo release menili.

Hoci mnohé z týchto skupín činností sa môžu zdať ako logicky nasledujúce po sebe, často sa implementujú reťazovo. Napríklad agilný vývoj zahŕňa malé reťazenia návrhu softvéru, buildy a testy, ktoré sa vykonávajú priebežne s podporou neustáleho plánovania. Aktivity testovania sa teda vykonávajú reťazovo, neustále v rámci prístupu vývoja. Aj pri sekvenčnom vývoji bude postupná logická sekvencia činností zahŕňať prekrývanie, kombináciu, súbežnosť alebo vynechanie, takže sa obyčajne požaduje nastavenie týchto hlavných činností na mieru v rámci kontextu systému a projektu.

Plánovanie testovania

Plánovanie testovania zahŕňa činnosti, ktoré definujú ciele testovania a prístup pre splnenie týchto cieľov testu v rámci obmedzení daných kontextom (napr. špecifikácia vhodných testovacích techník a úloh a formulácia rozvrhu testovania pre splnenie termínu). Plány testov sa môžu prepracovať na základe spätnej väzby z činností monitorovania a riadenia. Plánovanie testovania je ďalej vysvetlené v kapitole 5.2.

Monitorovanie a riadenie testovania

Monitorovanie testovania zahŕňa neustále porovnávanie pokroku s testovacím plánom pomocou každej metriky monitorovania testov zadefinovanej v testovacom pláne. Riadenie testu zahŕňa vykonávanie činností potrebných na splnenie cieľov testovacieho plánu (ktorý sa môže časom aktualizovať). Monitorovanie testovania a riadenie podporuje hodnotenie výstupných kritérií, ktoré sa označujú ako definícia v niektorých životných cykloch (pozrite si ISTQB-AT Foundation Level Agile Tester Extension Syllabus). Napríklad hodnotenie výstupných kritérií pre vykonanie testovania v rámci danej úrovne testovania môže zahŕňať:

- Kontrolu výsledkov testov a protokolov na základe stanovených kritérií pokrytia
- Hodnotenie úrovne kvality komponentu alebo kvality systému na základe výsledkov testovania a protokolov
- Stanovenie toho, či je potrebných viac testov (napr. či testy pôvodne určené na dosiahnutie istej úrovne pokrytia rizika produktu zlyhali, čo by si vyžadovalo napísanie a spustenie dodatočných testov)

Pokrok testovania v porovnaní s plánom sa oznamuje kľúčovým osobám formou testovacích reportov o pokroku, vrátane odchýlok od plánu a informácií na podporu rozhodnutia o zastavení testovania.

Monitorovanie a riadenie testovania sú ďalej popísané v časti 5.3.

Analýza testovania

Počas analýzy testovania sa základ testovania analyzuje, aby sa identifikovali testovateľné funkcie a aby sa zadefinovali príslušné testovacie podmienky. Inými slovami, analýza testovania určuje, „čo sa bude testovať“ v zmysle merateľných kritérií pokrytia.

Analýza testovania zahŕňa nasledujúce hlavné činnosti:

- Analýza základu testovania vhodného pre uvažovanú úroveň testu, napríklad:
  - Špecifikácie požiadaviek, ako sú biznis požiadavky, funkcionálne požiadavky, systémové požiadavky, používateľské príbehy, popisy, prípady použitia alebo podobné pracovné produkty, ktoré špecifikujú požadované funkcionálne alebo nefunkcionálne správanie komponentu alebo systému
  - Informácie o návrhu a implementácii, ako sú diagramy architektúry systému alebo softvéru alebo dokumenty, špecifikácie návrhu, toky volaní, diagramy modelovania (napr. UML alebo diagramy vzťahov entít), špecifikácie rozhraní alebo podobné pracovné produkty, ktoré špecifikujú štruktúru komponentu alebo systému
  - Implementácia samotného komponentu alebo systému, vrátane kódu, metadát databázy, dotazov a rozhraní
  - Správy o analýze rizík, ktoré môžu brať do úvahy funkcionálne, nefunkcionálne a štrukturálne aspekty komponentu alebo systému
- Hodnotenie základu testovania a položiek testu pre identifikáciu defektov rôznych typov, ako sú:
  - Nejasnosti
  - Vynechania
  - Nesúlad
  - Nepresnosti
  - Protiklady
  - Nadbytočné prehlásenia

- Identifikácia funkcií a sád funkcií, ktoré sa otestujú
- Zadefinovanie a uprednostňovanie testovacích podmienok pri každej funkcii na základe analýzy základu testovania a zváženia funkcionálnych, nefunkcionálnych a štrukturálnych vlastností, iných biznis a technických faktorov a úrovní rizík
- Zachytenie obojsmernej sledovateľnosti medzi jednotlivými prvkami základu testovania a príslušných testovacích podmienok (pozrite si časti 1.4.3 a 1.4.4)

Použitie techník testovania čiernej skrinky, bielej skrinky a techník testovania založených na skúsenostiach môže byť užitočné pri procese analýzy testovania (pozrite si kapitolu 4) pre zníženie pravdepodobnosti vynechania významných testovacích podmienok a pre zadefinovanie presnejších testovacích podmienok.

Analýza testovania vytvára testovacie podmienky, ktoré sa použijú ako ciele testovania v testovacích chartách. Testovacie charty sú typické pracovné produkty v niektorých typoch testovania založeného na skúsenostiach (pozrite si časť 4.4.2). Keď sú tieto ciele testovania sledovateľné až po základ testovania, dá sa zmerať pokrytie dosiahnuté počas takéhoto testovania založeného na skúsenostiach.

Identifikácia defektov počas analýzy testovania je významnou potenciálnou výhodou, najmä keď sa nepoužíva žiadny ďalší proces revízie a/alebo proces testovania je úzko spojený s procesom revízie. Takéto činnosti analýzy testovania nielenže overujú, či sú požiadavky konzistentné, správne vyjadrené a kompletné, ale zároveň potvrdzujú, či sa požiadavky správne zameriavajú na potreby zákazníka, používateľa a inej kľúčovej osoby. Napríklad techniky ako je vývoj riadený správaním (BDD) a vývoj riadený akceptačným testom (ATDD), ktoré zahŕňajú generovanie testovacích podmienok a testovacích prípadov z používateľských príbehov a akceptačných kritérií pred kódovaním, takisto overujú, potvrdzujú a detekujú defekty v používateľských príbehoch a akceptačných kritériách (pozrite si ISTQB Foundation Level Agile Tester Extension syllabus).

Návrhy testov

Počas návrhu testov sa testovacie podmienky vypracujú pre všeobecné testovacie prípady, sady všeobecných testovacích prípadov a iný testvér. Analýza testovania teda odpovedá na otázku „čo testovať?“, zatiaľ čo návrh testov odpovedá na otázku „ako testovať?“

Návrh testov zahŕňa nasledujúce hlavné činnosti:

- Navrhovanie a uprednostňovanie testovacích prípadov a sád testovacích prípadov
- Identifikáciu potrebných testovacích dát na podporu testovacích podmienok a testovacích prípadov
- Návrh testovacieho prostredia a identifikáciu každej požadovanej časti infraštruktúry a nástrojov
- Zachytenie obojsmernej sledovateľnosti medzi základom testovania, testovacími podmienkami, testovacími prípadmi a testovacími procedúrami (pozrite si časť 1.4.4)

Vypracovanie testovacích podmienok v zmysle testovacích prípadov a sád testovacích prípadov počas návrhu testov často zahŕňa používanie testovacích techník (pozrite si kapitolu 4).

Tak ako pri analýze testovania, aj návrh testov môže mať za následok identifikáciu podobných typov defektov v základe testovania. Tak ako pri analýze testovania, identifikácia defektov počas návrhu testov je významnou potenciálnou výhodou.

Implementácia testovania

Počas implementácie testovania sa vytvorí a/alebo dokončí testvér potrebný pre vykonávanie testov, vrátane sekvencovania testovacích prípadov do testovacích procedúr. Návrh testov teda odpovedá na otázku „ako testovať?“, zatiaľ čo implementácia testov odpovedá na otázku „máme všetko potrebné na to, aby sme mohli spustiť testovanie?“

Implementácia testov zahŕňa nasledujúce hlavné činnosti:

- Vývoj a uprednostňovanie testovacích procedúr a potenciálne vytváranie automatizovaných testovacích skriptov
- Vytváranie testovacích sád z testovacích procedúr a (ak existujú) automatických testovacích skriptov
- Usporiadanie testovacích sád v rozvrhu vykonania testov tak, aby vykonanie testov bolo efektívne (pozrite si časť 5.2.4)
- Vytváranie prostredia testu (potenciálne vrátane testovacích postrojov, virtualizácie služby, simulátorov a iných položiek infraštruktúry) a overenie toho, či sa správne nastavilo všetko potrebné
- Príprava testovacích dát a zaistenie toho, že sú správne vložené v testovacom prostredí
- Overenie a aktualizácia obojsmernej sledovateľnosti medzi základom testovania, testovacími podmienkami, testovacími prípadmi, testovacími procedúrami a testovacími sadami (pozrite si časť 1.4.4)

Návrh testov a úlohy implementácie testovania sa často kombinujú.

Pri prieskumnom testovaní a iných typoch testovania založeného na skúsenostiach sa môže vyskytnúť návrh testov a implementácia a ich dokumentácia v rámci vykonávania testovania. Prieskumné testovanie môže byť založené na testovacích chartách (vytvárajú sa v rámci analýzy testovania) a prieskumné testy sa vykonávajú ihneď po navrhnutí a implementácii (pozrite si časť 4.4.2).

Vykonávanie testovania

Počas vykonávania testovania sa spúšťajú testovacie sady v súlade s rozvrhom vykonania

testov. Vykonávanie testovania zahŕňa nasledujúce hlavné činnosti:

- Zaznamenávanie ID a verzií testovaných položiek alebo testovaného objektu, testovacieho nástroja a testvéru
- Vykonávanie testov buď manuálne alebo pomocou nástrojov na vykonávanie testovania
- Porovnanie aktuálnych výsledkov s očakávanými výsledkami
- Analýza anomálií pre stanovenie ich pravdepodobných príčin (napr. môžu sa vyskytnúť zlyhania v dôsledku defektov v kóde, ale môžu sa vyskytnúť aj klamné výsledky [pozrite si časť 1.2.3])
- Oznamovanie defektov na základe zistených zlyhaní (pozrite si časť 5.6)
- Zapísanie výsledku vykonaného testu (napr. úspešný, neúspešný, zablokovaný)
- Opakovanie testovacích aktivít buď ako výsledku činnosti vykonávanej pri anomálii alebo ako časť plánovaného testovania (napr. vykonanie opraveného testu, potvrdzujúce testovanie a/alebo regresné testovanie)
- Overenie a aktualizácia obojsmernej sledovateľnosti medzi základom testovania, testovacími podmienkami, testovacími prípadmi, testovacími procedúrami a výsledkami testu.

Dokončenie testovania

Aktivity v rámci dokončenia testovania zbierajú dáta z dokončených testovacích aktivít pre konsolidáciu skúseností, testvéru a všetkých ostatných potrebných informácií. Aktivity ukončenia testovania sa vyskytujú v míľnikoch projektu ako napríklad vtedy, keď sa vydá softvérový systém, dokončí sa (alebo zruší) projekt testovania, dokončí sa iterácia agilného projektu (napr. ako časť retrospektívneho stretnutia), dokončí sa úroveň testovania alebo release údržby je ukončený.

Dokončenie testovania zahŕňa nasledujúce hlavné činnosti:

- Kontrola toho, či sú uzavreté všetky reporty o defektoch, zadanie žiadostí o zmenu alebo nedokončených položiek produktov pri všetkých defektoch, ktoré zostanú nevyriešené po skončení testovania
- Vytvorenie testovacieho sumárneho reportu, ktorý sa oznámi kľúčovým osobám
- Dokončenie a archivácia testovacieho prostredia, testovacích dát, testovacej infraštruktúry a iného testvéru pre neskoršie opätovné použitie
- Presunutie testvéru na tímy údržby, iné projektové tímy a/alebo iné kľúčové osoby, ktoré by ho mohli využiť
- Analýza ponaučení z ukončených testovacích aktivít pre stanovenie zmien potrebných pre budúce plány, release a projekty
- Používanie informácií získaných pre zlepšenie zrelosti procesu testovania

1.4.3 Pracovné produkty testovania

Pracovné produkty testovania sa vytvárajú v rámci procesu testovania. Keďže sa výrazným spôsobom mení implementácia procesu testovania u organizácií, rovnako výrazne sa menia aj typy pracovných produktov vytvorených počas tohto procesu, pokiaľ ide o spôsoby, akými sú tieto pracovné produkty organizované a riadené a názvy používané pre tieto pracovné produkty. Táto učebná osnova dodržuje proces testovania uvedený vyššie a pracovné postupy popísané v tejto učebnej osnove a v glosári ISTQB. Norma ISO (ISO/IEC/IEEE 29119-3) môže slúžiť aj ako smernica pre pracovné produkty testovania.

Mnohé testovacie pracovné produkty popísané v tejto časti sa dajú zachytiť a riadiť pomocou nástroja na riadenie testovania a nástrojov na riadenie defektov (pozrite si kapitolu 6).

Pracovné produkty plánovania testovania

Pracovné produkty plánovania testovania obyčajne zahŕňajú jeden alebo viacero testovacích plánov. Testovací plán zahŕňa informácie o základoch testovania, s ktorými budú spojené ostatné testovacie pracovné produkty prostredníctvom informácií o sledovateľnosti (pozrite dole a časť 1.4.4), ako aj výstupné kritériá (alebo definíciu vykonaného), ktoré sa použijú počas monitorovania a riadenia testovania. Testovacie plány sú popísané v časti 5.2.

Pracovné produkty monitorovania a riadenia testovania

Pracovné produkty monitorovania a riadenia testovania obyčajne zahŕňa rôzne typy reportov o testovaní, vrátane reportu o pokroku testovania (ktoré sa vystavujú neustále a/alebo pravidelne) a sumárnych testovacích reportov (ktoré sa vystavujú pri rôznych míľnikoch). Všetky testovacie reporty by mali poskytovať dáta relevantné pre príjemcov vzťahujúce sa na pokrok testovania k dátumu vystavenia reportu, vrátane sumarizácie výsledkov vykonania testovania, keď budú dostupné.

Pracovné produkty monitorovania a riadenia testovania by sa mali zameriavať aj na úlohy riadenia projektu, ako je vykonávanie úloh, prideľovanie a používanie zdrojov a prácnosť.

Pracovné produkty monitorovania a riadenia testovania vytvorené počas týchto aktivít, sú ďalej popísané

v časti 5.3 tejto učebnej osnovy.

Pracovné produkty analýzy testovania

Pracovné produkty analýzy testovania zahŕňajú zadefinované a uprednostňované testovacie podmienky, pričom každá z nich je ideálne obojsmerne sledovateľná podľa špecifických prvkov základu testovania, ktoré pokrýva. Pri prieskumnom testovaní môže analýza testovania zahŕňať vytvorenie potvrdení o teste. Analýza testovania môže mať za následok aj objavovanie a oznamovanie defektov v základe testovania.

Pracovné produkty návrhu testov

Návrh testov má za následok testovacie prípady a sady testovacích prípadov pre dosiahnutie testovacích podmienok zadefinovaných v analýze testovania. Vždy je vhodné navrhnúť všeobecné testovacie prípady, bez konkrétnych hodnôt vstupných dát a očakávaných výsledkov. Takéto všeobecné testovacie prípady sa dajú opätovne použiť vo viacerých cykloch testu s rôznymi konkrétnymi dátami, pričom stále vhodne dokumentujú rozsah testovacieho prípadu. Ideálne je každý testovací prípad obojsmerne sledovateľný podľa testovacích podmienok, ktoré pokrýva.

Návrh testov má za následok aj návrh a/alebo identifikáciu potrebných testovacích dát, návrh testovacieho prostredia a identifikáciu infraštruktúry a nástrojov, hoci rozsah, v ktorom sa tieto výsledky dokumentujú, sa výrazne mení.

Testovacie podmienky zadefinované v analýze testovania sa môžu ďalej zdokonaliť v návrhu testov.

Pracovné produkty implementácie testovania

Pracovné produkty implementácie testovania:

- Testovacie procedúry a sekvenovanie týchto testovacích procedúr
- Testovacie sady
- Rozvrh vykonania testov

Ideálne, keď sa dokončí implementácia testov, dosiahnutie kritérií pokrytia stanovených v testovacom pláne sa dá preukázať obojsmernou sledovateľnosťou medzi testovacími procedúrami a špecifickými prvkami základu testovania, prostredníctvom testovacích prípadov a testovacích podmienok.

V niektorých prípadoch implementácia testovania zahŕňa vytváranie pracovných produktov, ktoré používajú alebo sú používané nástrojmi, ako je virtualizácia služby a automatizované testovacie skripty.

Implementácia testovania môže mať za následok aj vytvorenie a verifikáciu testovacích dát a prostredia testu. Kompletnosť dokumentácie dát a/alebo výsledky verifikácie prostredia sa môžu výrazne líšiť.

Testovacie dáta slúžia na pridelenie konkrétnych hodnôt vstupom a očakávaným výsledkom testovacích prípadov. Takéto konkrétne hodnoty, spolu s explicitnými pokynmi o používaní konkrétnych hodnôt, menia všeobecné testovacie prípady na konkrétne testovacie prípady. Rovnaký všeobecný testovací prípad môže používať odlišné testovacie dáta, keď sa vykonáva na rôznych releasoch testovaného objektu. Konkrétne očakávané výsledky, ktoré sú spojené s konkrétnymi testovacími dátami sa identifikujú pomocou testovacej prognózy.

Pri prieskumnom testovaní niektoré pracovné produkty návrhu testov a implementácie sa môžu vytvoriť počas vykonávania testovania, hoci rozsah, v ktorom sa prieskumné testy (a ich sledovateľnosť podľa špecifických prvkov základu testovania) dokumentujú, sa môže výrazne líšiť.

Testovacie podmienky zadefinované pri analýze testovania sa môžu ďalej zdokonaliť pri implementácii testovania.

Pracovné produkty vykonávania testovania

Pracovné produkty vykonávania testovania zahŕňajú:

- Dokumentáciu stavu jednotlivých testovacích prípadov alebo testovacích procedúr (napr. pripravený na spustenie, úspešný, neúspešný, zablokovaný, zámerne preskočený, atď.)
- Reporty o defektoch (pozrite si časť 5.6)
- Dokumentácia k testovanej(ým) položke(ám), testovaným objektom, testovacím nástrojom a testvéru zapojených do testovania

Ideálne, keď sa dokončí vykonanie testovania, stav každého prvku základu testovania sa dá stanoviť a oznámiť prostredníctvom obojsmernej sledovateľnosti podľa príslušných testovacích procedúr. Napríklad môžeme povedať, ktoré požiadavky boli úspešné pri všetkých naplánovaných testoch, ktoré požiadavky boli neúspešné pri testoch a/alebo ktoré obsahujú defekty a pri ktorých požiadavkách sa na vykonanie plánovaných testov stále ešte len čaká. To umožňuje verifikovať, či sa splnili kritériá pokrytia a umožňuje to reportovanie výsledkov testovania v tom zmysle, že sú pre kľúčové osoby zrozumiteľné.

Pracovné produkty dokončenia testovania

Pracovné produkty dokončenia testovania zahŕňajú sumárne testovacie reporty, akčné prvky zlepšovania následných projektov alebo iterácií (napr. podľa retrospektívy agilného projektu), žiadostí o zmenu alebo nedokončených položiek produktov a dokončeného testvéru.

1.4.4 Sledovateľnosť medzi základom testovania a pracovnými produktmi testovania

Ako je to uvedené v časti 1.4.3, pracovné produkty testovania a názvy týchto pracovných produktov sa výrazne líšia. Bez ohľadu na tieto odchýlky, je dôležité vytvoriť za účelom implementácie efektívneho monitorovania testovania a riadenia a udržiavať sledovateľnosť počas celého procesu testovania medzi jednotlivými prvkami základu testovania a rôznymi pracovnými produktmi testovania spojenými s týmto prvkom, ako je to popísané vyššie. Okrem hodnotenia pokrytia testov dobrá sledovateľnosť podporuje:

- Analýzu vplyvu zmien
- Zabezpečenie auditovateľnosti testovania
- Splnenie kritérií riadenia IT
- Zlepšenie zrozumiteľnosti testovacieho reportu o pokroku a sumárneho testovacieho reportu, ktoré budú zahŕňať stav prvkov základu testovania (napr. požiadavky, ktoré prešli testami, požiadavky, ktoré neprešli testami a požiadavky, ktoré na testy čakajú)
- Upravovanie technických aspektov testov pre kľúčové osoby tak, aby im porozumeli
- Poskytnutie informácií pre hodnotenie kvality produktu, schopnosti procesu a pokroku projektu na základe biznis cieľov

Niektoré nástroje na riadenie testovania poskytujú modely pracovných produktov testovania, ktoré zodpovedajú časti alebo všetkým pracovným produktom uvedených v tejto časti. Niektoré organizácie si vytvárajú svoje vlastné systémy riadenia na organizáciu pracovných produktov a poskytnutie informácií o sledovateľnosti, ktoré požadujú.

1.5 Psychológia testovania

Vývoj softvéru, vrátane testovania softvéru, zahŕňa ľudí. Preto má ľudská psychológia významný vplyv na testovanie softvéru.

1.5.1 Ľudská psychológia a testovanie

Identifikácia defektov počas statického testu, ako sú preskúmanie požiadaviek, zdokonaľovanie používateľského príbehu alebo identifikácia zlyhaní počas vykonávania dynamického testovania, sa môžu vnímať ako kritika produktu, ale autor ju môže vnímať ako jeho kritiku. Prvok ľudskej psychológie nazývaný potvrdzujúci základ môže sťažiť prijímanie informácií, ktoré nie sú v súlade s aktuálnym stavom znalostí. Napríklad, keďže vývojári očakávajú, že je ich kód správny, majú takzvaný potvrdzujúci základ, ktorý im sťažuje prijať to, že je ich kód nesprávny. Okrem tohto potvrdzujúceho základu môžu ostatné kognitívne základy sťažovať ľuďom pochopiť alebo prijať informácie zistené pri testovaní. A ďalej, bežnou ľudskou vlastnosťou je viniť nositeľa zlých správ a informácie zistené pri testovaní často obsahujú zlé správy.

V dôsledku týchto psychologických faktorov môžu niektorí ľudia vnímať testovanie ako deštruktívnu aktivitu, hoci v značnej miere prispieva k pokroku projektu a ku kvalite výsledného produktu (pozrite si časti 1.1 a 1.2). Ak sa chcete pokúsiť obmedziť tieto vnemy, informácie o defektoch a zlyhaniach by ste mali oznámiť vecným spôsobom. Týmto spôsobom sa dá znížiť napätie medzi testermi a analytikmi, vlastníkmi produktu, návrhármi a vývojármi. To sa uplatňuje počas statického aj dynamického testovania.

Testeri a vedúci testovania musia mať dobré vzájomné znalosti, aby dokázali efektívne komunikovať o defektoch, zlyhaniach, výsledkoch testov, pokroku testovania a rizikách a budovať pozitívne vzťahy s kolegami. Spôsoby dobrej komunikácie zahŕňajú nasledujúce príklady:

- Nebojujte, spolupracujte. Pripomeňte každému, že máte spoločný cieľ, ktorým je vyššia kvalita systémov
- Zdôraznite výhody testovania. Napríklad autorom môžu informácie o defektoch pomôcť zlepšiť ich pracovné produkty a zručnosti. Pokiaľ ide o organizáciu, defekty zistené a opravené počas testovania ušetria čas a peniaze a znížia celkové riziko pre kvalitu produktov
- Výsledky testov a iné zistenia komunikujte neutrálnym spôsobom, zameraným na fakty, bez kritizovania osoby, ktorá defektnú položku vytvorila. Zapíšte si cieľ a faktické správy o defektoch a overte zistenia
- Pokúste sa pochopiť ako sa cítia ostatní a pochopiť dôvody, prečo by mohli na informácie reagovať negatívne
- Potvrďte si, že druhá osoba pochopila, čo ste povedali, a naopak

Typické ciele testov sa spomenuli vyššie (pozrite si časť 1.1). Jasné zadefinovanie správneho súboru cieľov testu má dôležité psychologické súvislosti. Väčšina ľudí má tendenciu spájať svoje plány a správanie s cieľmi stanovenými tímom, manažmentom a inými kľúčovými osobami. Takisto je dôležité, aby títo testeri dodržiavali tieto ciele s minimálnou osobnou zaujatosťou.

1.5.2 Postoje testera a vývojára

Vývojári a testeri obyčajne premýšľajú iným spôsobom. Primárnym cieľom vývoja je navrhnúť a vytvoriť produkt. Ako sme uviedli vyššie, medzi ciele testovania patrí overenie a potvrdenie produktu, hľadanie defektov pred zverejnením a podobne. Toto sú rôzne ciele, ktoré si vyžadujú rôzne postoje. Ak tieto postoje spojíme, pomôže nám to dosiahnuť vyššiu úroveň kvality produktu.

Postoj odráža predpoklady jednotlivca a preferované metódy rozhodovania a riešenia problémov. Postoj testera by mal zahŕňať zvedavosť, profesionálny pesimizmus, kritické oko, pozornosť k detailom a motiváciu k dobrej a pozitívnej komunikácii i vzťahom. Postoj testera má tendenciu rásť a dozrievať, keď tester získava nové a nové skúsenosti.

Postoj vývojára môže zahŕňať niektoré prvky postoja testera, ale úspešní vývojári sú často viac zainteresovaní do návrhu a vytvárania riešení ako do zamýšľania sa nad tým, v čom by mohli byť tieto riešenia zlé. Okrem toho, vďaka potvrdzujúcemu základu sa ťažko hľadajú chyby v ich vlastnej práci.

So správnym postojom dokážu vývojári otestovať svoj vlastný kód. Rôzne modely životného cyklu vývoja softvéru majú často rôzne spôsoby organizácie testerov a aktivít testovania. Keďže niektoré aktivity testovania vykonávajú nezávislí testeri, zvyšuje sa tým efektívnosť detekcie defektov, čo je obzvlášť dôležité pre veľké, komplexné alebo bezpečnostné systémy. Nezávislí testeri prinášajú perspektívu, ktorá je odlišná od perspektívy autorov pracovného produktu (napr. biznis analytici, vlastníci produktu, návrhári a programátori), keďže majú odlišné kognitívne základy ako autori.

2 Testovanie počas celého životného cyklu vývoja softvéru — 100 minút

Kľúčové slová

akceptačné testovanie, alfa testovanie, beta testovanie, komerčný krabicový softvér (COTS), integračné testovanie komponentov, testovanie komponentov, potvrdzovacie (konfirmačné) testovanie, zmluvné akceptačné testovanie, funkcionálne testovanie, analýza dopadu, integračné testovanie, testovanie počas údržby, nefunkcionálne testovanie, prevádzkové akceptačné testovanie, regresné testovanie, regulačné akceptačné testovanie, sekvenčný model vývoja, systémové testovanie integrácie, systémové testovanie, základ testovania, testovací prípad, testovacie prostredie, úroveň testovania, testovaný objekt, účel testovania, typ testovania, používateľské akceptačné testovanie, testovanie bielej skrinky

Študijné ciele pre testovanie počas celého životného cyklu vývoja softvéru

2.1 Modely životného cyklu vývoja softvéru

FL-2.1.1 (K2) Vysvetliť vzťahy medzi aktivitami vývoja softvéru a aktivitami testovania v rámci životného cyklu vývoja softvéru

FL-2.1.2 (K1) Identifikovať dôvody, pre ktoré musia byť modely životného cyklu vývoja softvéru prispôsobené kontextu charakteristík projektu a produktu

2.2 Úrovne testovania

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

2.3 Typy testovania

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

FL-2.3.2 (K1) Rozpoznať, že funkcionálne testovanie, nefunkcionálne testovanie a testovanie bielej skrinky sú prítomné na všetkých úrovniach testovania

FL-2.3.3 (K2) Porovnať účely potvrdzovacieho (konfirmačného) testovania a regresného testovania

2.4 Testovanie počas údržby

FL-2.4.1 (K2) Zhrnúť faktory spustenia testovania počas údržby

FL-2.4.2 (K2) Popísať úlohy analýzy dopadov pri testovaní počas údržby

2.1 Modely životného cyklu vývoja softvéru

Model životného cyklu vývoja softvéru opisuje typy aktivít vykonávaných v každej etape v projekte vývoja softvéru a ako sa aktivity navzájom líšia logicky a chronologicky. Existuje celý rad rôznych modelov životného cyklu vývoja softvéru, z ktorých každý vyžaduje rôzne prístupy k testovaniu.

2.1.1 Vývoj softvéru a testovanie softvéru

Dôležitou súčasťou úlohy testera je oboznámenie sa s bežnými modelmi životného cyklu vývoja softvéru, aby mohol uskutočniť príslušné aktivity testovania.

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

- Pre každú vývojovú aktivitu existuje zodpovedajúca aktivita testovania.
- Každá úroveň testovania má účely testovania, ktoré sú špecifické pre danú úroveň.
- Analýza a návrh testovania pre danú úroveň testovania začínajú počas príslušnej vývojovej aktivity.
- Testeri sa zúčastňujú diskusií s cieľom stanoviť a spresniť požiadavky a návrhy a zúčastňujú sa na hodnotení pracovných produktov (napr. požiadaviek, návrhov, používateľských príbehov atď.) hneď, ako budú k dispozícii návrhy.

Bez ohľadu na to, ktorý model životného cyklu vývoja softvéru je vybraný, aktivity testovania by mali začať v počiatočných fázach životného cyklu a mali by dodržiavať testovací princíp včasného testovania.

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

- Modely sekvenčného vývoja
- Modely iteračného a inkrementálneho vývoja

Model sekvenčného vývoja opisuje proces vývoja softvéru ako lineárny postupný tok aktivít. To znamená, že akákoľvek fáza vývojového procesu by mala začať, keď je predchádzajúca fáza ukončená. Teoreticky neexistuje prekrývanie fáz, ale v praxi je prospešné získať včasnú spätnú väzbu z nasledujúcej fázy.

Vo vodopádovom modeli sú vývojové aktivity (napr. analýza požiadaviek, návrh, kódovanie, testovanie) ukončené postupne jedna po druhej. V tomto modeli sa aktivity testovania vyskytujú až po dokončení všetkých ostatných vývojových aktivít.

Na rozdiel od vodopádového modelu, V-model integruje proces testovania v celom vývojovom procese a implementuje princíp včasného testovania. Okrem toho model V obsahuje úrovne testovania súvisiace s každou zodpovedajúcou vývojovou fázou, čo ešte viac podporuje včasné testovanie (pozri časť 2.2 s diskusiou o úrovniach testovania). V tomto modeli prebieha vykonávanie testovania spojeného s každou úrovňou testovania postupne, ale v niektorých prípadoch dochádza k prekrývaniu.

Modely sekvenčného vývoja prinášajú softvér, ktorý obsahuje kompletný súbor vlastností, ale zvyčajne si vyžadujú mesiace alebo až roky na dodanie kľúčovým osobám a používateľom.

Inkrementálny vývoj zahŕňa stanovenie požiadaviek, navrhovanie, budovanie a testovanie systému v častiach, čo znamená, že vlastnosti softvéru rastú postupne. Veľkosť týchto zmien vlastností sa líši, pričom niektoré metódy majú väčšie časti a niektoré menšie časti. Zmeny vlastností môžu byť malé ako jediná zmena na obrazovke používateľského rozhrania alebo napríklad nová možnosť otázky.

Iteračný vývoj nastáva, keď sú skupiny vlastností špecifikované, navrhnuté, postavené a testované spoločne v sérii cyklov, často s pevnou dobou trvania. Iterácie môžu zahŕňať zmeny vlastností vytvorených v predchádzajúcich iteráciách spolu so zmenami v rozsahu projektu. Každá iteračná verzia prináša pracovný softvér, ktorý je rastúcou podmnožinou celkovej množiny vlastností, až kým sa nedosiahne konečný softvér alebo sa vývoj nezastaví.

Medzi príklady patria:

- Rational Unified Process: Každá iterácia má tendenciu byť relatívne dlhá (napríklad dva až tri mesiace) a zmeny vlastností sú zodpovedajúco veľké, napríklad dve alebo tri skupiny príbuzných vlastností
- Scrum: Každá iterácia má tendenciu byť relatívne krátka (napr. hodiny, dni alebo niekoľko týždňov) a zmeny vlastností sú zodpovedajúco malé, napríklad niekoľko vylepšení a/alebo dve alebo tri nové funkcie
- Kanban: Implementované s iteráciami s pevnou dĺžkou alebo bez, ktoré môžu po dokončení priniesť buď jediné vylepšenie alebo vlastnosť, alebo môžu spoločne zoskupovať vlastností na uvoľnenie naraz
- Špirála (alebo prototypovanie): Zahŕňa vytváranie experimentálnych zmien, z ktorých niektoré môžu byť výrazne prepracované alebo dokonca vynechané v rámci ďalších vývojových prác

Komponenty alebo systémy vyvinuté pomocou týchto metód často zahŕňajú prekrývanie a iteráciu úrovní testovania počas celého vývoja. V ideálnom prípade sa každá vlastnosť testuje na viacerých úrovniach testovania, keď sa posúva smerom k dodaniu. V niektorých prípadoch tímy využívajú nepretržité dodávanie alebo nepretržité nasadenie, pričom obe zahŕňajú významnú automatizáciu viacerých úrovní testovania ako súčasť dodávateľských reťazcov. Mnohé rozvojové snahy využívajúce tieto metódy zahŕňajú aj koncepciu samoorganizujúcich tímov, ktoré môžu zmeniť spôsob organizácie testovania ako aj vzťah medzi testerom a vývojárom.

Tieto metódy tvoria rastúci systém, ktorý môže byť sprístupnený pre koncových používateľov na báze jednotlivých funkcií, na báze jednotlivých iterácií alebo tradičnejším spôsobom celkového sprístupnenia. Bez ohľadu na to, či sú zmeny softvéru sprístupnené koncovým používateľom, regresné testovanie je čoraz dôležitejšie s tým, ako systém narastá.

Na rozdiel od sekvenciálnych modelov môžu iteračné a inkrementálne modely dodať do užívania použiteľný softvér v priebehu týždňov alebo dokonca niekoľkých dní, ale môžu dodávať kompletnú sadu požiadaviek na produkt len počas niekoľkých mesiacov alebo rokov.

Ďalšie informácie o testovaní softvéru v kontexte vývoja Agile môžete nájsť v ISTQB-AT Foundation Agile Tester Extension Syllabus, Black 2017, Crispin 2008 a Gregory 2015.

2.1.2 Modely životného cyklu vývoja softvéru v kontexte

Modely životného cyklu vývoja softvéru musia byť vybrané a prispôsobené kontextu charakteristík projektu a produktu. Vhodný model životného cyklu vývoja softvéru by mal byť vybraný a prispôsobený na základe cieľa projektu, druhu produktu, ktorý sa vyvíja, biznis priorít (napr. čas uvedenia na trh – Time to Market) a identifikovaných rizík produktov a projektov. Napríklad vývoj a testovanie menšieho vnútorného administratívneho systému by sa malo odlišovať od vývoja a testovania systému kritického z hľadiska bezpečnosti, akým je napríklad systém riadenia bŕzd automobilov. Ďalším príkladom môžu byť niektoré prípady organizačných a kultúrnych problémov, ktoré brzdia komunikáciu medzi členmi tímu, čo môže brániť iteratívnemu vývoju.

V závislosti od kontextu projektu môže byť potrebné kombinovať alebo reorganizovať úrovne testovania a/alebo aktivity testovania. Napríklad pre integráciu komerčného krabicovového softvéru (COTS) do väčšieho systému môže kupujúci vykonať testovanie spolupôsobiteľnosti a úrovni systémovo integračného testovania (napr. integrácia do infraštruktúry a iných systémov) a na úrovni akceptačného

testovania (funkcionálna a nefunkcionálna, spolu s akceptačným testovaním používateľa a prevádzkovým akceptačným testovaním). Pozri časť 2.2 pre diskusiu o úrovniach testovania a časť 2.3 pre diskusiu o typoch testovania.

Okrem toho je možné kombinovať samotné modely životného cyklu vývoja softvéru. Napríklad model V môže byť použitý na vývoj a testovanie backendových systémov a ich integrácií, zatiaľ čo vývojový model Agile môže byť použitý na vývoj a testovanie používateľského rozhrania (front-end user interface) a funkcionality. Prototypovanie sa môže použiť v skorej fáze projektu, s inkrementálnym vývojovým modelom prijatým po dokončení experimentálnej fázy.

Systémy Internetu vecí (IoT), ktoré pozostávajú z mnohých rôznych objektov, ako sú zariadenia, produkty a služby, zvyčajne používajú samostatné modely životného cyklu vývoja softvéru pre každý objekt. To predstavuje mimoriadnu výzvu pre vývoj systémových verzií Internetu vecí. Okrem toho životný cyklus vývoja softvéru takýchto objektov kladie väčší dôraz na neskoršie fázy životného cyklu vývoja softvéru po ich zavedení do prevádzky (napr. prevádzka, aktualizácia a vyradenie z prevádzky).

2.2 Úrovne testovania

Úrovne testovania sú skupiny aktivít testovania, ktoré sú spoločne organizované a riadené. Každá úroveň testovania je príkladom procesu testovania pozostávajúceho z aktivít opísaných v časti 1.4, ktoré sa vykonávajú vo vzťahu k softvéru na danej úrovni vývoja, od jednotlivých jednotiek alebo komponentov až po kompletné systémy alebo prípadne systémy systémov. Úrovne testovania súvisia s inými aktivitami v rámci životného cyklu vývoja softvéru. Úrovne testovania v tejto učebne osnove sú:

- Testovanie komponentov
- Integračné testovanie
- Systémové testovanie
- Akceptačné testovanie

Úrovne testovania sú charakterizované týmito atribútmi:

- Špecifické ciele
- Základ testovania, uvádzaný na odvodenie prípadov testovania
- Testovaný objekt (t. j., čo sa testuje)
- Typické defekty a zlyhania
- Špecifické prístupy a zodpovednosti

Pre každú úroveň testovania je potrebné vhodné testovacie prostredie. Pri akceptačnom testovaní je napríklad ideálne testovacie prostredie, ktoré je podobné produkčnému prostrediu, zatiaľ čo pri testovaní komponentov vývojári zvyčajne používajú vlastné vývojové prostredie.

2.2.1 Testovanie komponentov

Ciele testovania komponentov

Testovanie komponentov (tiež známe ako jednotkové testovanie alebo testovanie modulov) sa zameriava na komponenty, ktoré sú samostatne testovateľné. Ciele testovania komponentov zahŕňajú:

- Zníženie úrovne rizika
- Overenie, či funkcionálne a nefunkcionálne správanie komponentu je podľa návrhu a špecifikácie
- Budovanie dôvery v kvalitu komponentov
- Nájdenie chýb v komponente
- Zabránenie úniku defektov na vyššiu úroveň testovania

V niektorých prípadoch, hlavne v inkrementálnych a iteračných vývojových modeloch (napr. Agile), kde prebiehajú zmeny kódu, automatizované regresné testy komponentov zohrávajú kľúčovú úlohu pri budovaní dôvery, že zmeny nepoškodili existujúce komponenty.

Testovanie komponentov sa často robí izolovane od zvyšku systému v závislosti od modelu životného cyklu vývoja softvéru a samotného systému, ktorý môže vyžadovať falošné objekty, virtualizáciu služieb, postroje, testovacie kódy a ovládače. Testovanie komponentov môže zahŕňať funkcionalitu (napr. správnosť výpočtov), nefunkcionálne charakteristiky (napr. hľadanie úniku pamäte) a štrukturálne vlastnosti (napr. testovanie rozhodovania).

Základ testovania

Príklady pracovných produktov, ktoré možno použiť ako základ testovania pre testovanie komponentov, zahŕňajú:

- Podrobný návrh
- Kód
- Dátový model
- Špecifikácia komponentov

Testované objekty

Typické testované objekty pri testovaní komponentov zahŕňajú:

- Komponenty, jednotky a moduly
- Štruktúry kódu a dát
- Triedy
- Databázové moduly

Typické defekty a zlyhania

Príklady typických chýb a zlyhania pri testovaní komponentov zahŕňajú:

- Nesprávnu funkcionalitu (napr. nie je opísaná v špecifikáciách návrhu)
- Problémy s tokom dát
- Nesprávny kód a logika

Chyby sa zvyčajne odstránia hneď, ako sa nájdu, často bez formálneho manažmentu defektov. Avšak, keď vývojári nahlásia chyby, poskytujú tým dôležité informácie pre analýzu prvotných príčin a zlepšenie procesu.

Špecifické prístupy a zodpovednosti

Testovanie komponentov zvyčajne vykonáva vývojár, ktorý napísal kód, ale vyžaduje to aspoň prístup k testovanému kódu. Vývojári môžu alternovať medzi testovaním komponentov a nachádzaním a opravou defektov. Vývojári budú často písať a vykonávať testovanie po napísaní kódu pre komponent. Avšak, najmä pri vývoji Agile, písanie automatizovaných testovacích prípadov komponentov môže predchádzať písaniu aplikačného kódu.

Napríklad, uvažujte o vývoji riadenom testami (TDD). Vývoj riadený testami je vysoko iteračný a je založený na cykloch vývoja automatizovaných testovacích prípadov, následnom vytváraní a integrácii malých častí kódu, vykonávaní testovania komponentov, opravách akýchkoľvek problémov a opätovnom faktorovaní kódu. Tento proces pokračuje až do úplného vybudovania komponentu a absolvovania všetkých testov komponentu. Vývoj riadený testami je príkladom prístupu „test-first“ (najprv test). Zatiaľ čo vývoj riadený testami pochádza z eXtreme Programming (XP), rozšíril sa na iné formy Agile a tiež na sekvenčné životné cykly (pozri ISTQB-AT Foundation Level Agile Tester Extension Extension).

2.2.2 Integračné testovanie

Ciele integračného testovania

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

- Zníženie úrovne rizika
- Overenie, či funkcionálne a nefunkcionálne správanie rozhraní je podľa návrhu a špecifikácií
- Budovanie dôvery v kvalitu rozhraní
- Hľadanie defektov (ktoré môžu byť v samotných rozhraniach alebo v rámci komponentov alebo systémov)
- Zabránenie úniku defektov na vyššiu úroveň testovania

Tak ako pri testovaní komponentov, v niektorých prípadoch automatizované integračné regresné testy poskytujú dôveru, že zmeny nepoškodili existujúce rozhrania, komponenty alebo systémy.

V tejto učebnej osnove sú opísané dve rôzne úrovne integračného testovania, ktoré sa môžu vykonávať na testovaných objektoch rôznej veľkosti nasledovne:

- Testovanie integrácie komponentov sa zameriava na interakcie a rozhrania medzi integrovanými komponentmi. Testovanie integrácie komponentov sa vykonáva po testovaní komponentov a je všeobecne automatizované. V iteračnom a inkrementálnom vývoji sú integračné testy komponentov zvyčajne súčasťou procesu kontinuálnej integrácie.
- Testovanie systémovej integrácie sa zameriava na interakcie a rozhrania medzi systémami, súbormi programov a mikroslužbami. Testovanie systémovej integrácie môže zahŕňať aj interakcie s externými organizáciami a rozhrania poskytované externými organizáciami (napr. webové služby). V tomto prípade vývojárska organizácia nekontroluje externé rozhrania, čo môže vytvoriť rôzne výzvy pre testovanie (napr. zabezpečiť, že defekty blokujúce testy v kóde externých organizácii, sú vyriešené, usporiadanie testovacích prostredí atď.). Testovanie systémovej integrácie sa môže vykonať po systémovom testovaní alebo súbežne s prebiehajúcimi systémovými testovacími aktivitami (v sekvenčnom vývoji aj v iteračnom a inkrementálnom vývoji).

Základ testovania

Príklady pracovných produktov, ktoré môžu byť použité ako základ testovania pre integračné testovanie, zahŕňajú:

- Návrh softvéru a systému
- Sekvenčné diagramy
- Špecifikácie rozhraní a komunikačných protokolov
- Prípady použitia
- Architektúra na úrovni komponentov alebo systému
- Pracovný tok (Workflow)
- Definície externých rozhraní

Testované objekty

Typické testované objekty pre testovanie integrácie zahŕňajú:

- Subsystémy
- Databázy
- Infraštruktúru
- Rozhrania
- API (rozhrania pre programovanie aplikácií)
- Mikroslužby

Typické defekty a zlyhania

Príklady typických chýb a zlyhaní pri testovaní integrácie komponentov zahŕňajú:

- Nesprávne dáta, chýbajúce dáta alebo nesprávne kódovanie dát
- Nesprávne sekvencovanie alebo načasovanie volaní rozhrania
- Nesúlad rozhraní
- Zlyhania v komunikácii medzi komponentami
- Neriešené alebo nesprávne riešené zlyhania komunikácie medzi komponentmi
- Nesprávne predpoklady o význame, jednotkách alebo hraniciach dát prenášaných medzi komponentmi

Príklady typických defektov a porúch pri testovaní systémovej integrácie zahŕňajú:

- Nekonzistentné štruktúry správ medzi systémami
- Nesprávne údaje, chýbajúce údaje alebo nesprávne kódovanie dát
- Nesúlad rozhraní
- Zlyhania v komunikácii medzi systémami
- Neriešené alebo nesprávne riešené zlyhania komunikácie medzi systémami

- Nesprávne predpoklady o význame, jednotkách alebo hraniciach dát prenášaných medzi systémami
- Nesplnenie povinných bezpečnostných predpisov

Špecifické prístupy a zodpovednosti

Testovanie integrácie komponentov a systémové testovanie by sa mali sústrediť na samotnú integráciu. Napríklad, ak sa integruje modul A s modulom B, testovanie by sa malo zamerať na komunikáciu medzi modulmi a nie na funkcionalitu jednotlivých modulov, ktoré by malo byť úlohou v rámci testovania komponentov. Ak integrujeme systém X so systémom Y, testovanie by sa malo zamerať na komunikáciu medzi systémami, nie na funkcionalitu jednotlivých systémov, keďže tá by mala byť úlohou v rámci systémového testovania. Používajú sa funkcionálne, nefunkcionálne a štrukturálne typy testovania.

Testovanie integrácie komponentov je často zodpovednosťou vývojárov. Testovanie systémovej integrácie je vo všeobecnosti zodpovednosťou testerov. V ideálnom prípade by testeri, ktorí vykonávajú testovanie systémovej integrácie, mali pochopiť architektúru systému a mali by ovplyvniť plánovanie integrácie.

Ak sú integračné testovanie a integračná stratégia naplánované pred vytvorením komponentov alebo systémov, môžu byť tieto komponenty alebo systémy vytvorené v poradí, ktoré je potrebné pre čo najúčinnejšie testovanie. Systémové stratégie integrácie môžu byť založené na architektúre systému (napríklad postup zhora nadol a zdola nahor), funkcionálne úlohy, sekvencie spracovania transakcií alebo niektoré ďalšie aspekty systému alebo komponentov. S cieľom zjednodušiť izoláciu chýb a skoré zistenie chýb by integrácia mala byť normálne inkrementálna (t. j. malý počet ďalších komponentov alebo systémov naraz) a nie vo forme „veľký tresk“ (t. j. integrácia všetkých komponentov alebo systémov v jednom kroku). Analýza rizík najkomplexnejších rozhraní môže pomôcť so zameraním integračného testovania.

Čím väčší je rozsah integrácie, tým ťažšie sa izolujú chyby na určitý komponent alebo systém, čo môže viesť k zvýšenému riziku a dodatočnému času na riešenie problémov. To je jeden z dôvodov, prečo sa kontinuálna integrácia, v ktorej je softvér integrovaný na báze jednotlivých zložiek (t. j. funkcionálna integrácia), stáva bežnou praxou. Takáto kontinuálna integrácia často zahŕňa automatické regresné testovanie, ideálne na viacerých úrovniach testovania.

2.2.3 Systémové testovanie

Ciele systémového testovania

Testovanie systému sa zameriava na správanie a schopnosti celého systému alebo produktu, pričom často berie do úvahy end-to-end úlohy, ktoré systém môže vykonávať a nefunkcionálne správanie, ktoré prejavuje pri vykonávaní týchto úloh. Ciele systémového testovania zahŕňajú:

- Zníženie úrovne rizika
- Overenie, či funkcionálne a nefunkcionálne správanie systému je podľa návrhu a špecifikácie
- Overenie, že systém je kompletný a bude pracovať podľa očakávania
- Budovanie dôvery v kvalitu systému ako celku
- Vyhľadávanie chýb
- Zabránenie úniku defektov na vyššiu úroveň testovania alebo do produkcie
- Pre určité systémy môže byť cieľom overovanie kvality údajov.

Rovnako ako v prípade testovania komponentov a integračného testovania v niektorých prípadoch automatizované systémové regresné testy poskytujú istotu, že zmeny nepoškodili existujúce funkcie alebo end-to-end schopnosti. Systémové testovanie často poskytuje informácie, ktoré používajú kľúčové osoby na rozhodnutia o vydaní. Systémové testovanie môže tiež spĺňať právne alebo regulačné požiadavky alebo

štandardy.

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

Základ testovania

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

- Špecifikácie systémových a softvérových požiadaviek (funkcionálne a nefunkcionálne)
- Správy o analýze rizík
- Prípady použitia
- Popisy a používateľské príbehy
- Modely správania systému
- Stavové diagramy
- Systémové a používateľské príručky

Testované objekty

Typické testované objekty na systémové testovanie zahŕňajú:

- Aplikácie
- Hardvérové/softvérové systémy
- Operačné systémy
- Testovaný systém (SUT)
- Konfigurácia systému a konfiguračné dáta

Typické chyby a zlyhania

Príklady typických chýb a zlyhaní pri systémovom testovaní zahŕňajú:

- Nesprávne kalkulácie
- Nesprávne alebo neočakávané funkcionálne alebo nefunkcionálne správanie systému
- Nesprávne riadenie a/alebo toky dát v rámci systému
- Neschopnosť riadne a úplne vykonať funkcionálne úlohy typu end-to-end
- Neschopnosť systému správne fungovať v produkčnom prostredí(-iach)
- Neschopnosť systému fungovať podľa popisu v systémových a používateľských príručkách

Špecifické prístupy a zodpovednosti

Systémové testovanie by sa malo sústrediť na celkové end-to-end správanie systému ako celku, a to funkcionálne aj nefunkcionálne. Systémové testovanie by malo používať najvhodnejšie techniky (pozri kapitolu 4) pre aspekty testovaného systému. Môže sa napríklad vytvoriť rozhodovacia tabuľka, aby sa overilo funkcionálne správanie ako je popísané v biznis pravidlách.

Systémové testovanie bežne vykonávajú nezávislí testeri. Defekty v špecifikáciách (napr. chýbajúce používateľské príbehy, nesprávne stanovené biznis požiadavky atď.) môžu viesť k nedostatočnému pochopeniu alebo nezhodám o očakávanom správaní systému. Takéto situácie môžu spôsobiť nežiadúce pozitíva a negatíva, ktoré spôsobujú stratu času a znižujú účinnosť detekcie defektov. Včasné zapojenie testerov do zdokonaľovania používateľských príbehov alebo aktivít statického testovania, ako sú revízie, 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:

- Vytváranie dôvery v kvalitu systému ako celku
- Overenie, že systém je kompletný a bude pracovať podľa očakávania
- Overenie, či funkcionálne a nefunkcionálne správanie systému je podľa špecifikácie

Akceptačné testovanie môže poskytnúť informácie na posúdenie pripravenosti systému na nasadenie a používanie zákazníkom (koncový používateľ). V priebehu akceptačného testovania sa môžu vyskytnúť defekty, ale zistenie defektov často nie je cieľom a nájdenie významného počtu defektov počas akceptačného testovania môže byť v niektorých prípadoch považované za veľké riziko projektu. Akceptačné testovanie môže tiež spĺňať právne alebo regulačné požiadavky alebo štandardy.

Bežné formy akceptačného testovania zahŕňajú:

- Používateľské akceptačné testovanie
- Prevádzkové akceptačné testovanie
- Zmluvné a regulačné akceptačné testovanie
- Alfa a beta testovanie

Každý typ testovania je opísaný v nasledujúcich štyroch pododdieloch.

Požívateľské akceptačné testovanie (UAT)

Akceptačné testovanie systému používateľmi sa zvyčajne zameriava na overenie vhodnosti používania systému určenými používateľmi v reálnom alebo simulovanom prevádzkovom prostredí. Hlavným cieľom je vybudovať dôveru, že používatelia môžu používať systém na uspokojenie svojich potrieb, splnenie požiadaviek a vykonávanie biznis procesov s minimálnymi ťažkosťami, nákladmi a rizikami.

Prevádzkové akceptačné testovanie (OAT)

Akceptačné testovanie systému prevádzkovým personálom alebo personálom pre správu systémov sa zvyčajne vykonáva v (simulovanom) produkčnom prostredí. Testy sa zameriavajú na prevádzkové aspekty a môže zahŕňať:

- Testovanie zálohovania a obnovy
- Inštalácia, odinštalácia a upgrade
- Obnova po nehode
- Správa používateľov
- Úlohy údržby

- Úlohy plnenia dát a migrácie
- Kontrola zraniteľnosti zabezpečenia
- Testovanie výkonnosti

Hlavným cieľom prevádzkového akceptačného testovania je budovanie dôvery, že prevádzkovatelia alebo správcovia systému vedia udržiavať systém pri jeho správnom fungovaní pre používateľov v prevádzkovom prostredí, dokonca aj za mimoriadnych alebo ťažkých podmienok.

Zmluvné a regulačné akceptačné testovanie

Zmluvné akceptačné testovanie sa vykonáva voči zmluvným akceptačným kritériám pre softvér vyvinutý na mieru. Akceptačné kritériá by sa mali definovať zároveň so zmluvnými dohodami. Zmluvné akceptačné testovanie sa často vykonáva používateľmi alebo nezávislými testermi.

Regulačné akceptačné testovanie sa vykonáva voči všetkým predpisom, ktoré sa musia dodržiavať, ako sú napríklad vládne, zákonné alebo bezpečnostné predpisy. Regulačné akceptačné testovanie často vykonávajú používatelia alebo nezávislí testeri, niekedy s výsledkami, ktoré sú osvedčené alebo auditované regulačnými agentúrami.

Hlavným cieľom zmluvného a regulačného akceptačného testovania je budovanie dôvery, že sa dosiahol zmluvný alebo regulačný súlad.

Alfa a beta testovanie

Alfa a beta testovanie zvyčajne používajú vývojári softvéru COTS, ktorí chcú získať spätnú väzbu od potenciálnych alebo existujúcich používateľov, zákazníkov a/alebo prevádzkovateľov pred uvedením softvérového produktu na trh. Alfa testovanie sa vykonáva na mieste vývojárskej organizácie, nie vývojovým tímom, ale potenciálnymi alebo existujúcimi zákazníkmi a/alebo prevádzkovateľmi alebo nezávislým testovacím tímom. Beta testovanie vykonávajú potenciálni alebo existujúci zákazníci a/alebo prevádzkovatelia v ich vlastných lokalitách. Beta testovanie môže prísť po alfa testovaní alebo sa môže vykonať bez akéhokoľvek predchádzajúceho alfa testovania.

Jedným z cieľov alfa testovania a beta testovania je budovanie dôvery medzi potenciálnymi alebo existujúcimi zákazníkmi a/alebo prevádzkovateľmi vzhľadom na to, že môžu používať systém za bežných, každodenných podmienok a v prevádzkovom prostredí(-iach) na dosiahnutie svojich cieľov s minimálnymi ťažkosťami, nákladmi a rizikom. Ďalším cieľom môže byť zistenie defektov súvisiacich s podmienkami a prostredím (prostrediami), v ktorých sa systém bude používať, najmä ak vývojové tímy ťažko replikujú tieto podmienky a prostredie (prostredia).

Základ testovania

Príklady pracovných produktov, ktoré možno použiť ako skúšobný základ pre akúkoľvek formu akceptačného testovania, zahŕňajú:

- Biznis procesy
- Požiadavky používateľa alebo biznis požiadavky
- Predpisy, právne zmluvy a normy
- Prípady použitia
- Systémové požiadavky
- Systémovú dokumentáciu alebo dokumentáciu používateľa
- Postupy inštalácie
- Report o analýze rizík

Okrem toho ako základ testovania pre odvodenie testovacích prípadov pre prevádzkové akceptačné testovanie možno použiť jeden alebo viac z týchto pracovných produktov:

- Procedúry zálohovania a obnovy
- Procedúry obnovy po nehode
- Nefunkcionálne požiadavky
- Prevádzková dokumentácia
- Pokyny pre nasadenie a inštaláciu
- Výkonnostné ciele
- Databázové balíky
- Bezpečnostné štandardy alebo regulácie

Typické testované objekty

Typické testované objekty pre akúkoľvek formu akceptačného testovania zahŕňajú:

- Testovaný systém
- Konfigurácie systému a konfiguračné dáta
- Biznis procesy pre plne integrovaný systém
- Systémy obnovy a záložné pracoviská (hot sites) (pre kontinuitu prevádzky a testovanie obnovy po havárii)
- Prevádzkové a údržbové procesy
- Formy
- Správy
- Existujúce a konvertované výrobné dáta

Typické chyby a zlyhania

Príklady typických defektov pre akúkoľvek formu akceptačného testovania zahŕňajú:

- Systémové pracovné toky (workflow) nespĺňajú biznis alebo používateľské požiadavky
- Biznis pravidlá sa nerealizujú správne
- Systém nespĺňa zmluvné alebo regulačné požiadavky
- Nefunkcionálne zlyhania, ako napríklad zraniteľnosti zabezpečenia, nedostatočná výkonnosť pri vysokej záťaži alebo nesprávne fungovanie na podporovanej platforme

Špecifické prístupy a zodpovednosti

Akceptačné testovanie je často zodpovednosťou zákazníkov, biznis používateľov, vlastníkov produktov alebo prevádzkovateľov systému a iné kľúčové osoby môžu byť taktiež zapojené.

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

- Akceptačné testovanie COTS softvérového produktu sa môže vyskytnúť pri inštalácii alebo integrácii
- Akceptačné testovanie nového rozšírenia funkcionality sa môže uskutočniť pred testovaním systému

V iteračnom vývoji môžu projektové tímy používať rôzne formy akceptačného testovania počas a na konci

každej iterácie, ako sú tie, ktoré sa zameriavajú na overenie novej funkcie v porovnaní s jej akceptačnými kritériami a tie, ktoré sú zamerané na overenie toho, že nová funkcia uspokojuje potreby používateľov. Okrem toho sa môže realizovať alfa testovanie a beta testovanie buď na konci každej iterácie, po dokončení každej iterácie alebo po sérii iterácií. Akceptačné testovanie používateľmi, prevádzkové akceptačné testovanie, regulačné akceptačné testovanie a zmluvné akceptačné testovanie sa môžu vyskytnúť buď na konci každej iterácie, po ukončení každej iterácie alebo po sérii iterácií.

2.3 Typy testovania

Typ testovania je skupina testovacích aktivít zameraná na testovanie špecifických vlastností softvérového systému alebo časti systému založená na špecifických cieľoch testovania. Tieto ciele môžu zahŕňať:

- Hodnotenie funkcionálnych kvalitatívnych charakteristík, ako sú úplnosť, správnosť a vhodnosť
- Hodnotenie nefunkcionálnych kvalitatívnych charakteristík, ako sú spoľahlivosť, výkonnosť, bezpečnosť, kompatibilita a použiteľnosť
- Hodnotenie, či je štruktúra alebo architektúra komponentu alebo systému správna, úplná a podľa špecifikácií
- Hodnotenie dôsledkov zmien, ako je potvrdenie toho, že boli odstránené defekty (potvrdzovacie testovanie) a hľadanie neúmyselných zmien v správaní v dôsledku zmeny softvéru alebo prostredia (regresné testovanie)

2.3.1 Funkcionálne testovanie

Funkcionálne testovanie systému zahŕňa testovanie, ktoré hodnotí funkcie, ktoré má systém vykonávať. Funkcionálne požiadavky môžu byť popísané v pracovných produktoch, ako sú špecifikácie biznis požiadaviek, popisy, používateľské príbehy, prípady použitia alebo funkcionálne špecifikácie alebo môžu byť nedokumentované. Funkcie sú „to“, čo by mal systém robiť.

Funkcionálne testy by sa mali vykonať na všetkých úrovniach testovania (napr. testovanie komponentov môže byť založené na špecifikácii komponentu), aj keď je zameranie na každej úrovni odlišné (pozri časť 2.2).

Funkcionálne testovanie uvažuje o správaní softvéru, takže môžu byť využité postupy čiernej skrinky na odvodenie testovacích podmienok a testovacích prípadov funkcionality komponentu alebo systému (pozri časť 4.2).

Dôkladnosť testovania funkcionality sa dá merať funkcionálnym pokrytím. Funkcionálne pokrytie je rozsah, v akom niektoré typy funkcionálnych prvkov boli vykonané testovaním a sú vyjadrené ako percentuálny podiel na type prvku, ktorý je pokrývaný. Napríklad použitím sledovateľnosti medzi testovaním a funkcionálnymi požiadavkami sa môže vypočítať percento týchto požiadaviek, ktoré sú predmetom testovania, a tak potenciálne identifikovať medzery pokrytia.

Návrh funkcionálneho testovania a jeho vykonanie môže vyžadovať špeciálne zručnosti alebo vedomosti, napríklad znalosť konkrétneho biznis problému, ktorý softvér rieši (napr. softvér geologického modelovania pre ropný a plynárenský priemysel) alebo konkrétnu úlohu, ktorú softvér vykonáva (napr. počítačový herný softvér poskytuje interaktívnu zábavu).

2.3.2 Nefunkcionálne testovanie

Nefunkcionálne testovanie systému hodnotí charakteristiky systémov a softvéru, ako je použiteľnosť, výkonnosť alebo bezpečnosť. Informácie o klasifikácii kvalitatívnych charakteristík softvérových produktov nájdete v norme ISO (ISO / IEC 25010). Nefunkcionálne testovanie je testovanie „ako dobre“ sa systém správa.

Na rozdiel od bežných chybných predstáv sa nefunkcionálne testovanie môže a často by aj malo vykonávať na všetkých úrovniach testovania a malo by sa realizovať čo najskôr. Neskorá identifikácia nefunkcionálnych defektov môže byť extrémne nebezpečná pre úspech projektu.

Na odvodenie testovacích podmienok a prípadov testovania pre nefunkcionálne testovanie sa môžu použiť techniky čiernej skrinky (pozri časť 4.2). Analýza hraničných hodnôt môže byť napríklad použitá na definovanie stresových podmienok pre testovanie výkonu.

Dôkladnosť nefunkcionálneho testovania sa môže merať prostredníctvom nefunkcionálneho pokrytia. Nefunkcionálne pokrytie je rozsah, v akom bol určitý typ nefunkcionálneho prvku vykonaný testovaním a je vyjadrený ako percentuálny podiel typu(-ov) prvku, ktorý je pokrytý. Napríklad pomocou sledovateľnosti medzi testami a podporovanými zariadeniami pre mobilnú aplikáciu možno vypočítať percento zariadení, ktoré sú riešené testovaním kompatibility a potenciálne identifikovať medzery pokrytia.

Návrh nefunkcionálneho testovania a samotné vykonávanie nefunkcionálneho testovania môže zahŕňať špeciálne zručnosti alebo znalosti, ako sú napríklad vedomosti o inherentných slabých miestach návrhu alebo technológie (napr. zraniteľnosť zabezpečenia spojená s konkrétnymi programovacími jazykmi) alebo konkrétnej základne používateľov (napr. osoby používateľov systémov riadenia zdravotníckych zariadení).

Ďalšie podrobnosti týkajúce sa testovania nefunkcionálnych kvalitatívnych charakteristík nájdete v ISTQB-ATA Advanced Level Test Analyst Syllabus, ISTQB-ATTA Advanced Level Technical Test Analyst Syllabus, ISTQB-SEC Advanced Level Security Tester Syllabus a ďalších špecializovaných moduloch ISTQB.

2.3.3 Testovanie bielej skrinky

Testovanie bielej skrinky odvodzuje testovanie založené na vnútornej štruktúre alebo implementácii systému. Vnútorná štruktúra môže zahŕňať kód, architektúru, pracovné toky a/alebo toky údajov v rámci systému (pozri časť 4.3).

Dôkladnosť testovania bielej skrinky je možné merať prostredníctvom štrukturálneho pokrytia. Štrukturálne pokrytie je miera, do akej bol určitý typ štrukturálneho prvku vykonaný testami, a je vyjadrený ako percentuálny podiel na pokrytom prvku.

Na úrovni testovania komponentov je pokrytie kódu založené na percentuálnej časti testovaného kódu komponentov, a môžu sa merať z hľadiska rôznych aspektov kódu (pokrytie položiek), ako je percento vykonateľných príkazov testovaných v komponente alebo percentuálny podiel testovaných výsledkov rozhodnutí. Tieto typy pokrytia sa kolektívne nazývajú pokrytia kódu. Na úrovni testovania integrácie komponentov môže byť testovanie bielej skrinky založené na architektúre systému, ako sú napríklad rozhrania medzi komponentmi a štrukturálne pokrytie môže byť merané z hľadiska percentuálneho podielu rozhraní vykonaných testami.

Návrh a vykonanie testovania bielej skrinky môže zahŕňať špeciálne zručnosti alebo vedomosti, ako napríklad spôsob, akým je kód zostavený (napr. používanie nástrojov na pokrytie kódu), ako sú dáta uložené (napr. vyhodnotiť možné databázové otázky) a ako používať nástroje pokrytia a správne interpretovať ich výsledky.

2.3.4 Testovanie súvisiace so zmenami

Pri vykonávaní zmien v systéme, či už kvôli oprave defektu alebo kvôli novým alebo meniacim sa funkcionalitám, je potrebné vykonať testovanie na potvrdenie toho, že zmeny opravili defekt alebo správne implementovali funkcionalitu a nevyvolali žiadne nepredvídateľné nepriaznivé následky.

- Potvrdzovacie (konfirmačné) testovanie: Po odstránení defektu môže byť softvér testovaný so všetkými prípadmi testovania, ktoré z dôvodu defektu zlyhali, ktoré by sa mali opätovne vykonať v novej verzii softvéru. Softvér môže byť tiež testovaný pomocou nových testov, ak bola defektom napríklad chýbajúca funkcionalita. Prinajmenšom je potrebné v novej verzii softvéru opätovne vykonať kroky na reprodukciu zlyhania (í) spôsobených defektom. Účelom potvrdzovacieho testovania je potvrdiť, či bol pôvodný defekt úspešne opravený.

- Regresné testovanie: Je možné, že zmena vykonaná v jednej časti kódu, či už oprava alebo iný typ zmeny, môže náhodne ovplyvniť správanie iných častí kódu, či už v rámci toho istého komponentu, v iných komponentoch toho istého systému alebo dokonca aj v iných systémoch. Zmeny môžu zahŕňať zmeny prostredia, napríklad novú verziu operačného systému alebo systému správy databáz. Takéto nezamýšľané vedľajšie účinky sa nazývajú regresie. Regresné testovanie zahŕňa spustenie testov na zistenie takýchto nezamýšľaných vedľajších účinkov.

Potvrdzovacie testovanie a regresné testovanie sa vykonávajú na všetkých úrovniach testovania.

Najmä v iteračných a inkrementálnych vývojových životných cykloch (napr. Agile) nové funkcie, zmeny existujúcich funkcií a refaktorovanie kódu vedú k častým zmenám kódu, čo tiež vyžaduje testovanie súvisiace so zmenami. Vzhľadom na evolúciu systému sú veľmi dôležité potvrdzovacie a regresné testovania. Je to dôležité najmä pre systémy Internetu vecí, kde sa jednotlivé objekty (napríklad zariadenia) často aktualizujú alebo nahrádzajú.

Regresné testovacie sady sú spustené mnohokrát a všeobecne sa vyvíjajú pomaly, takže regresné testovanie je silným kandidátom na automatizáciu. Automatizácia týchto testov by mala začať na začiatku projektu (pozri kapitolu 6).

2.3.5 Typy testovania a úrovne testovania

Je možné vykonávať ktorýkoľvek z vyššie uvedených typov testovania na akejkoľvek úrovni testovania. Na ilustráciu sa uvádzajú príklady funkcionálneho, nefunkcionálneho, testovania bielej skrinky a testovania súvisiaceho so zmenami na všetkých úrovniach testovania pre bankovú aplikáciu, počínajúc funkcionálnym testami:

- Pri testovaní komponentov sú testy navrhnuté na základe toho, ako by mal komponent kalkulovať zložený úrok.
- Pri testovaní integrácie komponentov sú testy navrhnuté na základe toho, ako sa informácie o účte zachytené v používateľskom rozhraní prenesú do biznis logiky.
- Pri systémovom testovaní sú testy navrhnuté na základe toho, ako môžu vlastníci účtov požiadať o úverovú linku na svojich bežných účtoch.
- Pri testovaní systémovej integrácie sú testy navrhnuté na základe toho, ako systém používa externé mikroservisy na kontrolu kreditného skóre vlastníka účtu.
- Pri akceptačných testoch sú testy navrhnuté na základe toho, ako bankár spracováva schvaľovanie alebo odmietnutie žiadosti o úver.

Nasledujú príklady nefunkcionálnych testov:

- Pri testovaní komponentov sú testy výkonnosti navrhnuté tak, aby vyhodnotili počet CPU cyklov potrebných na vykonanie komplexného výpočtu celkového úroku.
- Pri testovaní integrácie komponentov sú testy bezpečnosti navrhnuté na zraniteľnosť pretečenia vyrovnávacej pamäte v dôsledku údajov prenášaných z používateľského rozhrania do biznis logiky.
- Pri systémovom testovaní sú testy prenosnosti navrhnuté tak, aby skontrolovali, či prezentačná úroveň funguje na všetkých podporovaných prehliadačoch a mobilných zariadeniach.
- Pri testovaní systémovej integrácie sú testy spoľahlivosti navrhnuté tak, aby vyhodnotili robustnosť systému, ak mikroservis kreditného hodnotenia nereaguje.
- Pri akceptačnom testovaní sú testy použiteľnosti navrhnuté tak, aby vyhodnotili dostupnosť rozhrania spracovania kreditov bankou pre ľudí so zdravotným postihnutím.

Nasledujú príklady testovania bielej skrinky:

- Pri testovaní komponentov sú testy navrhnuté tak, aby sa dosiahli úplné pokrytie prehlásení a

rozhodnutí (pozri časť 4.3) pre všetky komponenty, ktoré vykonávajú finančné výpočty.

- Pri testovaní integrácie komponentov sú testy navrhnuté tak, aby každá obrazovka v rozhraní prehliadača preniesla údaje na ďalšiu obrazovku a do biznis logiky.
- Pri systémovom testovaní sú testy navrhnuté tak, aby zahŕňali sekvencie webových stránok, ktoré sa môžu vyskytnúť počas žiadosti o úverovú linku.
- Pri testovaní systémovej integrácie sú testy navrhnuté tak, aby vykonávali všetky možné typy dotazov posielané do mikroservisu kreditného hodnotenia.
- Pri akceptačnom testovaní sú testy navrhnuté tak, aby zahŕňali všetky podporované štruktúry súborov finančných údajov a rozsahy hodnôt pre prevody medzi bankami.

Nakoniec sú uvedené príklady testov súvisiacich so zmenami:

- Pri testovaní komponentov sú pre každý komponent vytvorené automatické regresné testy a sú zahrnuté do kontinuálneho integračného rámca.
- Pri testovaní integrácie komponentov sú testy navrhnuté tak, aby potvrdili opravy defektov súvisiacich s rozhraním, pretože opravy sú založené do úložiska kódov.
- Pri systémovom testovaní sa všetky testy pre daný pracovný tok (workflow) opätovne vykonajú, ak sa zmení niektorá obrazovka v tomto pracovnom toku (workflow).
- Pri testovaní integrácie systému sa denne opätovne testujú aplikácie interagujúce s mikroservisom kreditného hodnotenia ako súčasť nepretržitého nasadenia tohto mikroservise.
- Pri akceptačnom testovaní sa všetky skoršie zlyhané testy opätovne vykonajú po oprave defektu nájdeného v akceptačnom teste.

Kým v tejto časti sú uvedené príklady každého typu testovania na každej úrovni, nie je potrebné, aby bol pre každý softvér reprezentovaný každý typ testovania na všetkých úrovniach. Je však dôležité spustiť príslušné typy testovania na každej úrovni, najmä najskoršiu úroveň, kde sa vyskytuje typ testu.

2.4 Testovanie počas údržby

Po nasadení do produkčných prostredí je potrebné softvér a systémy udržiavať. Zmeny rôznych druhov sú v dodávanom softvéri a systémoch takmer nevyhnutné, a to buď na opravu defektov zistených v prevádzkovom použití, na pridanie nových funkcionalít alebo na vymazanie alebo zmenu už dodaných funkcionalít. Údržba je tiež potrebná na zachovanie alebo zlepšenie nefunkcionálnych kvalitatívnych charakteristík komponentu alebo systému počas jeho životnosti, najmä výkonnostnej činnosti, kompatibility, spoľahlivosti, bezpečnosti, a prenositeľnosti..

Ak sa vykonajú nejaké zmeny ako súčasť údržby, malo by sa vykonať testovanie počas údržby, aby sa vyhodnotil úspech, s akým boli zmeny vykonané, a aby sa overili možné vedľajšie účinky (napr. regres) v častiach systému, ktoré zostávajú nezmenené (ktorá je zvyčajne väčšina systému).
Testovanie počas údržby sa zameriava na testovanie zmien systému a testovanie nezmenených častí, ktoré by mohli byť zmenami ovplyvnené. Údržba môže zahŕňať plánované release a neplánované release (hotfix).

Release údržby môže vyžadovať testovanie počas údržby na viacerých úrovniach testovania s použitím rôznych typov testovania na základe jeho rozsahu. Rozsah testovania počas údržby závisí od:

- Stupeň rizika zmeny, napríklad stupeň, do akej zmenená oblasť softvéru komunikuje s inými komponentmi alebo systémami
- Veľkosť existujúceho systému
- Veľkosť zmeny

2.4.1 Spúšťače údržby

Existuje niekoľko dôvodov, prečo prebieha údržba softvéru, a tým aj testovanie počas údržby, a to pre plánované aj neplánované zmeny.

Spúšťače údržby môžeme klasifikovať nasledovne:

- Modifikácie, ako sú plánované vylepšenia (napr. na základe vydania), nápravné a núdzové zmeny, zmeny v prevádzkovom prostredí (napríklad plánované upgrady operačného systému alebo databázy), upgrady softvéru COTS a opravy chýb a zraniteľností
- Migrácia, napríklad z jednej platformy na druhú, ktorá môže vyžadovať prevádzkové testovanie nového prostredia, ako aj zmeneného softvéru, alebo testovanie konverzie dát, keď dáta migrujú z inej aplikácie do systému, ktorý je predmetom údržby
- Vyradenie, napríklad keď aplikácia dosiahne koniec životného cyklu

Keď dôjde k vyradeniu aplikácie alebo systému, môže to vyžadovať testovanie migrácie údajov alebo archivácie, ak sa vyžadujú dlhé obdobia uchovávania údajov. Môžu byť tiež potrebné aj testovania procedúr obnovy/záchrany po archivácii pre dlhé obdobia uchovávania. Okrem toho môže byť potrebné regresné testovanie, aby sa zabezpečilo, že všetky funkcionality, ktoré zostanú v prevádzke, ešte stále fungujú.

V prípade systémov Internetu vecí sa údržbové testovanie môže spustiť zavedením úplne nových alebo upravených vecí, ako sú hardvérové zariadenia a softvérové služby zahrnuté do celého systému. Údržbové testovanie pre takéto systémy kladie mimoriadny dôraz na integračné testovanie na rôznych úrovniach (napr. úroveň siete, úroveň aplikácie) a bezpečnostné aspekty, najmä tie, ktoré sa týkajú osobných údajov.

2.4.2 Analýza dopadov pre údržbu

Analýza dopadu hodnotí zmeny, ktoré boli vykonané pre release údržby, s cieľom identifikovať zamýšľané dôsledky, ako aj očakávané a možné vedľajšie účinky zmeny a určiť oblasti v systéme, ktoré budú zmenou ovplyvnené. Analýza dopadu môže tiež pomôcť identifikovať vplyv zmeny na existujúce testy. Vedľajšie efekty a ovplyvnené oblasti v systéme musia byť testované na regresiu, prípadne po aktualizácii akýchkoľvek existujúcich testov ovplyvnených touto zmenou.

Analýza dopadu sa môže vykonať pred vykonaním zmeny, aby to pomohlo rozhodovaniu, či sa má táto zmena vykonať, a to na základe možných následkov v iných oblastiach systému.

Analýza dopadu môže byť zložitá, ak:

- Špecifikácie (napr. biznis požiadavky, používateľské príbehy, architektúra) sú zastarané alebo chýbajú
- Prípady testovania nie sú zdokumentované alebo sú zastarané
- Obojsmerná sledovateľnosť medzi testami a základom testovania nebola dodržaná
- Podpora nástrojmi je slabá alebo neexistuje
- Zainteresovaní ľudia nemajú znalosti o doméne a/alebo systéme
- Počas vývoja sa udržovateľnosti softvéru venovala nedostatočná pozornosť

3 Statické testovanie — 135 minút

Kľúčové slová

ad hoc revízia, revízia založená na kontrolných zoznamoch, dynamické testovanie, formálna revízia, neformálna revízia, inšpekcia, čítanie založené na perspektíve, revízia, revízia založená na roliach, revízia založená na scenároch, statická analýza, statické testovanie, technická revízia, walkthrough

Študijné ciele pre statické testovanie

3.1 Základy statického testovania

FL-3.1.1 (K1) Rozpoznať softvérové pracovné produkty, ktoré môžu byť predmetom skúmania pomocou rôznych statických techník

FL-3.1.2 (K2) Na príkladoch vysvetliť užitočnosť statického testovania

FL-3.1.3 (K2) Vysvetliť rozdiel medzi statickými a dynamickými technikami s ohľadom na ciele, typy identifikovaných defektov a úlohu, ktorú tieto techniky zohrávajú v životnom cykle softvéru

3.2 Proces revízie

FL-3.2.1 (K2) Zhrnúť aktivity procesu revízie pracovného produktu

FL-3.2.2 (K1) Rozpoznať rôzne role a zodpovednosti vo formálnej revízii

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

FL-3.2.4 (K3) Aplikovať techniku revízie na pracovnom produkte za účelom identifikácie chýb

FL-3.2.5 (K2) Vysvetliť faktory spájajúce sa s úspešným procesom revízie

3.1 Základy statického testovania

Na rozdiel od dynamického testovania, ktoré vyžaduje spustenie softvéru, sú statické testovacie techniky založené na manuálnom preskúmaní pracovných produktov (napr. v prípade revízie) alebo na hodnotení kódu alebo iných pracovných produktov pomocou nástrojov (napr. pri statickej analýze). Oba typy statického testovania posudzujú kód alebo iný pracovný produkt, ktorý je predmetom testovania, bez reálneho spustenia testovaného kódu alebo pracovného produktu.

Statická analýza je obzvlášť dôležitá pre bezpečnostné počítačové systémy z odvetví kritickej infraštruktúry (napr. letecký, medicínsky alebo jadrový softvér), no stala sa aj bežnou súčasťou praxe v iných odvetviach (je napríklad dôležitou súčasťou testovania bezpečnosti). Statická analýza je často zahrnutá do automatizovaných buildov a dodávok, napríklad v agilnom vývoji, kontinuálnom dodávaní a nasadení.

3.1.1 Pracovné produkty, ktoré je možné preskúmať statickým testovaním

Takmer každý pracovný produkt môže byť predmetom skúmania pomocou statického testovania (revíziou a/alebo statickou analýzou).

Príklady:

- Špecifikácie vrátane biznis požiadaviek, funkcionálnych požiadaviek a bezpečnostných požiadaviek
- Epika, používateľské príbehy a akceptačné kritériá
- Architektúra a špecifikácie návrhov
- Kód
- Testvér zahrňujúci testovacie plány, testovacie prípady, testovacie procedúry a automatizované testovacie skripty
- Návody pre používateľov
- Webové stránky
- Zmluvy, projektové plány, rozvrhy a rozpočty
- Modely, ako sú diagramy aktivít, ktoré môžu byť použité na testovanie založené na modeloch (pozri ISTQB-MBT Foundation Level Model-Based Tester Extension Syllabus and Kramer 2016)

Revízie sa môžu použiť na akýkoľvek pracovný produkt, ktorý účastníci vedia čítať a pochopiť. Statická analýza môže byť efektívne aplikovaná na akýkoľvek pracovný produkt s formálnou štruktúrou (typicky kód alebo modely), pre ktorý existuje vhodný nástroj na statickú analýzu. Statická analýza môže byť dokonca aplikovaná pomocou nástrojov, ktoré vyhodnocujú pracovné produkty napísané v prirodzenom jazyku, ako sú požiadavky (napr. kontrola pravopisu, gramatiky a čitateľnosti).

3.1.2 Prínosy statického testovania

Techniky statického testovania poskytujú rôzne prínosy. Pri aplikácii na začiatku životného cyklu vývoja softvéru statické testovanie umožňuje skoré odhalenie chýb pred vykonávaním dynamického testovania (napr. pri revíziách požiadaviek alebo špecifikácií dizajnu, vylepšení nevybavených produktov atď.). Skoro odhalené defekty sa často oveľa lacnejšie odstraňujú, než defekty zistené neskôr v životnom cykle, najmä v porovnaní s defektmi, ktoré sa objavia po nasadení softvéru a pri jeho aktívnom používaní. Použitie techník statického testovania na zistenie defektov a následné opravenie týchto defektov je pre organizáciu takmer vždy omnoho lacnejšie, než pri použití dynamického testovania na nájdenie defektov v testovanom objekte a ich následnej oprave, najmä pri zohľadnení dodatočných nákladov spojených s aktualizáciou iných pracovných produktov a vykonanie potvrdzovacieho a regresného testovania.

Ďalšie prínosy statického testovania zahŕňajú:

- Efektívnejšiu detekciu a opravu defektov, a pred vykonaním dynamických testov
- Identifikáciu defektov, ktoré sa nedajú ľahko nájsť pomocou dynamického testovania
- Prevencia defektom v návrhoch alebo v kóde odhalením nezrovnalostí, nejasností, rozporov, vynechaní, nepresností a redundancií v požiadavkách
- Zvýšenie produktivity vývoja (napríklad vďaka zdokonalenému dizajnu, udržateľnejšiemu kódu)
- Zníženie nákladov a času na vývoj
- Zníženie nákladov a času na testovanie
- Zníženie celkových nákladov na kvalitu počas životnosti softvéru v dôsledku menšieho počtu zlyhaní v neskorších fázach životného cyklu alebo po uvedení produktu do prevádzky
- Zlepšenie komunikácie medzi členmi tímu počas účasti na revízii

3.1.3 Rozdiely medzi statickým a dynamickým testovaním

Statické testovanie a dynamické testovanie môžu mať rovnaké ciele (pozri oddiel 1.1.1). Príkladom je určovanie kvality pracovných produktov a včasná identifikácia defektov. Statické a dynamické testovanie odhaľuje rôzne typy defektov, čo zabezpečuje vzájomné dopĺňanie oboch typov (procesov).

Jedným z hlavných rozdielov je, že statické testovanie slúži na odhalenie defektov v pracovných produktoch skôr priamo, no neidentifikuje zlyhanie spôsobené defektmi počas spustenia softvéru. Defekt sa môže nachádzať v pracovnom produkte dlhší čas bez toho, aby spôsobil zlyhanie. Scenár, v ktorom sa defekt nachádza, môže byť zriedkavo spúšťaný alebo ťažko dostupný, takže nie je ľahké vytvoriť a vykonať dynamický test, ktorý ho identifikuje. Statické testovanie môže byť schopné nájsť tieto defekty s vynaložením oveľa menšej prácnosti.

Ďalším rozdielom je, že statické testovanie môže byť použité na zlepšenie konzistencie a vnútornej kvality pracovných produktov, zatiaľ čo dynamické testovanie sa zvyčajne zameriava na navonok viditeľné správanie.

V porovnaní s dynamickým testovaním, typické defekty, ktoré sú jednoduchšie a lacnejšie na identifikáciu a opravu pomocou statického testovania zahŕňajú:

- Defekty v požiadavkách (napr. nekonzistentnosti, nejednoznačnosti, protirečenia, opomenutia, nepresnosti a redundancie)
- Defekty návrhov (napr. neefektívne algoritmy alebo databázové štruktúry, vysoká väzba, nízka súdržnosť)
- Defekty v kóde (napr. premenné s nedefinovanými hodnotami, premenné, ktoré sú deklarované, ale nikdy nepoužívané, nedostupný kód, duplicitný kód)
- Odchýlky od štandardov (napríklad nedostačujúce dodržiavanie štandardov kódovania)
- Nesprávne špecifikácie rozhrania (napr. rôzne jednotky merania používané medzi volajúcim a volaným systémom)
- Bezpečnostné slabiny (napr. náchylnosť na pretečenie vyrovnávacej pamäte)
- Medzery alebo nepresnosti z hľadiska vysledovateľnosti alebo pokrytia testovacej bázy (napr. chýbajúce testy pre akceptačné kritérium)

Navyše väčšinu typov defektov udržovateľnosti možno identifikovať len statickým testovaním (napr. nesprávnu moduláciu, nedostatočnú opätovnú použiteľnosť komponentov, kód, ktorý je ťažké analyzovať a upraviť bez toho, aby došlo k zavlečeniu nových defektov do kódu).

3.2 Proces revízie

Existujú rôzne druhy revízií, od neformálnych po formálne. Neformálne revízie sú charakteristické tým, že nesledujú vopred definovaný proces a nemajú formálne zdokumentovaný výstup. Pre formálne revízie je naopak charakteristická tímová účasť, zdokumentovanie výsledkov a dokumentácia procesných postupov revízie. Formálnosť procesu revízie súvisí s faktormi ako sú model životného cyklu vývoja softvéru, zrelosť vývojového procesu, zložitosť pracovného produktu, ktorý je predmetom revízie, akékoľvek právne alebo regulačné požiadavky a/lebo nutnosť vytvorenia záznamu pre potreby auditu.

Účel revízie závisí od odsúhlasených cieľov revízie (napr. objavenie/nájdenie defektov, dosiahnutie porozumenia, vzdelávanie účastníkov ako sú testeri a noví členovia tímu, alebo diskusia a rozhodovanie na základe konsenzu).

Norma ISO (ISO / IEC 20246) obsahuje podrobnejšie opisy procesu posudzovania pracovných produktov vrátane rolí a revíznych techník.

3.2.1 Proces revízie pracovného produktu

Proces revízie zahŕňa tieto hlavné činnosti:

Plánovanie

- Definovanie rozsahu, ktorý zahŕňa účel revízie, aké dokumenty alebo časti dokumentov sú predmetom revízie a charakteristiky kvality, ktoré sú predmetom hodnotenia
- Odhad prácnosti a časového rámca
- Identifikácia charakteristík revízie, ako je napríklad typ revízie s rolami, aktivitami a kontrolnými zoznamami
- Výber účastníkov revízie a pridelenie rolí
- Definovanie vstupných a výstupných kritérií pre formálnejšie typy revízie (napr. inšpekcie)
- Kontrola splnenia vstupných kritérií (pre formálnejšie typy revízií)

Inicializácia revízie

- Distribúcia pracovného produktu (fyzicky alebo elektronicky) a iných materiálov, ako sú formuláre pre zápis nálezov, kontrolné zoznamy a súvisiace pracovné produkty
- Vysvetlenie rozsahu, cieľov, procesov, úloh a pracovných produktov účastníkom revízie
- Poskytnutie odpovedí na akékoľvek otázky, ktoré môžu mať účastníci ohľadom revízie

Individuálna revízia (t. j. individuálna príprava)

- Revízia celého pracovného produktu alebo jeho časti
- Zaznamenávanie možných defektov, odporúčaní a otázok

Komunikácia problému a analýza

- Diskusia nad identifikovanými potenciálnymi defektami (napr. počas revízneho stretnutia)
- Analýza potenciálnych defektov, priradenie vlastníka a stavu defektom
- Vyhodnotenie a zdokumentovanie charakteristík kvality
- Vyhodnotenie záverov revízie voči výstupným kritériám, a vykonanie rozhodnutia ohľadom záveru samotnej revízie (zamietnutie, sú potrebné závažnejšie zmeny; akceptácia, prípadne s menšími zmenami)

Oprava a reportovanie

- Vytváranie reportov o defektoch pre nálezy vyžadujúce zmeny
- Oprava zistených defektov (zvyčajne vykonaná autorom) v pracovnom produkte, ktorý je predmetom revízie
- Odkomunikovanie defektov zodpovednej osobe alebo tímu (ak sú objavené v pracovnom produkte súvisiacom s produktom, ktorý je predmetom revízie)
- Záznam aktualizovaného stavu defektov (v rámci formálnych revízií), prípadne vrátane súhlasu pôvodcu komentára
- Zhromažďovanie metrík (pre formálnejšie typy revízií)
- Kontrola splnenia výstupných kritérií (pre formálnejšie typy revízií)
- Akceptácia pracovného produktu pri dosiahnutí výstupných kritérií

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

3.2.2 Úlohy a zodpovednosti vo formálnej revízii

Typická formálna revízia zahŕňa nasledovné úlohy:

Autor

- Vytvára pracovný produkt, ktorý je predmetom revízie
- Opravuje defekty v pracovnom produkte, ktorý je predmetom revízie (ak je to potrebné)

Manažment

- Zodpovedá za plánovanie revízií
- Rozhoduje o vykonaní revízií
- Priraďuje personál, rozpočet a čas
- Monitoruje priebežnú efektívnosť nákladov
- Vykonáva kontrolné rozhodnutia v prípade neprimeraných výsledkov

Moderátor

- Zabezpečuje efektívny priebeh revíznych stretnutí (ak sú organizované)
- Funguje ako mediátor (ak je to potrebné) pri pluralite názorov
- Často je to osoba, na ktorej závisí úspech revízie

Líder revízie

- Preberá celkovú zodpovednosť za revíziu
- Rozhoduje, kto bude zapojený do revízie a určuje, kedy a kde sa bude konať

Revidujúci

- Môžu byť odborníkmi v danom odbore, osoby pracujúce na projekte, kľúčové osoby so záujmom o pracovný produkt a/alebo jednotlivci so špecifickým technickým alebo biznisovým vzdelaním (skúsenosťami)
- Identifikujú potenciálne defekty v pracovnom produkte, ktorý je predmetom revízie
- Môžu stelesňovať rôzne perspektívy (napr. testera, programátora, používateľa, operátora, biznis analytika, odborníka na použiteľnosť atď.).

Zapisovateľ (alebo zaznamenávateľ)

- Zhromažďuje záznamy o potenciálnych defektoch zistených počas individuálnych revízií
- Zaznamenáva nové potenciálne defekty, otvorené body a rozhodnutia vyplývajúce z revízneho stretnutia (ak sa organizuje)

U niektorých typov revízií môže jedna osoba reprezentovať viac ako jednu rolu a činnosti spojené s každou z rolí sa môžu líšiť v závislosti od typu revízie. Navyše, s príchodom nástrojov podporujúcich proces revízie (najmä zaznamenávanie defektov, otvorených bodov a rozhodnutí) nie je často potreba obsadenia role zapisovateľa.

Existuje aj detailnejšie rozdelenie úloh, ktorého popis obsahuje norma ISO (pozri ISO/IEC 20246).

3.2.3 Typy revízií

Aj keď revízie môžu byť použité na rôzne účely, jedným z hlavných cieľov je odhalenie defektov. Všetky typy revízií môžu pomôcť pri odhaľovaní defektov, no výber typu revízie by mal okrem iných kritérií zohľadňovať potreby projektu, dostupnosť zdrojov, typ produktu a riziká, biznis oblasti a kultúru spoločnosti.

Revízie je možné klasifikovať podľa rôznych atribútov. Nasledujúci zoznam obsahuje štyri najbežnejšie typy revízii a s nimi súvisiace atribúty.

Neformálna revízia (napr. buddy check, párovanie, pair review – vykonaná kolegom)

- Hlavný cieľ: detekcia potencionálnych defektov
- Ďalšie ciele: vytváranie nových nápadov alebo riešení, rýchle riešenie menších problémov
- Nie je založená na formálnom (zdokumentovanom) procese
- Nemusí zahŕňať revízne stretnutie
- Môže byť vykonávaná kolegom autora (buddy check) alebo viacerými ľuďmi
- Výsledky môžu byť zdokumentované
- Užitočnosť výsledku závisí od kvality revidujúcich
- Použitie kontrolných zoznamov je voliteľné
- Veľmi často sa používa v agilnom vývoji

Walkthrough

- Hlavné ciele: nájsť defekty, zlepšiť softvérový produkt, zvážiť alternatívne implementácie, zhodnotiť súlad so štandardami a špecifikáciami
- Ďalšie ciele: výmena názorov o rozličných technikách alebo štýloch, školenie účastníkov, dosiahnutie konsenzu
- Individuálna príprava pred revíznym stretnutím je nepovinná
- Revízne stretnutie je zvyčajne vedené autorom pracovného produktu
- Účasť zapisovateľa je povinná
- Použitie kontrolných zoznamov je voliteľné
- Môže mať formu scenárov, dry run (skúška „nanečisto“) alebo simulácií
- Výstupom môžu byť protokoly o potenciálnych defektoch a reporty o revízii
- V praxi sa môžu líšiť od relatívne neformálnych až po veľmi formálne

Technická revízia

- Hlavné ciele: dosiahnuť konsenzus, objaviť potenciálne defekty
- Ďalšie ciele: vyhodnotiť kvalitu a budovať dôveru v pracovný produkt, vytvárať nové myšlienky, motivovať autorov a umožniť im zlepšiť budúce pracovné produkty, zvážiť alternatívne implementácie
- Revidujúci by mali byť technickými spolupracovníkmi autora a technickými odborníkmi z rovnakej alebo ostatných disciplín
- Individuálna príprava pred revíznym stretnutím je vyžadovaná
- Revízne stretnutie je dobrovoľné, v ideálnom prípade je vedené zaškoleným moderátorom (zvyčajne nie autorom)
- Úloha zapisovateľa je povinná, v ideálnom prípade to nie je (danú rolu nevykonáva autor)
- Použitie kontrolných zoznamov je voliteľné
- Zvyčajne sa vytvárajú protokoly o potenciálnych defektoch a reporty o revízii

Inšpekcia

- Hlavné ciele: zisťovanie potenciálnych defektov, hodnotenie kvality a budovanie dôvery v pracovný produkt, prevencia pred podobnými defektmi prostredníctvom školenia autorov a analýzy prvotných príčin
- Možné ciele: motivácia a umožnenie autorom zlepšiť ich budúce pracovné produkty a proces vývoja softvéru, dosiahnutie konsenzu
- Postupuje sa podľa vopred definovaného procesu s formálne zdokumentovanými výstupmi založenými na pravidlách a kontrolných zoznamoch
- Využíva jasne definované role špecifikované v časti 3.2.2, ktoré sú povinné a môžu zahŕňať určeného čitateľa (ktorý prečíta pracovný produkt počas revízneho stretnutia nahlas)
- Individuálna príprava pred revíznym stretnutím je vyžadovaná
- Revidujúci sú buď kolegovia autora z odboru alebo odborníci v iných disciplínach, ktoré sú relevantné pre pracovný produkt
- Existujú vopred špecifikované vstupné a výstupné kritériá procesu

- Úloha zapisovateľa je povinná
- Revízne stretnutie vedie vyškolený moderátor (nie autor)
- Autor nemôže pôsobiť ako líder revízie, čitateľ alebo zapisovateľ
- Vytvoria sa záznamy potenciálnych defektov a report o revízii
- Zhromažďujú a používajú sa metriky na zlepšenie celého procesu vývoja softvéru vrátane procesu inšpekcie

Jeden pracovný produkt môže byť predmetom viac ako jedného typu revízie. Ak sa využíva viac ako jeden typ revízie, poradie sa môže líšiť. Napríklad, pred technickou revíziou možno vykonať neformálnu revíziu, aby sa zabezpečilo, že pracovný produkt je pripravený na technickú revíziu.

Typy revízií popísaných vyššie môžu byť vedené ako peer revízie, t. j. vykonané kolegami na podobnej alebo približnej organizačnej úrovni.

Typy defektov identifikovaných počas revízie sa líšia, najmä v závislosti od pracovného produktu, ktorý je predmetom revízie. Pozri časť 3.1.3 pre príklady defektov, ktoré možno nájsť v revíziách rôznych pracovných produktov . Pozri Gilb 1993 pre viac informácií o formálnych revíziách.

3.2.4 Použitie techník revízie

Existuje niekoľko techník revízie, ktoré možno použiť počas individuálnej revízie (t. j. individuálnej prípravy) na odhalenie defektov. Tieto techniky môžu byť použité naprieč jednotlivými typmi revízií. Účinnosť techník sa môže líšiť v závislosti od typu použitej revízie. Príklady rôznych individuálnych techník revízie pre rôzne typy revízií sú uvedené nižšie.

Ad hoc

Pri ad hoc revízii je revidujúcim poskytnuté málo pokynov alebo žiadne, na spôsob vykonania úlohy. Revidujúci často prechádzajú pracovný produkt postupne, identifikujú a dokumentujú problémy v poradí v ktorom ich nachádzajú. Revízia ad hoc je bežne používanou technikou, ktorá si vyžaduje minimálnu prípravu. Táto technika je vo veľkej miere závislá od zručností revidujúcich a z ich strany dôjsť k zvýšenému počtu duplicitných nálezov problémov.

Revízia na základe kontrolného zoznamu

Revízia založená na kontrolnom zozname je systematickou technikou, pri ktorej revidujúci odhaľujú problémy pomocou kontrolných zoznamov, ktoré sú distribuované pri začiatku procesu kontroly (napr. moderátorom). Kontrolný zoznam revízie pozostáva zo súboru otázok založených na potenciálnych defektoch, ktoré môžu byť založené na predošlých skúsenostiach. Kontrolné zoznamy by mali byť špecifické pre typ pracovného produktu, ktorý je predmetom revízie a mali by byť udržiavané tak, aby pokrývali typy problémov opomenutých pri predchádzajúcich revíziách. Hlavnou výhodou techniky založenej na kontrolnom zozname je systematické pokrytie typických defektov. Je potrebné dbať na to, aby revidujúci jednoducho nepostupovali pri jednotlivých revíziách podľa kontrolného zoznamu, ale tiež hľadali defekty mimo neho.

Scenáre a dry runy („nanečisto“)

V revíziách založených na scenári sú revidujúcim poskytované štruktúrované pokyny na čítanie pracovného produktu. Tento prístup podporuje revidujúcich pri vykonávaní dry runs na pracovnom produkte tým, že poskytuje revidujúcim očakávané scenáre použitia pracovného produktu (ak je pracovný produkt zdokumentovaný vo vhodnom formáte, ako sú prípady použitia). Tieto scenáre predávajú revidujúcim lepšie pokyny na identifikáciu konkrétnych typov defektov než jednoduché body kontrolného zoznamu. Tak ako pri revíziách založených na kontrolnom zozname, aby sa neopomenuli ďalšie typy defektov (napr. chýbajúce funkcie), revidujúci by v procese nemali byť obmedzovaní iba na zdokumentované scenáre.

Revízia založená na role

Revízia založená na role je technika, pri ktorej revidujúci hodnotia pracovný produkt z pohľadu rolí jednotlivých kľúčových osôb. Medzi typické roly patria konkrétne typy koncových používateľov (skúsení, neskúsení, starší, deti atď.) a špecifické roly v organizácii (správca, správca systému, performance tester atď.).

Revízia založená na perspektíve

Pri revízií založenej na perspektíve, podobne ako pri revízii založenej na roliach, revidujúci počas individuálnej revízie preberajú pohľady rôznych kľúčových osôb. Typické pozície kľúčových osôb zahŕňajú: pohľad koncového používateľa, marketingu, designéra, testera alebo prevádzky. Vďaka rozličným perspektívam dochádza k väčšiemu dôrazu na detail počas individuálnej revízie a k menšej duplicite hlásených problémov zo strany revidujúcich.

Navyše, čítanie založené na perspektíve tiež vyžaduje, aby sa revidujúci pokúsili používať revidovaný pracovný produkt na vytvorenie produktu, ktorý by z neho odvodili. Napríklad, tester by sa mal pokúsiť generovať návrhy akceptačných testov, ak vykonáva čítanie založené na perspektíve nad špecifikáciou požiadaviek, aby zistil, či v nej sú i zahrnuté všetky potrebné informácie. Okrem toho sa pri čítaní založenom na perspektíve očakáva, že sa použijú kontrolné zoznamy.

Empirické štúdie ukázali, že čítanie založené na perspektíve je najefektívnejšou všeobecnou technikou pre revíziu požiadaviek a technických pracovných produktov. Kľúčovým faktorom úspechu je primerané zohľadnenie rôznych názorov kľúčových osôb na základe rizík. Pozri Shul 2000, pre podrobnosti o čítaní založenom na perspektíve a Sauer 2000 pre efektívnosť rôznych typov revízií.

3.2.5 Faktory úspechu revízií

Pre realizáciu úspešnej revízie je nutné zvážiť vhodný typ revízie a použitých techník. Okrem toho existuje niekoľko ďalších faktorov, ktoré majú vplyv na výsledok revízie.

Organizačné faktory úspechu revízie zahŕňajú:

- Každá revízia má jasné ciele, definované počas plánovania revízie, ktoré sa používajú ako hodnoty pre výstupné kritériá
- Používajú sa typy revízií, ktoré sú vhodné na dosiahnutie cieľov revízie a sú vhodné pre typ a úroveň pracovných produktov a účastníkov
- Akékoľvek použitie revíznych techník, ako revízia založená na kontrolnom zozname alebo revízia založená na role, je vhodné na účinnú identifikáciu defektov v pracovnom produkte, ktorý je predmetom revízie
- Akékoľvek použité kontrolné zoznamy sú zacielené na hlavné riziká a sú aktuálne
- Rozsiahle dokumenty sú písané a revidované po malých častiach, takže kontrola kvality sa vykonáva tým, že sa autorom poskytne včasná a pravidelná spätná väzba o defektoch
- Účastníci majú dostatok času na prípravu
- Revízie sú plánované s dostatočným časovým predstihom
- Manažment poskytuje proces revízie (napr. zahrnutím primeraného času na revízie aktivít v rozvrhoch projektov)

Ľudské faktory úspechu revízie zahŕňajú:

- Na plnení cieľov revízie sa podieľajú správni ľudia, napríklad ľudia s rôznymi zručnosťami alebo vnímaním problému, ktorí môžu použiť dokument ako pracovný vstup pre svoju prácu
- Testeri sú považovaní za hodnotných revidujúcich, ktorí nielen prispievajú k úspechu revízie ale aj spoznávajú pracovný produkt, čo im umožňuje pripraviť efektívnejšie testy a pripraviť tieto testy skôr
- Účastníci venujú primeraný čas a pozornosť detailom
- Revízie sa vykonávajú po malých častiach, aby revidujúci počas individuálnej revízie a/alebo revízneho stretnutia (ak sa koná) nestratili koncentráciu
- Zistené defekty sú potvrdené, oceňované a narába sa s nimi objektívne
- Stretnutie je dobre zorganizované, takže účastníci ho považujú za cenné využitie svojho času
- Revízia sa uskutočňuje v atmosfére dôvery; jej výsledok sa nepoužije ako hodnotenie účastníkov
- Účastníci sa vyhýbajú reči tela a správaniu, ktoré môžu naznačovať nudu, rozčuľovanie alebo nepriateľstvo voči ostatným účastníkom
- Je poskytnutá adekvátna odborná príprava, najmä pri formálnych revíziách, ako sú napríklad inšpekcie
- Podporuje sa kultúra vzdelávania a zlepšovania procesov

Pozri Gilb 1993, Wiegers 2002, a van Veenendaal 2004 pre viac informácií o úspešných revíziách.

4 Testovacie techniky — 330 minút

Kľúčové slová

testovacia technika čiernej skrinky, analýza hraničných hodnôt, testovanie založené na kontrolných zoznamoch, pokrytie, pokrytie rozhodovania, testovanie rozhodovacích tabuliek, odhadovanie omylov, rozdelenie ekvivalencie, testovacia technika založená na skúsenostiach, prieskumné testovanie, testovanie prechodu stavov, pokrytie príkazov, testovacia technika, testovanie prípadov použitia, testovacia technika bielej skrinky

Študijné ciele pre testovacie techniky

4.1 Kategórie testovacích techník

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

4.2 Testovacie techniky čiernej skrinky

FL-4.2.1 (K3) Použiť rozdelenie ekvivalencie pre odvodenie testovacích prípadov od daných požiadaviek

FL-4.2.2 (K3) Použiť analýzu hraničných hodnôt pre odvodenie testovacích prípadov od daných požiadaviek

FL-4.2.3 (K3) Použiť testovanie rozhodovacích tabuliek pre odvodenie testovacích prípadov od daných požiadaviek

FL-4.2.4 (K3) Použiť testovanie prechodu stavov pre odvodenie testovacích prípadov od daných požiadaviek

FL-4.2.5 (K2) Vysvetliť ako odvodiť testovacie prípady od prípadu použitia

4.3 Testovacie techniky bielej skrinky

FL-4.3.1 (K2) Vysvetliť pokrytie príkazov

FL-4.3.2 (K2) Vysvetliť pokrytie rozhodovaní

FL-4.3.3 (K2) Vysvetliť hodnotu pokrytia príkazu a pokrytie rozhodovaní

4.4 Testovacie techniky založené na skúsenostiach

FL-4.4.1 (K2) Vysvetliť odhadovanie omylov

FL-4.4.2 (K2) Vysvetliť prieskumné testovanie

FL-4.4.3 (K2) Vysvetliť testovanie založené na kontrolných zoznamoch

4.1 Kategórie testovacích techník

Účelom testovacej techniky, vrátane tých, ktoré sú popísané v tejto časti, je pomôcť pri identifikácii testovacích podmienok, testovacích prípadov a testovacích údajov.

4.1.1 Výber testovacích techník

Výber testovacích techník, ktoré sa majú použiť, závisí od množstva faktorov vrátane nasledujúcich:

- Typ komponentu alebo systému
- Komplexnosť komponentu alebo systému
- Regulačné štandardy
- Požiadavky zákazníka alebo zmluvné požiadavky
- Úrovne rizika
- Typy rizík
- Ciele testovania
- Dostupná dokumentácia
- Znalosti a zručnosti testerov
- Dostupné nástroje
- Čas a rozpočet
- Model životného cyklu vývoja softvéru
- Predpokladané využitie softvéru
- Predchádzajúce skúsenosti s použitím testovacích techník na testovanom komponente alebo systéme
- Typy defektov očakávané v komponente alebo systéme

Niektoré techniky sú viac aplikovateľné na určité situácie a úrovne testov; iné sa vzťahujú na všetky úrovne testovania. Pri vytváraní testovacích prípadov používajú testeri všeobecne kombináciu testovacích techník na dosiahnutie najlepších výsledkov z prácnosti testovania.

Použitie testovacích techník v aktivitách analýzy pre testovanie, návrhu testovania a implementácie testovania sa môže pohybovať od veľmi neformálneho (malá až žiadna dokumentácia) až po veľmi formálne. Príslušná úroveň formálnosti závisí od kontextu testovania vrátane zrelosti testovacích a vývojových procesov, časových obmedzení, požiadaviek na bezpečnosť alebo reguláciu, znalostí a zručností zúčastnených osôb a dodržiavania modelu životného cyklu vývoja softvéru.

4.1.2 Kategórie testovacích techník a ich charakteristiky

V tejto učebnej osnove sú testovacie techniky klasifikované ako techniky čiernej skrinky, bielej skrinky alebo techniky založené na skúsenostiach.

Testovacie techniky čiernej skrinky (tiež nazývané behaviorálne techniky alebo techniky založené na správaní) sú založené na analýze vhodného základu testovania (napr. dokumenty formálnych požiadaviek, špecifikácie, prípady použitia, používateľské príbehy alebo biznis procesy). Tieto techniky sa uplatňujú na funkcionálne aj nefunkcionálne testovanie. Testovacie techniky čiernej skrinky sa sústreďujú na vstupy a výstupy testovaného objektu bez odkazu na jeho vnútornú štruktúru.

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

Testovacie techniky založené na skúsenostiach využívajú skúsenosti vývojárov, testerov a používateľov pri navrhovaní, implementácii a vykonávaní testov. Tieto techniky sa často kombinujú s testovacími technikami čiernej a bielej skrinky.

Spoločné charakteristiky testovacích techník čiernej skrinky sú nasledovné:

- Testovacie podmienky, testovacie prípady a testovacie dáta sú odvodené od základu testovania, ktorý môže zahŕňať softvérové požiadavky, špecifikácie, prípady použitia a používateľské príbehy.
- Testovacie prípady sa môžu použiť na zisťovanie nedostatkov medzi požiadavkami a implementáciou požiadaviek, ako aj odchýlok od požiadaviek.
- Pokrytie sa meria na základe položiek testovaných v testovacom základe a techniky, ktorá sa aplikuje na základ testovania.

Spoločné charakteristiky testovacích techník bielej skrinky sú nasledovné:

- Testovacie podmienky, testovacie prípady a testovacie dáta sú odvodené od základu testovania, ktorý môže zahŕňať kód, architektúru softvéru, detailný dizajn alebo akýkoľvek iný zdroj informácií o štruktúre softvéru.
- Pokrytie sa meria na základe položiek testovaných v rámci vybranej štruktúry (napr. kód alebo rozhranie).
- Špecifikácie sa často používajú ako dodatočný zdroj informácií na určenie očakávaného výsledku testovacích prípadov.

Spoločné charakteristiky testovacích techník založených na skúsenostiach sú nasledovné:

- Testovacie podmienky, testovacie prípady a testovacie dáta sú odvodené zo základu testovania, ktorý môže zahŕňať znalosti a skúsenosti testerov, vývojárov, používateľov a iných kľúčových osôb

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

Medzinárodný štandard (ISO/IEC/IEEE 29119-4) obsahuje popisy testovacích techník a zodpovedajúce opatrenia pokrytia (pozri Craig 2002 a Copeland 2004 pre viac informácií o technikách).

4.2 Testovacie techniky čiernej skrinky

4.2.1 Rozdelenie ekvivalencie

Rozdelenie ekvivalencie rozdeľuje dáta na sekcie (známe tiež ako triedy ekvivalencie) pričom sa očakáva, že všetci členovia danej sekcie budú spracovávaní rovnakým spôsobom (pozri Kaner 2013 a Jorgensen 2014). Existujú sekcie ekvivalencie pre platné aj neplatné hodnoty.

- Platné hodnoty sú hodnoty, ktoré by mali komponent alebo systém akceptovať. Oblasť ekvivalencie obsahujúca platné hodnoty sa nazýva „platná sekcia ekvivalencie“.
- Neplatné hodnoty sú hodnoty, ktoré by mali komponent alebo systém odmietnuť. Oblasť ekvivalencie obsahujúca neplatné hodnoty sa nazýva „neplatná sekcia ekvivalencie“.
- Sekcie je možné identifikovať pre akýkoľvek údajový prvok súvisiaci s testovaným objektom vrátane vstupov, výstupov, interných hodnôt, hodnôt súvisiacich s časom (napr. pred alebo po udalosti) a parametrov rozhrania (napr. integrované komponenty testované počas integračného testovania).
- Každú sekciu je možné v prípade potreby rozdeliť na podsekcie.
- Každá hodnota musí patriť k jednej a jedinej sekcii ekvivalencie.
- Ak sa v testovacích prípadoch používajú neplatné sekcie ekvivalencie, musia byť testované jednotlivo, t. j. nie v spojení s inými neplatnými sekciami ekvivalencie, aby sa zabezpečilo, že zlyhania nie sú maskované. Zlyhania môžu byť maskované, keď sa vyskytnú viaceré poruchy súčasne, ale iba jedna je viditeľná, čo spôsobí, že ostatné poruchy nebudú identifikované.

Pre dosiahnutie 100 % pokrytia touto technikou, testovacie prípady musia pokryť všetky identifikované sekcie (vrátane neplatných sekcií) použitím minimálne jednej hodnoty z každej sekcie. Pokrytie sa meria ako počet sekcií ekvivalencie testovaných aspoň jednou hodnotou, vydelenej celkovým počtom identifikovaných sekcií ekvivalencie, zvyčajne vyjadrených ako percento. Rozdelenie ekvivalencie sa uplatňuje na všetkých testovacích úrovniach.

4.2.2 Analýza hraničných hodnôt

Analýza hraničných hodnôt (BVA) je rozšírením rozdelenia ekvivalencie, ale môže byť použitá len pri zoradení sekcií, pozostávajúceho z číselných alebo sekvenčných údajov. Minimálne a maximálne hodnoty (alebo prvé a posledné hodnoty) sekcie sú jej hraničnými hodnotami (Beizer 1990).

Predpokladajme napríklad, že vstupné pole akceptuje ako vstup jeden celočíselný údaj, pomocou klávesnice pre obmedzenie vstupov, takže nie je možné zadávať výlučne celé čísla. Platný rozsah je od 1 do 5 vrátane. Existujú teda tri sekcie ekvivalencie: neplatné (príliš nízke); platné; neplatné (príliš vysoké). Pre platnú sekciu ekvivalencie sú hraničné hodnoty 1 a 5. Pre neplatnú (príliš vysokú) sekciu sú hraničné hodnoty 6 a 9. Pre neplatnú (príliš nízku) sekciu existuje iba jedna hraničná hodnota, t. j. 0, pretože toto je oblasť s jedným členom.

V príklade vyššie identifikujeme dve hraničné hodnoty na hranici. Hranica medzi neplatnou (príliš nízkou) a platnou udáva testovacie hodnoty 0 a 1. Hranica medzi platnou a neplatnou (príliš vysoká) udáva testovacie hodnoty 5 a 6. Niektoré varianty tejto techniky identifikujú tri hraničné hodnoty na hranicu: hodnoty pred, na a tesne nad hranicou. V predchádzajúcom príklade s použitím trojbodových hraničných hodnôt sú spodné hraničné testovacie hodnoty 0, 1 a 2 a horné hraničné testovacie hodnoty sú 4, 5 a 6 (Jorgensen 2014).

Správanie na hraniciach sekcií ekvivalencie je pravdepodobnejšie nesprávne ako správanie vo vnútri sekcií. Je dôležité mať na pamäti, že obidve špecifikované a implementované hranice môžu byť presunuté do pozícií nad alebo pod ich zamýšľanými pozíciami, môžu byť úplne vynechané alebo môžu byť doplnené nežiadúcimi ďalšími hranicami. Analýza a testovanie hraničných hodnôt odhalí takmer všetky takéto defekty tým, že prinúti softvér, aby ukázal správanie z inej sekcie, než je tá, ktorej by hraničná hodnota mala patriť.

Analýza hraničných hodnôt sa môže použiť na všetkých úrovniach testovania. Táto technika sa všeobecne používa na testovanie požiadaviek, ktoré vyžadujú rozsah čísel (vrátane dátumov a časov). Hraničné pokrytie sekcie sa meria ako počet testovaných hraničných hodnôt vydelený celkovým počtom identifikovaných hraničných testovacích hodnôt, normálne vyjadrený ako percento.

4.2.3 Testovanie rozhodovacích tabuliek

Kombinatorické testovacie techniky sú užitočné pri testovaní implementácie systémových požiadaviek, ktoré určujú, ako rôzne kombinácie podmienok vedú k rôznym výsledkom. Jedným z prístupov k takýmto testom je testovanie rozhodovacích tabuliek.

Rozhodovacie tabuľky sú dobrým spôsobom na zaznamenávanie zložitých biznis pravidiel, ktoré musí systém implementovať. Pri vytváraní rozhodovacích tabuliek tester určuje podmienky (často vstupy) a výsledné činností (často výstupy) systému. Tie tvoria riadky tabuľky, zvyčajne s podmienkami v hornej časti a činnosťami v dolnej časti. Každý stĺpec zodpovedá rozhodovaciemu pravidlu, ktoré definuje jedinečnú kombináciu podmienok, ktoré vedú k vykonaniu činností spojených s týmto pravidlom. Hodnoty podmienok a činností sa zvyčajne zobrazujú ako booleovské hodnoty (pravdivé alebo nepravdivé) alebo diskrétne hodnoty (napr. červená, zelená, modrá), ale môžu to byť aj čísla alebo rozsahy čísel. Tieto rôzne typy podmienok a činností možno nájsť spolu v rovnakej tabuľke.

Spoločná notácia v rozhodovacích tabuľkách je nasledujúca:

Pre podmienky:

- Y znamená, že podmienka je pravdivá (môže byť vyjadrené tiež ako T alebo 1)
- N znamená, že podmienka je nepravdivá (môže byť vyjadrené tiež ako F alebo 0)
- — znamená, že na hodnote podmienky nezáleží (môže byť vyjadrené tiež ako N/A) Pre

činnosti:

- X znamená, že akcia by mala nastať (môže byť vyjadrené tiež ako Y alebo T alebo 1)
- Prázdne miesto znamená, že akcia by nemala nastať (môže byť vyjadrené tiež ako N alebo F alebo 0)

Celá rozhodovacia tabuľka má dostatok stĺpcov na pokrytie každej kombinácie podmienok. Tabuľku je možné zjednodušiť odstránením stĺpcov, ktoré obsahujú nemožné kombinácie podmienok, stĺpce obsahujúce možné, ale neuskutočniteľné kombinácie podmienok a stĺpce, ktoré testujú kombinácie podmienok, ktoré neovplyvňujú výsledok. Ďalšie informácie o tom, ako zjednodušiť rozhodovacie tabuľky, nájdete v osnove ISTQB-ATA Advanced Level Analyst.

Spoločným minimálnym štandardom pokrytia pre testovanie rozhodovacích tabuliek je, aby v tabuľke bol aspoň jeden testovací prípad na pravidlo rozhodovania. To zvyčajne zahŕňa pokrytie všetkých kombinácií podmienok. Pokrytie sa meria ako počet rozhodovacích pravidiel testovaných najmenej jedným testovacím prípadom, vydelený celkovým počtom rozhodovacích pravidiel, zvyčajne vyjadrený ako percento.

Výhodou testovania rozhodovacích tabuliek je, že pomáha identifikovať všetky dôležité kombinácie podmienok, z ktorých niektoré by sa inak mohli prehliadnuť. Umožňuje tiež nájsť medzery v požiadavkách. Môže sa uplatniť na všetky situácie, v ktorých správanie softvéru závisí od kombinácie podmienok na akejkoľvek úrovni testu.

4.2.4 Testovanie prechodu stavov

Komponenty alebo systémy môžu reagovať odlišne na udalosť v závislosti od aktuálnych podmienok alebo predchádzajúcej histórie (napr. udalosti, ku ktorým došlo od inicializácie systému). Predchádzajúcu históriu možno zhrnúť pomocou konceptu stavov. Diagram prechodu stavov zobrazuje možné stavy softvéru, ako aj to, ako softvér vstupuje, vystupuje a prechádza medzi jednotlivými stavmi. Prechod je iniciovaný udalosťou (napr. zadanie hodnoty do poľa používateľom). Výsledkom udalosti je prechod. Ak tá istá udalosť môže viesť k dvom alebo viacerým rôznym prechodom z toho istého stavu, táto udalosť môže byť podmienená ochrannou podmienkou. Zmena stavu môže viesť k tomu, že softvér vykoná akciu (napr. výstup výpočtu alebo chybové hlásenie).

Tabuľka prechodov stavov zobrazuje všetky platné prechody a potenciálne neplatné prechody medzi stavmi, ako aj udalosti, ochranné podmienky a výsledné činnosti pre platné prechody. Diagramy prechodu stavov zvyčajne zobrazujú iba platné prechody a nezobrazujú neplatné prechody.

Testy môžu byť navrhnuté tak, aby zahŕňali typickú sekvenciu stavov, aby mohli vykonávať všetky stavy, vykonávať každý prechod, vykonávať špecifické sekvencie prechodov alebo testovať neplatné prechody.

Testovanie prechodu stavov sa používa pre aplikácie založené na menu a je široko používané v rámci integrovaného softvérového priemyslu. Táto technika je tiež vhodná na modelovanie biznis scenára s konkrétnymi stavmi alebo na testovanie navigácie na obrazovke. Koncept stavu je abstraktný - môže predstavovať niekoľko riadkov kódu alebo celý biznis proces.

Pokrytie sa bežne meria ako počet testovaných identifikovaných stavov alebo prechodov vydelený celkovým počtom identifikovaných stavov alebo prechodov v testovanom objekte, normálne vyjadrený ako percento. Viac informácií o kritériách pokrytia testovania prechodu stavov nájdete v osnove ISTQB-ATA Advanced Level Analyst.

4.2.5 Testovanie prípadov použitia

Testovacie prípady môžu byť odvodené z prípadov použitia, ktoré sú špecifickým spôsobom navrhovania interakcií so softvérovými položkami a obsahujú požiadavky na softvérové funkcie, ktoré predstavujú prípady použitia. Prípady použitia sú spojené s činiteľmi (účastníkmi - osobami - používateľmi, externým hardvérom alebo inými komponentmi alebo systémami) a subjektmi (komponentami alebo systémami, na ktoré sa aplikuje prípad používania).

Každý prípad použitia určuje určité správanie, ktoré môže subjekt vykonávať v spolupráci s jedným alebo viacerými činiteľmi (účastníkmi) (UML 2.5.1 2017). Prípad použitia možno opísať interakciami a aktivitami, ako aj predpokladmi, podmienkami a prirodzeným jazykom tam, kde je to vhodné. Interakcie medzi činiteľmi (účastníkmi) a subjektom môžu viesť k zmenám stavu subjektu. Interakcie môžu byť reprezentované graficky pracovnými tokmi (workflows), diagramami aktivít alebo modelmi biznis procesov.

Prípad použitia môže zahŕňať možné varianty jeho základného správania vrátane výnimočného správania a riešenia chýb (reakcia systému a obnovenie v prípade chýb programovania, aplikácie a komunikácie, ktoré napr. majú za následok chybové hlásenie). Testy sú navrhnuté tak, aby vykonávali definované správanie (základné, výnimočné alebo alternatívne a riešenie chýb). Pokrytie sa môže merať podľa percentuálneho podielu prípadov použitia testovaného správania vydeleného celkovým počtom špecifikovaných prípadov použitia správania, pričom sa zvyčajne vyjadruje ako percento. Viac informácií o kritériách pokrytia pre testovanie prípadov použitia nájdete v osnove ISTQB-ATA Advanced Level Analyst.

4.3 Testovacie techniky bielej skrinky

Testovanie bielej skrinky je založené na vnútornej štruktúre testovaného objektu. Testovacie techniky bielej skrinky sa môžu používať vo všetkých testovacích úrovniach, ale dve techniky súvisiace s kódmi, popísané v tejto časti, sú najčastejšie používané na testovacej úrovni komponentov. Existujú pokročilejšie techniky, ktoré sa používajú v niektorých prostrediach kritických z hľadiska bezpečnosti, poslania alebo prostrediach vysokej integrity, aby sa dosiahlo dôkladnejšie pokrytie, avšak tieto sa tu nepopisujú. Viac informácií o týchto technikách nájdete v osnove ISTQB Advanced Technical Analyst.

4.3.1 Testovanie a pokrytie príkazov

Testovanie príkazov vykonáva spustiteľné príkazy v kóde. Pokrytie sa meria ako počet príkazov vykonaných testovacími prípadmi vydelený celkovým počtom spustiteľných výkazov v testovanom objekte, bežne vyjadrené ako percento.

4.3.2 Testovanie a pokrytie rozhodovaní

Testovanie rozhodovaní vykonáva rozhodnutia v kóde a testuje kód, ktorý sa vykonáva na základe výsledkov rozhodnutia. Aby to nastalo, testovacie prípady sa riadia riadiacimi tokmi, ktoré sa vyskytujú z rozhodovacieho bodu (napr. pre IF príkaz, jeden pre pravdivý výstup a jeden pre nepravdivý výstup, pre príkaz CASE by sa vyžadovali testovacie prípady pre všetky možné výstupy vrátane predvoleného výstupu).

Pokrytie sa meria ako počet rozhodovacích výstupov vykonaných testovacími prípadmi vydelený celkovým počtom rozhodovacích výstupov v testovanom objekte, normálne vyjadrený ako percento.

4.3.3 Hodnota testovania príkazov a rozhodovaní

Keď sa dosiahne 100 % pokrytie príkazov, zabezpečí sa tým, že všetky spustiteľné príkazy v kóde boli testované aspoň raz, ale nezabezpečuje sa tým, že bola testovaná celá logika rozhodovania. Z dvoch techník bielej skrinky, o ktorých sa hovorí v tejto osnove, môže mať testovanie príkazov menšie pokrytie ako pri testovaní rozhodovaní.

Keď sa dosiahne 100 % pokrytie rozhodovaní, vykonajú sa tým všetky rozhodovacie výstupy, čo zahŕňa testovanie pravdivého výstupu a tiež nepravdivého výstupu, aj keď neexistuje žiadny explicitný nepravdivý príkaz (napr. v prípade príkazu IF bez ELSE prechodu). Pokrytie príkazov pomáha nájsť chyby v kóde, ktorý nebol vykonaný inými testovacími prípadmi. Pokrytie rozhodovania pomáha nájsť chyby v kóde, kde iné testovacie prípady nezačlenili pravdivé a nepravdivé výstupy.

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

4.4 Testovacie techniky založené na skúsenosti

Pri použití techník založených na skúsenostiach sú testovacie prípady odvodené od zručnosti a intuície testerov a ich skúseností s podobnými aplikáciami a technológiami. Tieto techniky môžu byť užitočné pri identifikácii testovacích prípadov, ktoré neboli ľahko identifikovateľné inými systematickejšími technikami. V závislosti od prístupu a skúseností testera môžu tieto techniky dosiahnuť veľmi rozdielne stupne pokrytia a efektívnosti. Pokrytie môže byť ťažké posúdiť a nemusí byť merateľné pomocou týchto techník.

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

4.4.1 Odhadovanie omylov

Odhadovanie omylov je technika, ktorá sa používa na predvídanie výskytu chýb, defektov a porúch na základe znalostí testera, vrátane:

- Ako fungovala aplikácia v minulosti
- Aké typy chýb majú vývojári tendenciu robiť
- Zlyhania, ku ktorým došlo v iných aplikáciách

Metodickým prístupom k metóde odhadovania chýb je vytvorenie zoznamu možných chýb, defektov a zlyhaní a návrhov testov, ktoré odhalia tieto zlyhania a defekty, ktoré ich spôsobili. Tieto zoznamy omylov, defektov a zlyhaní môžu byť vytvorené na základe skúseností, údajov o defektoch a zlyhaniach alebo z bežných vedomostí o tom, prečo softvér zlyháva.

4.4.2 Prieskumné testovanie

Pri prieskumnom testovaní sú neformálne (nie preddefinované) testy navrhnuté, vykonané, zaznamenané a vyhodnotené dynamicky počas vykonávania testu. Výsledky testov sa používajú na získanie ďalších informácií o komponente alebo systéme a na vytvorenie testov pre oblasti, ktoré môžu potrebovať ďalšie testovanie.

Prieskumné testovanie sa niekedy vykonáva pomocou testovania založeného na sedeniach na štruktúrovanie aktivity. Pri testovaní založeného na úsekoch sa prieskumné testovanie uskutočňuje v rámci definovaného časového rámca a tester používa testovaciu chartu obsahujúcu testovacie ciele na usmerňovanie testovania. Tester môže použiť listy testovania založeného na úsekoch na zdokumentovanie krokov, ktoré boli vykonané, a zaznamenanie zistení.

Prieskumné testovanie je najužitočnejšie, ak existuje málo alebo nedostatočný počet špecifikácií alebo významný časový tlak na testovanie. Prieskumné testovanie je užitočné aj na doplnenie ďalších formálnych testovacích techník.

Prieskumné testovanie je silne spojené s reaktívnymi testovacími stratégiami (pozri časť 5.2.2). Prieskumné testovanie môže zahŕňať použitie iných techník čiernej skrinky, bielej skrinky a techník založených skúsenostiach.

4.4.3 Testovanie založené na kontrolných zoznamoch

V testoch založených na kontrolných zoznamoch testeri navrhujú, implementujú a vykonávajú testy na pokrytie testovacích podmienok nachádzajúcich sa v kontrolných zoznamoch. V rámci analýzy testeri vytvárajú nový kontrolný zoznam alebo rozširujú existujúci kontrolný zoznam, pričom testeri môžu tiež použiť existujúci kontrolný zoznam bez akejkoľvek zmeny. Takéto kontrolné zoznamy môžu byť zostavené na základe skúseností, vedomostí o tom, čo je pre používateľa dôležité, alebo pochopení toho, prečo a ako softvér zlyháva.

Kontrolné zoznamy môžu byť vytvorené na podporu rôznych typov testov vrátane funkcionálnych a nefunkcionálnych testov. Ak chýbajú podrobné testovacie prípady, testovanie na základe kontrolných

zoznamov môže poskytnúť usmernenia a určitý stupeň konzistencie. Keďže sa jedná o zoznamy na vysokej úrovni, je pravdepodobné, že dôjde k nejakej variabilite pri skutočnom testovaní, čo môže mať za následok potenciálne väčšie pokrytie, ale menšiu opakovateľnosť.

5 Manažment testovania — 225 minút

Kľúčové slová

konfiguračný manažment, manažment defektov, vstupné kritériá, výstupné kritéria, produktové riziko, projektové riziko, riziko, úroveň rizika, testovanie založené na riziku, testovací prístup, riadenie testovania, odhadovanie testovania, vedúci testovania, monitorovanie testovania, plán testovania, plánovanie testovania, report o pokroku testovania, stratégia testovania, súhrnný report z testovania, tester

Študijné ciele pre manažment testovania

5.1 Organizácia testovania

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

5.1.2 (K1) Identifikovať úlohy vedúceho testovania a testera

5.2 Plánovanie a odhadovanie testovania

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

FL-5.2.2 (K2) Rozlišovať medzi rôznymi stratégiami testovania

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

FL-5.2.4 (K3) Aplikovať znalosti o stanovení priorít a technických a logických vzťahoch, aby ste vytvorili rozvrh vykonávania testovania pre danú sadu testovacích prípadov

FL-5.2.5 (K1) Identifikovať faktory, ktoré ovplyvňujú prácnosť spojenú s testovaním

FL-5.2.6 (K2) Vysvetliť rozdiel medzi dvoma technikami odhadovania: technika založená na metrikách a technika založená na expertnom prístupe

5.3 Monitorovanie a riadenie testovania

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

FL-5.3.2 (K2) Zhrnúť účel, obsah a cieľové skupiny pre report 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 použitím pravdepodobnosti (výskytu) a dopadu

FL-5.5.2 (K2) Rozlíšiť projektové a produktové riziká

FL-5.5.3 (K2) Popísať, s použitím príkladov, ako môže analýza produktového rizika ovplyvniť dôkladnosť a rozsah testovania

5.6 Manažment defektov

FL-5.6.1 (K3) Napísať report o defektoch, pokrývajúce defekty identifikované počas testovania

5.1 Organizácia testovania

5.1.1 Nezávislé testovanie

Úlohy testovania môžu vykonávať osoby v konkrétnej testovacej role alebo osoby v inej role (napr. zákazníci). Určitý stupeň nezávislosti často robí testera efektívnejším pri hľadaní defektov v dôsledku rozdielov medzi kognitívnymi predsudkami autora a testera (pozri časť 1.5). Nezávislosť však nie je náhradou znalosti a vývojári môžu efektívne nájsť veľa defektov vo svojom vlastnom kóde.

Stupne nezávislosti testovania zahŕňajú nasledovné (od nízkej až po vysokú úroveň nezávislosti):

- Žiadni nezávislí testeri. Jedinou dostupnou formou testovania sú vývojári, ktorí testujú svoj vlastný kód
- Nezávislí vývojári alebo testeri v rámci vývojových tímov alebo projektového tímu. Môže ísť o vývojárov testujúcich produkty svojich kolegov
- Nezávislý testovací tím alebo skupina v rámci organizácie, podriadený projektovému alebo výkonnému manažmentu
- Nezávislí testeri z biznis organizácie alebo používateľskej komunity alebo so špecializáciou na špecifické typy testov, ako sú použiteľnosť, bezpečnosť, výkonnosť, súlad so zákonmi/nariadeniami alebo prenositeľnosť
- Nezávislí testeri, ktorí sú z externého prostredia mimo organizácie, pracujú on-site (insourcing) alebo off-site (outsourcing)

Pre väčšinu typov projektov je zvyčajne najlepšie mať viacero úrovní testovania, pričom niektoré z týchto úrovní sú vykonávané nezávislými testermi. Vývojári by sa mali zúčastňovať testovania, najmä na nižších úrovniach, aby mohli vykonávať kontrolu nad kvalitou vlastnej práce.

Spôsob implementácie nezávislosti 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 metódy, môžu byť títo testeri tiež považovaní za súčasť väčšieho nezávislého testovacieho tímu. Okrem toho môžu vlastníci produktov v takýchto organizáciách vykonať akceptačné testovanie na overenie používateľských príbehov na konci každej iterácie.

Potenciálne výhody nezávislosti testovania zahŕňajú:

- Nezávislí testeri pravdepodobne rozpoznávajú iné druhy zlyhaní v porovnaní s vývojármi z dôvodu ich odlišného pozadia, technických perspektív a predsudkov
- Nezávislý tester môže overiť, napadnúť alebo vyvrátiť predpoklady zo strany kľúčových osôb počas špecifikácie a implementácie systému

Potenciálne nevýhody nezávislosti testovania zahŕňajú:

- Izolácia od vývojového tímu, ktorá vedie k nedostatočnej spolupráci, oneskoreniam pri poskytovaní spätnej väzby vývojovému tímu alebo k nepriateľskému vzťahu s vývojovým tímom
- Vývojári môžu stratiť zmysel zodpovednosti za kvalitu
- Nezávislí testeri môžu byť považovaní za úzke hrdlo alebo vinení za omeškania pri uvoľnení dodávky softvéru
- Nezávislým testerom môžu chýbať niektoré dôležité informácie (napr. o testovanom objekte)

Mnohé organizácie dokážu úspešne dosiahnuť výhody nezávislosti testovania a pritom sa vyhnúť nevýhodám.

5.1.2 Úlohy vedúceho testovania a testera

Táto učebná osnova sa zaoberá dvomi testovacími rolami, vedúci testovania a tester. Aktivity a úlohy vykonávané ľuďmi v týchto dvoch rolách sú závislé na kontexte projektu a produktu, kvalifikácii ľudí v daných rolách a organizácii.

Úlohou vedúceho testovania je celková zodpovednosť za proces testovania a úspešné vedenie aktivít testovania. Rola vedúceho testovania môže byť vykonávaná profesionálnym vedúcim testovania, projektovým manažérom, manažérom vývoja alebo manažérom pre zabezpečenie kvality. Vo väčších projektoch alebo organizáciách môže niekoľko testovacích tímov podliehať jednému vedúcemu testovania, koučovi testovania alebo koordinátorovi testovania, pričom každý tím má lídra testovania alebo vedúceho testera.

Typické úlohy vedúceho testovania môžu zahŕňať:

- Vypracujte alebo preskúmajte politiku a stratégiu testovania v organizácii
- Naplánujte aktivity testovania s prihliadnutím na kontext a s pochopením cieľov a rizík testovania. Môže to zahŕňať výber testovacích prístupov, odhady času testovania, prácnosti a nákladov, získanie zdrojov, definovanie úrovní testovania a cyklov testovania a plánovanie manažmentu defektov
- Napíšte a aktualizujte plán(-y) testovania
- Koordinujte plán(-y) testovania s projektovými manažérmi, vlastníkmi produktov a ďalšími osobami
- Zdieľajte perspektívy testovania s inými projektovými aktivitami, ako je integračné plánovanie
- Spustite analýzu, návrh, implementáciu a vykonanie testov, monitorujte pokrok a výsledky testovania a skontrolujte stav výstupných kritérií (alebo definíciu vykonaných)
- Pripravte a doručte reporty o pokroku testovania a súhrnné reporty z testovania na základe zhromaždených informácií
- Prispôsobte plánovanie na základe výsledkov testovania a pokroku (niekedy sú tieto zdokumentované v reportoch o pokroku testovania a/alebo v súhrnných reportoch z iného testovania, ktoré už bolo dokončené v rámci projektu) a realizujte akékoľvek činnosti potrebné na riadenie testovania
- Podporte nastavenie systému manažmentu defektov a adekvátne nastavenie manažmentu testvéru
- Zaveďte vhodné metriky na meranie pokroku testovania a hodnotenie kvality testovania a produktu
- Podporte výber a implementáciu nástrojov na podporu testovacieho procesu vrátane odporúčania rozpočtu na výber nástrojov (a prípadne nákup a/alebo podpora), prideľovania času a prácnosti na pilotné projekty a poskytovania stálej podpory pri používaní nástroja(-ov)
- Rozhodnite o implementácii testovacieho prostredia(-í)
- Podporte a obhajujte testerov, testovací tím a profesiu testera v rámci organizácie
- Rozvíjajte zručnosti a kariéru testerov (napr. prostredníctvom vzdelávacích plánov, hodnotenia výkonov, koučingu atď.)

Spôsob, akým sa vykonáva rola vedúceho testovania, sa líši v závislosti od životného cyklu vývoja softvéru. Napríklad v agilnom vývoji niektoré z vyššie uvedených úloh rieši Agile tím, najmä tie úlohy, ktoré sa týkajú každodenného testovania v tíme, často testerom pracujúcim v tíme. Niektoré úlohy, ktoré sa týkajú viacerých tímov alebo celej organizácie, alebo ktoré sa týkajú personálneho manažmentu, môžu vykonávať vedúci testovania mimo vývojového tímu, ktorí sa niekedy nazývajú kouči testovania. Ďalšie informácie o manažmente procesu testovania pozri Black 2009.

Typické úlohy testera môžu zahŕňať:

- Revidujte a podieľajte sa na plánoch testovania
- Analyzujte, revidujte a posudzujte požiadavky, používateľské príbehy a akceptačné kritériá, špecifikácie a modely testovateľnosti (t. j. základ testovania)
- Identifikujte a dokumentujte podmienky testovania a zachyťte sledovateľnosť medzi testovacími prípadmi, testovacími podmienkami a základom testovania
- Navrhnite, nastavte a overte testovacie prostredie(-a), pričom je potrené koordinovať so správcami systému a sietí
- Navrhnite a implementujte testovacie prípady a testovacie procedúry
- Pripravte a získajte testovacie dáta
- Vytvorte podrobný rozvrh vykonania testov
- Vykonajte testy, vyhodnoťte výsledky a zdokumentujte odchýlky od očakávaných výsledkov
- Použite vhodné nástroje na uľahčenie procesu testovania
- Automatizujte testovanie podľa potreby (eventuálne s podporou vývojára alebo experta na automatizáciu testovania)
- Vyhodnoťte nefunkcionálne charakteristiky, ako napríklad efektívna výkonnosť, spoľahlivosť, použiteľnosť, bezpečnosť, kompatibilita a prenosnosť
- Revidujte testy vytvorené inými osobami

Ľudia, ktorí pracujú na analýze testovania, návrhu testovania, špecifických typoch testovania alebo na automatizácii testovania, môžu byť špecialistami v týchto rolách. V závislosti na rizikách vzťahujúcich sa k produktu a projektu a životného cyklu vývoja softvéru môžu rolu testera prevziať rôzni ľudia na rôznych úrovniach testovania. Napríklad na úrovni testovania komponentov a úrovni testovania integrácie komponentov rolu testera vykonávajú často vývojári. Na akceptačnej úrovni testovania je rola testera často vykonávaná biznis analytikmi, odborníkmi v danej oblasti a používateľmi. Na úrovni systémového testovania a úrovni testovania systémovej integrácie je úloha testera často vykonávaná nezávislým testovacím tímom. Na prevádzkovej akceptačnej úrovni testovania je úloha testera často vykonáva prevádzkarom a/alebo pracovníkmi správy systémov.

5.2 Plánovanie a odhadovanie testovania

5.2.1 Účel a obsah testovacieho plánu

Plán testovania popisuje činnosti testovania pre projekty vývoja a údržby. Plánovanie je ovplyvnené politikou a stratégiou testovania v organizácii, životnými cyklami vývoja a použitými metódami (pozri časť 2.1), rozsahom testovania, cieľmi, rizikami, obmedzeniami, kritickosťou, testovateľnosťou a dostupnosťou zdrojov.

Pri pokroku projektu a plánovania testovania sa postupne získava viac a viac informácií spolu s podrobnosťami, ktoré môžu byť zahrnuté do testovacích plánov. Plánovanie testovania je kontinuálna aktivita a vykonáva sa počas celého životného cyklu produktu. (Uvedomte si, že životný cyklus produktu môže presahovať rozsah projektu tak, aby zahŕňal fázu údržby.) Spätná väzba z aktivít testovania by sa mala použiť na rozpoznanie meniacich sa rizík, aby plánovanie bolo možné upraviť. Plánovanie sa môže zdokumentovať v hlavnom testovacom pláne a v samostatných testovacích plánoch pre úrovne testovania, ako je systémové testovanie a akceptačné testovanie alebo pre samostatné typy testovaní, ako je testovanie použiteľnosti a testovanie výkonnosti. Činnosti plánovania testovania môžu zahŕňať nasledujúce činnosti a niektoré z nich môžu byť zdokumentované v pláne testovania:

- Určenie rozsahu, cieľov a rizík testovania
- Definovanie celkového prístupu k testovaniu
- Integrácia a koordinovanie aktivít testovania do činností životného cyklu softvéru
- Rozhodovania o tom, čo sa má testovať, o ľuďoch a iných zdrojoch potrebných na vykonávanie rôznych činností testovania a o tom, ako sa budú vykonávať činnosti testovania
- Naplánovanie harmonogramu analýzy testovania, navrhovania, implementácie, vykonávania testovania a činností vyhodnocovania buď v konkrétnych termínoch (napr. v sekvenčnom vývoji) alebo v kontexte každej iterácie (napr. v iteratívnom vývoji)
- Výber metrík na monitorovanie a riadenie testovania
- Určenie rozpočtu pre aktivity testovania
- Určenie úrovne podrobnosti a štruktúry testovacej dokumentácie (napr. poskytnutím vzorov alebo príkladov dokumentov)

Obsah plánov testovania sa líši a môže presahovať rámec vyššie uvedených tém. Vzorové testovacie plány nájdete v norme ISO (ISO/IEC/IEEE 29119-3).

5.2.2 Stratégia testovania a prístup k testovaniu

Stratégia testovania poskytuje všeobecný popis procesu testovania, zvyčajne na úrovni produktu alebo organizácie. Medzi bežné typy stratégií testovania patria:

- **Analytický:** Tento typ stratégie testovania je založený na analýze niektorého faktora (napr. požiadavky alebo rizika). Testovanie založené na riziku je príkladom analytického prístupu, pri ktorom sa testy navrhujú a prioritizujú na základe úrovne rizika.
- **Založený na modeloch:** V tomto type stratégie testovania sú testy navrhnuté na základe určitého modelu požadovaného aspektu produktu, ako je funkcia, biznis proces, vnútorná štruktúra alebo nefunkcionálna charakteristika (napr. spoľahlivosť). Príklady takýchto modelov zahŕňajú modely biznis procesov, stavové modely a modely rastu spoľahlivosti.

- **Metodický:** Tento typ stratégie testovania sa opiera o systematické používanie niektorých preddefinovaných súborov testov alebo testovacích podmienok, ako je napríklad taxonómia bežných alebo pravdepodobných typov zlyhaní, zoznam dôležitých kvalitatívnych charakteristík alebo celopodnikové štandardy „look-and-feel“ pre mobilné aplikácie alebo webové stránky.
- **Založený na procesoch** (alebo na štandardoch): Tento typ stratégie testovania zahŕňa analýzu, navrhovanie a implementáciu testov založených na vonkajších pravidlách a štandardoch, ako sú tie, ktoré sú definované štandardami špecifickými pre daný priemysel, procesnou dokumentáciou, dôslednou identifikáciou a používaním základu testovania alebo akýmkoľvek procesom alebo ktorý bol organizácii alebo organizáciou nariadený.
- **Riadený** (alebo konzultatívny): Tento typ stratégie testovania je založený hlavne na radách, usmerneniach alebo pokynoch kľúčových osôb, odborníkov z biznis oblasti alebo technologických expertov, ktorí môžu byť mimo testovacieho tímu alebo mimo samotnej organizácie.
- **Regresno-averzný:** Tento typ stratégie testovania je motivovaný túžbou zamedziť regresii existujúcich schopností. Táto stratégia testovania zahŕňa opätovné použitie existujúceho testvéru (najmä testovacích prípadov a testovacích dát), rozsiahlu automatizáciu regresných testov a štandardné testovacie sady.
- **Reaktívny:** V tomto type stratégie testovania je testovanie reakciou na testovaný komponent alebo systém a udalosti vyskytujúce sa počas vykonávania testu namiesto toho, aby boli vopred plánované (ako v prípade predchádzajúcich stratégií). Testy sú navrhnuté a implementované a môžu byť ihneď vykonané v reakcii na poznatky získané z predchádzajúcich výsledkov testov. Prieskumné testovanie je bežnou technikou používanou v reaktívnych stratégiách.

Vhodná stratégia testovania sa často vytvára kombináciou niekoľkých typov stratégií testovania. Napríklad testovanie založené na riziku (analytická stratégia) môže byť kombinované s prieskumným testovaním (reaktívna stratégia); navzájom sa dopĺňajú a môžu dosiahnuť efektívnejšie testovanie pri spoločnom použití.

Zatiaľ čo stratégia testovania poskytuje všeobecný popis testovacieho procesu, prístup k testovaniu prispôsobí stratégiu testovania pre konkrétny projekt alebo release. Prístup k testovaniu je východiskovým bodom pre výber testovacích techník, úrovní testovania a typov testovania a pre definovanie vstupných a výstupných kritérií (alebo definície stavu pripravený a vykonaný). Prispôsobenie stratégie sa zakladá na rozhodnutiach týkajúcich sa zložitosti a cieľov projektu, druhu vyvíjaného produktu a analýzy rizika produktu. Vybraný prístup závisí od kontextu a môže zvážiť faktory, ako sú riziká, bezpečnosť, dostupné zdroje a zručnosti, technológia, povaha systému (napr. vytvorený na zákazku versus bežne predávaný), ciele testovania a predpisy.

5.2.3 Vstupné a výstupné kritériá (definovanie stavu pripravený a vykonaný)

S cieľom účinne kontrolovať kvalitu softvéru a testovania je vhodné mať kritériá, ktoré určujú, kedy má začať daná aktivita testovania a kedy je aktivita ukončená. Vstupné kritériá (zvyčajne nazývané definícia stavu pripravený v agilnom vývoji) definujú predpoklady na vykonanie danej aktivity testovania. Ak nie sú splnené vstupné kritériá, je pravdepodobné, že aktivita bude komplikovanejšia, časovo náročnejšia, drahšia a rizikovejšia. Výstupné kritériá (zvyčajne nazývané definícia stavu vykonaný v agilnom vývoji) určujú, aké podmienky je potrebné dosiahnuť, aby sa deklarovala úroveň testovania alebo súbor testov ako ukončený. Vstupné a výstupné kritéria by sa mali definovať pre každú úroveň testovania a typ testu a budú sa líšiť na základe cieľov testovania.

Typické vstupné kritériá zahŕňajú:

- Dostupnosť testovateľných požiadaviek, používateľských príbehov a/alebo modelov (napr. pri aplikovaní stratégie založenej na modeloch)

- Dostupnosť testovacích položiek, ktoré splnili výstupné kritériá pre akékoľvek predchádzajúce úrovne testov
- Dostupnosť testovacieho prostredia
- Dostupnosť potrebných testovacích nástrojov
- Dostupnosť testovacích dát a ďalších potrebných zdrojov

Typické výstupné kritériá zahŕňajú:

- Plánované testy boli vykonané
- Dosiahla sa definovaná úroveň pokrytia (napr. požiadaviek, používateľských príbehov, akceptačných kritérií, rizík, kódu)
- Počet nevyriešených defektov, ktorý neprekročil dohodnutú hranicu
- Počet odhadovaných zostávajúcich defektov je dostatočne nízky
- Hodnotené úrovne spoľahlivosti, výkonnosti, použiteľnosti, bezpečnosti a iných relevantných kvalitatívnych charakteristík sú dostatočné

Aj bez splnenia výstupných kritérií je bežné, že aktivity testovania sú ukončené kvôli vyčerpaniu rozpočtu, uplynutiu naplánovaného času a/alebo tlaku na uvedenie produktu na trh. Ukončenie testovania môže byť prijateľné za takýchto okolností, ak kľúčové osoby projektu a biznis vlastníci pochopili a prijali riziko vyplývajúce z uvedenia do prevádzky bez ďalšieho testovania.

5.2.4 Rozvrh vykonania testovania

Akonáhle sa vytvoria rôzne testovacie prípady a testovacie procedúry (s niektorými testovacími procedúrami potenciálne automatizovanými) a zostavia sa do testovacích sád, tieto testovacie sady môžu byť usporiadané do rozvrhu vykonania testovania, ktorý definuje poradie, v ktorom majú byť vykonané. Rozvrh vykonania testovania by mal brať do úvahy také faktory, ako sú prioritizácia, vzájomné závislosti, konfirmačné testy, regresné testy a najefektívnejšia postupnosť pri vykonávaní testov.

V ideálnom prípade by sa testovacie prípady mali zoradiť na základe úrovne ich priority, zvyčajne najprv vykonaním testovacích prípadov s najvyššou prioritou. Tento postup však nemusí fungovať, ak majú testovacie prípady alebo testované funkcie vzájomné závislosti. Ak je testovací prípad s vyššou prioritou závislý od testovacieho prípadu s nižšou prioritou, najskôr sa musí vykonať testovací prípad s nižšou prioritou. Podobne, ak existujú závislé vzťahy medzi testovacími prípadmi, musia byť vhodne zoradené bez ohľadu na ich relatívne priority. Priority konfirmačných a regresných testov musia byť tiež určené, a to na základe dôležitosti rýchlej spätnej väzby o zmenách, pričom sa tu môžu opäť vyskytnúť vzájomné závislosti.

V niektorých prípadoch sú možné rôzne sekvencie testov s rôznymi úrovňami účinnosti spojenými s týmito sekvenciami. V takýchto prípadoch sa musia urobiť kompromisy medzi efektívnosťou vykonávania testu a dodržaním stanovenia priorít.

5.2.5 Faktory ovplyvňujúce prácnosť testovania

Odhadovanie prácnosti testovania zahŕňa predpovedanie množstva práce súvisiacej s testovaním, ktoré bude potrebné na splnenie cieľov testovania konkrétneho projektu, release alebo iterácie. Faktory ovplyvňujúce prácnosť testovania môžu zahŕňať charakteristiky produktu, charakteristiky vývojového procesu, charakteristiky ľudí a výsledky testov, ako je uvedené nižšie.

Charakteristiky produktu

- Riziká spojené s produktom

- Kvalita základu testovania
- Veľkosť produktu
- Zložitosť oblasti používania produktu
- Požiadavky na kvalitatívne charakteristiky (napr. bezpečnosť, spoľahlivosť)
- Požadovaná úroveň detailu pre dokumentáciu testu
- Požiadavky na dodržiavanie právnych predpisov a nariadení

Charakteristiky procesu vývoja

- Stabilita a zrelosť organizácie
- Použitý model vývoja
- Prístup k testovaniu
- Použité nástroje
- Proces testovania
- Časový tlak

Charakteristiky ľudí

- Zručnosti a skúsenosti zapojených ľudí, najmä s podobnými projektmi a produktmi (napr. znalosť oblasti používania)
- Súdržnosť a vedenie tímu

Výsledky testov

- Počet a závažnosť zistených defektov
- Požadovaný rozsah prepracovania

5.2.6 Techniky odhadu prácnosti testovania

Existuje celý rad techník odhadu prácnosti, ktoré sa používajú na stanovenie prácnosti požadovanej pre adekvátne testovanie. Dve z najbežnejšie používaných techník sú:

- Technika založená na metrikách: odhady prácnosti testovania na základe metrík predchádzajúcich podobných projektov alebo na základe typických hodnôt
- Technika založená na expertoch: odhady prácnosti testovania na základe skúseností vlastníkov testovacích úloh alebo expertov.

Napríklad v agilnom vývoji sú burndown grafy príkladmi prístupu založeného na metrikách, pretože sa prácnosť zachytáva a reportuje a následne sa používa pri meraní rýchlosti tímu (velocity) na určenie množstva práce, ktorú tím môže vykonať v ďalšej iterácii. Na druhej strane je príkladom prístupu založeného na expertoch (plánovací poker), keďže členovia tímu odhadujú prácnosť na vývoj funkcie podľa svojich skúsenosti (ISTQB-AT Foundation Level Agile Tester Extension).

Pri sekvenčných projektoch sú modely na odstránenie defektov príkladom prístupu založeného na metrikách, kde sa zachytáva a reportuje množstvo defektov a čas potrebný na ich odstránenie, čo potom poskytuje základ pre odhad budúcich projektov podobnej povahy; zatiaľ čo technika odhadovania Wideband Delphi je príkladom prístupu založeného na expertoch, v ktorom skupiny expertov poskytujú odhady na základe svojich skúseností (ISTQB-ATM Advanced Level Test Manager Syllabus).

5.3 Monitorovanie a riadenie testovania

Účelom monitorovania testovania je zhromažďovanie informácií a poskytovanie spätnej väzby a viditeľnosti o aktivitách testovania. Informácie, ktoré sa majú monitorovať, sa môžu zbierať manuálne alebo automaticky a mali by sa použiť na posúdenie pokroku testovania a na meranie toho, či sú splnené výstupné kritéria testu alebo úlohy testovania súvisiace s definíciou stavu vykonané v rámci agilného projektu, ako sú napríklad splnenie cieľov pre pokrytie produktových rizík, požiadaviek alebo akceptačných kritérií.

Riadenie testovania opisuje akékoľvek usmernenia alebo nápravné opatrenia, ktoré sa realizovali v dôsledku zhromaždených a (prípadne) hlásených informácií a metrík. Činnosti môžu zahŕňať akúkoľvek aktivitu testovania a môžu ovplyvniť akúkoľvek inú aktivitu životného cyklu softvéru.

Príklady činností riadenia testovania zahŕňajú:

- Opätovné stanovenie priorít testov, keď sa objaví identifikované riziko (napr. softvér je dodaný neskoro)
- Zmena rozvrhu testov v dôsledku dostupnosti alebo nedostupnosti testovacieho prostredia alebo iných zdrojov
- Opätovné posúdenie, či testovacia položka spĺňa vstupné alebo výstupné kritériá v dôsledku prepracovania

5.3.1 Metriky použité pri testovaní

Metriky je možné zhromažďovať počas a na konci aktivít testovania za účelom posúdenia:

- Pokroku v plánovanom rozvrhu a rozpočte
- Aktuálnej kvality testovaného objektu
- Primeranosti testovacieho prístupu
- Účinnosti činností testovania vo vzťahu k cieľom

Bežné metriky testovania zahŕňajú:

- Percentuálny podiel plánovanej práce vykonanej pri príprave testovacích prípadov (alebo percentuálny podiel plánovaných testovacích prípadov, ktoré boli implementované)
- Percentuálny podiel plánovanej práce vykonanej v príprave testovacieho prostredia
- Vykonanie testovacích prípadov (napr. počet testovacích prípadov vykonaných/ nevykonaných, testovacie prípady úspešné/neúspešné a/alebo testovacie podmienky úspešné/ neúspešné)
- Informácie o defektoch (napr. hustota defektov, nájdené a opravené defekty, výsledky zlyhaní a výsledky konfirmačných testov)
- Pokrytie požiadaviek, používateľských príbehov, akceptačných kritérií, rizík alebo kódu testami
- Dokončenosť úlohy, alokácia a využitie zdrojov a prácnosti
- Náklady na testovanie vrátane porovnania dodatočných nákladov voči prínosu nájdenia ďalšieho defektu alebo porovnania dodatočných nákladov voči prínosu vykonania ďalšieho testu

5.3.2 Účel, obsah a cieľoví adresáti reportov z testovania

Účelom reportingu z testovania je zhrnúť a oznámiť informácie o aktivitách testovania počas a na konci aktivity testovania (napr. na jednej testovacej úrovni). Report z testovania pripravený počas aktivity testovania, môže byť označený ako report o pokroku testovania, zatiaľ čo report z testovania pripravený na konci činnosti testovania môže byť označený ako súhrnný report z testovania.

Počas monitorovania a riadenia testovania vytvára vedúci testovania pravidelné reporty o pokroku testovania pre kľúčové osoby. Okrem obsahu bežného pre reporty o pokroku testovania a súhrnné reporty z testovania môžu typické reporty o pokroku testovania obsahovať aj:

- Stav činností testovania a pokrok oproti plánu testovania
- Faktory, ktoré bránia prokroku
- Testovanie plánované pre nasledujúce reportované obdobie
- Kvalita testovaného objektu

Po dosiahnutí výstupných kritérií, vedúci testovania vydá súhrnný report z testovania. Tento report poskytuje súhrn informácií o vykonanom testovaní na základe najnovšieho reportu o pokroku testovania a všetky ďalšie relevantné informácie.

Typické reporty o pokroku testovania a súhrnné reporty z testovania môžu obsahovať:

- Zhrnutie vykonaného testovania
- Informácie o tom, čo sa stalo počas testovacieho obdobia
- Odchýlky od plánu, vrátane odchýlok v rozvrhu, trvaní alebo prácnosť aktivít testovania
- Stav testovania a kvalita produktu s ohľadom na výstupné kritériá alebo definíciu stavu vykonané
- Faktory, ktoré zablokovali alebo naďalej blokujú pokrok
- Metriky defektov, testovacích prípadov, pokrytia testov, pokroku aktivít a spotreby zdrojov (napr. ako je opísané v 5.3.1)
- Reziduálne riziká (pozri časť 5.5)
- Vytvorené opakovane použiteľné testovacie produkty

Obsah reportu z testovania sa bude líšiť v závislosti od projektu, organizačných požiadaviek a životného cyklu vývoja softvéru. Napríklad komplexný projekt s mnohými kľúčovými osobami alebo regulovaný projekt môžu vyžadovať podrobnejší a dôkladnejší reporting než rýchla aktualizácia softvéru. Ďalším príkladom je, že v agilnom vývoji môže byť report o pokroku testovania zahrnutý v úlohách riadenia (boardov), výkazov o defektoch a burndown grafov, ktoré môžu byť diskutované počas denného stretnutia (pozri ISTQB-AT Foundation Level Agile Tester Extension).

Okrem prispôsobenia reportov z testovania na základe kontextu projektu by mali byť reporty z testovania prispôsobené podľa ich cieľových adresátov. Typ a množstvo informácií, ktoré by sa mali zahrnúť pre technickú cieľovú skupinu alebo testovací tím, sa môže líšiť od toho, čo by malo byť zahrnuté do súhrnného reportu pre manažment. V prvom prípade môžu byť dôležité podrobné informácie o typoch defektov a trendoch. V druhom prípade môže byť vhodnejší všeobecnejší report (napr. zhrnutie stavu defektov podľa priorít, rozpočtu, rozvrhu a testovacích podmienok úspešných / neúspešných / netestovaných).

Norma ISO (ISO / IEC / IEEE 29119-3) sa vzťahuje na dva typy reportov z testovania – reporty o pokroku testovania a reporty o dokončení testovania (nazývané súhrnné reporty v tejto osnove) a obsahuje štruktúry a príklady pre každý jeden uvedený typ.

5.4 Konfiguračný manažment

Účelom konfiguračného manažmentu je vytvorenie a udržovanie integrity komponentu alebo systému, testvéru a ich vzťahov počas projektového a produktového životného cyklu.

Pre správnu podporu testovania môže konfiguračný manažment zaistiť nasledovné:

- Všetky testované položky sú jednoznačne identifikované, ich verzie sú riadené, zmeny sledované a položky sú vzájomne prepojené
- Všetky položky testvéru sú jednoznačne identifikované, ich verzie sú riadené, zmeny sledované, vzájomne prepojené a prepojenie s testovanými položkami existujúce tak, že je možné udržať sledovateľnosť počas celého procesu testovania
- Na všetky identifikované dokumenty a položky softvéru sa dá jednoznačne odkázať v testovacej dokumentácii

Procedúry a infraštruktúra (nástroje) konfiguračného manažmentu by mali byť zvolené, dokumentované a implementované počas plánovania testov

5.5 Riziká a testovanie

5.5.1 Definícia rizika

Riziko zahŕňa možnosť udalosti v budúcnosti, ktorá má negatívne dôsledky. Úroveň rizika je určená pravdepodobnosťou udalosti a vplyvom (veľkosťou škody) tejto udalosti.

5.5.2 Produktové a projektové riziká

Produktové riziko zahŕňa možnosť, že pracovný produkt (napr. špecifikácia, komponent, systém alebo test) nemusí spĺňať legitímne potreby jeho používateľov a/alebo kľúčových osôb. Produktové riziká spojené so špecifickými kvalitatívnymi charakteristikami produktu (napr. funkcionálna vhodnosť, spoľahlivosť, účinnosť, použiteľnosť, bezpečnosť, kompatibilita, udržiavateľnosť a prenosnosť), sa tiež nazývajú riziká kvality. Medzi príklady produktových rizík patria:

- Softvér nemusí vykonávať zamýšľané funkcie podľa špecifikácie
- Softvér nemusí vykonávať zamýšľané funkcie podľa potrieb používateľov, zákazníkov a/alebo kľúčových osôb
- Architektúra systému nemusí adekvátne podporovať niektoré nefunkcionálne požiadavky
- Za určitých okolností môže byť nesprávne vykonaný určitý výpočet
- Riadenia slučky môže byť nesprávne naprogramované
- Rýchlosť odozvy môžu byť nedostatočné pre vysoko výkonný systém spracovania transakcií
- Spätná väzba na základe skúseností používateľov (User Experience – UX) nemusí spĺňať očakávania spojené s produktom.

Projektové riziká zahŕňajú situácie, ktoré by v prípade výskytu mohli mať negatívny vplyv na schopnosť projektu dosiahnuť požadované ciele. Medzi príklady projektových rizík patria:

- Projektové problémy:
  - Môžu sa vyskytnúť oneskorenia pri dodaní, plnení úloh alebo splnení výstupných kritérií alebo definície vykonanej práce
  - Nepresné odhady, prerozdelenie finančných prostriedkov na projekty s vyššou prioritou alebo celkové zníženie nákladov v celej organizácii môžu viesť k nedostatočnému financovaniu
  - Neskoré zmeny môžu viesť k významným prepracovaniam
- Organizačné problémy:
  - Schopnosti, školenia a personál nemusia byť postačujúce
  - Problémy s personálom môžu spôsobiť konflikty a problémy
  - Používatelia, biznis personál alebo odborníci v oblasti užívania nemusia byť k dispozícii protichodným biznis prioritám
- Politické problémy:
  - Testeri nemusia primerane komunikovať svoje potreby a/alebo výsledky testovania
  - Vývojári a/alebo testeri nemusia dostatočne nadviazať na informácie získané pri testovaní a revíziách (napr. nebudú zlepšovať postupy vývoja a testovania)
  - Môže existovať nesprávny postoj k testovaniu alebo očakávania od testovania (napr. neoceňovanie hodnoty zistených defektov počas testovania)
- Technické problémy:
  - Požiadavky nemusia byť dostatočne definované
  - Požiadavky nemusia byť splnené vzhľadom na existujúce obmedzenia
  - Testovacie prostredie nemusí byť pripravené včas
  - Konverzia dát, plánovanie migrácie a podpora nástrojov môže byť oneskorená
  - Nedostatky v procese vývoja môžu mať vplyv na konzistenciu alebo kvalitu projektových pracovných produktov, ako je návrh, kód, konfigurácia, testovacie dáta a testovacie prípady
  - Slabý manažment defektov a podobné problémy môžu mať za následok nahromadenie defektov a iné technické dlhy
- Dodávateľské problémy:
  - Tretia strana môže zlyhať pri dodaní potrebného produktu alebo služby alebo môže dôjsť k bankrotu
  - Zmluvné problémy môžu spôsobiť problémy pre projekt

Projektové riziká môžu ovplyvniť vývojové aktivity a aktivity testovania. V niektorých prípadoch sú projektoví manažéri zodpovední za riešenie všetkých projektových rizík, ale nie je nezvyčajné, aby vedúci testovania boli zodpovední za projektové riziká súvisiace s testovaním.

5.5.3 Testovanie založené na rizikách a kvalita produktu

Riziko sa používa na zameranie prácnosti vyžadovaného počas testovania. Používa sa pri rozhodovaní o tom, kde a kedy začať testovanie a na určenie oblastí, ktoré potrebujú väčšiu pozornosť. Testovanie sa používa na zníženie pravdepodobnosti vzniku nežiaducej udalosti alebo na zníženie vplyvu nežiaducej udalosti. Testovanie sa používa ako aktivita na zmiernenie rizika, poskytuje spätnú väzbu o identifikovaných rizikách a poskytuje spätnú väzbu o reziduálnych (nevyriešených) rizikách.

Prístup k testovaniu založený na riziku poskytuje proaktívne príležitosti na zníženie úrovne produktového rizika. Zahŕňa analýzu produktového rizika, ktorá pozostáva z identifikácie produktových rizík a posúdenia pravdepodobnosti a vplyvu každého rizika. Výsledné informácie o produktovom riziku sa používajú na usmerňovanie plánovania testovania, špecifikácie, prípravy a vykonávania testovacích prípadov a monitorovanie a riadenie testovania. Včasná analýza produktových rizík prispieva k úspechu projektu.

V rámci prístupu založeného na riziku sa používajú výsledky analýzy produktového rizika na:

- Určenie testovacích techník, ktoré sa majú použiť
- Určenie konkrétnych úrovní a typov testov, ktoré sa majú vykonať (napr. testovanie bezpečnosti, testovanie prístupnosti)
- Určenie rozsahu testovania, ktoré sa má vykonať
- Prioritizovanie testovania s cieľom nájsť kritické defekty čo najskôr
- Zistenie, či by sa na zníženie rizika mohli použiť aj iné aktivity ako testovanie (napr. poskytnutie školení pre neskúsených návrhárov)

Testovanie založené na riziku vychádza zo spoločných vedomostí a znalostí kľúčových osôb projektu vykonať analýzu produktového rizika. Aby sa zabezpečila minimalizácia pravdepodobnosti zlyhania produktu, aktivity manažmentu rizík poskytujú zavedený prístup pre:

- Analýzu (a pravidelné prehodnotenie) toho, čo sa môže pokaziť (riziká)
- Určenie rizík, s ktorými sa treba zaoberať
- Zavedenie opatrení na zmiernenie týchto rizík
- Vytvorenie pohotovostných plánov na riešenie rizík, ak tieto nastanú

Navyše, testovanie môže identifikovať aj nové riziká, pomôcť určiť, ktoré riziká by sa mali zmierniť, a znížiť neistotu vzhľadom na riziká.

5.6 Manažment defektov

Keďže jedným z cieľov testovania je nájsť defekty, je potrebné defekty zistené počas testovania zaznamenať. Spôsob, akým sú defekty zaznamenávané, sa môže líšiť v závislosti od kontextu testovaného komponentu alebo systému, úrovne testovania a modelu životného cyklu vývoja softvéru. Zistené defekty by mali byť preskúmané a mali by sa sledovať od ich identifikácie a klasifikácie až po ich vyriešenie (napr. oprava defektov a úspešné konfirmačné testovanie riešenia, odloženie na následné release, akceptácia ako trvalé obmedzenie produktu atď.). Aby boli všetky defekty vyriešené, organizácia by mala vytvoriť proces manažmentu defektov, ktorý zahŕňa pracovný tok (workflow) a pravidlá pre klasifikáciu. Tento proces musí byť odsúhlasený všetkými účastníkmi v oblasti manažmentu defektov vrátane návrhárov, vývojárov, testerov a vlastníkov produktov. V niektorých organizáciách môže byť zaznamenávanie a sledovanie defektov veľmi neformálne.

Počas procesu manažmentu defektov sa niektoré reporty môžu ukázať ako informácie o falošných pozitívach, nie ako skutočné zlyhania spôsobené defektmi. Napríklad, test môže zlyhať, ak sa preruší sieťové pripojenie alebo pripojenie sa ukončí vypršaním času. Toto správanie nevyplýva z defektu testovaného objektu, ale je to anomália, ktorá sa musí vyšetriť. Testeri by sa mali pokúsiť minimalizovať počet falošných pozitív hlásených ako defekty.

Defekty môžu byť hlásené počas programovania, statickej analýzy, revízií, dynamického testovania alebo používania softvérového produktu. Defekty môžu byť hlásené pre problémy v kóde alebo bežiacich (spustených) systémoch alebo v akomkoľvek type dokumentácie vrátane požiadaviek, používateľských príbehov a akceptačných kritérií, vývojových dokumentov, testovacích dokumentov, používateľských alebo inštalačných príručiek. V záujme efektívneho a účinného procesu manažmentu defektov môžu organizácie definovať štandardy pre atribúty, klasifikáciu a pracovný tok (workflow) defektov.

Typické reporty o defekte majú nasledovné ciele:

- Poskytnúť vývojárom a ďalším stranám informácie o akýchkoľvek nepriaznivých udalostiach, ktoré sa vyskytli, aby mohli identifikovať konkrétne následky, izolovať problém s minimálnym testom reprodukovania a opraviť prípadný defekt/defekty podľa potreby alebo inak vyriešiť problém
- Poskytnúť vedúcim testovania prostriedok na sledovanie kvality pracovného produktu a jeho vplyvu na testovanie (napr. ak je hlásených veľa defektov, testeri budú venovať veľa času reportingu namiesto testovania a bude potrebné viac konfirmačného testovania)
- Poskytnúť nápady pre zlepšovanie vývojového a testovacieho procesu

Report o defekte vytvorený počas dynamického testovania zvyčajne zahŕňa:

- Identifikátor
- Názov a krátke zhrnutie reportovaného defektu
- Dátum reportu o defekte, organizácia a autor, ktorý report podávajú
- Identifikáciu testovanej položky (testovaná konfiguračná položka) a prostredia
- Fázu(-y) životného cyklu vývoja, v ktorých bol defekt pozorovaný
- Popis defektu umožňujúci reprodukovanie a vyriešenie, vrátane protokolov, screenshotov databázových dumpov alebo záznamov (ak sa zistí počas vykonávania testu)
- Očakávané a skutočné výsledky
- Rozsah alebo stupeň vplyvu (závažnosti) defektu na záujmy kľúčových osôb,
- Naliehavosť/priorita opravy

- Stav reportu o defekte (napr. otvorený, odložený, duplicitný, čakajúci na opravenie, čakanie na konfirmačné testovanie, opätovne otvorený, zatvorený)
- Závery, odporúčania a schválenia
- Globálne problémy, napríklad iné oblasti, ktoré môžu byť ovplyvnené zmenou vyplývajúcou z defektu
- Históriu zmien, ako je napríklad postupnosť činností prijatých členmi projektového tímu s ohľadom na defekt s cieľom izolovať, opraviť a potvrdiť jeho opravu
- Referencie vrátane testovacieho prípadu, ktorý odhalil problém

Niektoré z týchto detailov môžu byť automaticky zahrnuté a/alebo manažované pri používaní nástrojov manažmentu defektov, napr. automatické priradenie identifikátora, priradenie a aktualizácia stavu reportu o defekte počas pracovného toku (workflow) atď. Defekty identifikované počas statického testovania sa zvyčajne zdokumentujú iným spôsobom, napr. v poznámkach revízneho stretnutia.

Príklad obsahu reportu o defektoch možno nájsť v norme ISO (ISO / IEC / IEEE 29119-3) (kde sa reportom o defektoch hovorí reporty o incidentoch).

6 Podporné nástroje pre testovanie — 40 minút

Kľúčové slová

testovanie riadené dátami, testovanie riadené kľúčovými slovami, nástroj na testovanie výkonu, automatizácia testovania, nástroj na vykonanie testu, nástroj na správu testov

Študijné ciele pre testovacie nástroje

6.1 Špecifiká testovacích nástrojov

FL-6.1.1 (K2) Klasifikovať testovacie nástroje podľa ich účelu a podporovaných testovacích aktivít

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

FL-6.1.3 (K1) Pripomenúť si špecifiká pre vykonávanie testov a nástrojov pre manažment testovania

6.2 Efektívne používanie nástrojov

FL-6.2.1 (K1) Identifikovať hlavné princípy výberu nástroja

FL-6.2.2 (K1) Pripomenúť si ciele používania pilotných projektov pre zavádzanie nástrojov

FL-6.2.3 (K1) Identifikovať faktory úspechu pre vyhodnotenie, implementáciu, nasadenie a nepretržitú podporu testovacích nástrojov v organizácii

6.1 Špecifiká testovacích nástrojov

Testovacie nástroje môžu byť použité pre podporu jednej alebo viacerých testovacích aktivít. Tieto nástroje zahŕňajú:

- Nástroje, ktoré sú priamo používané v testovaní ako napríklad nástroje pre vykonanie testov alebo nástroje pre prípravu testovacích dát
- Nástroje, ktoré pomáhajú v riadení požiadaviek, testovacích prípadov, postupov testovania, automatických testovacích skriptov, výsledkov testov, dát, defektov a pre reportovanie a monitorovanie vykonania testov
- Nástroje, používané pri preskúmaní a hodnotení
- Akýkoľvek nástroj, ktorý napomáha pri testovaní (tabuľka je v tomto zmysle tiež testovacím nástrojom)

6.1.1 Klasifikácia testovacích nástrojov

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

- Zvýšenie efektívnosti testovacích aktivít automatizáciou opakovaných úloh alebo úloh, ktoré vyžadujú výrazné zdroje pri manuálnom vykonávaní (napr. vykonávanie testov, regresné testovanie)
- Zvýšenie efektívnosti testovacích aktivít podporovaním manuálnych testovacích aktivít počas celého procesu testovania (pozri časť 1.4)
- Zvýšenie kvality testovacích aktivít, konzistentným testovaním a vyššou úrovňou reprodukovateľnosti defektov
- Automatizovanie aktivít, ktoré nemôžu byť vykonávané manuálne (napr. testovanie výkonu veľkého rozsahu)
- Zvýšenie spoľahlivosti testovania (napr. automatizáciou porovnávania veľkého množstva dát alebo simulovaním správania)

Nástroje môžu byť klasifikované na základe viacerých kritérií ako sú účel, cena, model licencie (napr. komerčný alebo voľne dostupný (open-source, pozn. prekladu) alebo použitá technológia. Nástroje sú v tejto osnove roztriedené podľa testovacích aktivít, ktoré podporujú.

Niektoré nástroje podporujú práve jednu aktivitu, resp. hlavnú aktivitu. Iné môžu podporovať viacero aktivít. Takéto nástroje sú zaradené pod aktivitu, s ktorou sú najviac spojené. Nástroje od jedného poskytovateľa, obzvlášť tie, ktoré boli navrhnuté, aby pracovali spolu, môžu byť dodané ako integrovaná sada programov.

Niektoré typy testovacích nástrojov môžu byť narúšajúce takým spôsobom, že môžu ovplyvniť skutočný výsledok testu. Napríklad nameraný čas môže byť rôzny z dôvodu, že sú nástrojom pre testovanie výkonu navyše vykonávané inštrukcie, alebo pokrytie kódu môže byť ovplyvnené nástrojom na meranie pokrytia. Následok narúšajúcich nástrojov nazývame vplyv skúšania.

Niektoré nástroje ponúkajú podporu vhodnú skôr pre vývojárov (napr. nástroje, ktoré sú použité počas testovania komponentov alebo integračného testovania komponentov). Takéto nástroje sú v rozdelení uvedenom ďalej označené písmenom “V”.

Podporné nástroje pre manažment testovania a testov

Nástroje pre manažment sa môžu použiť na všetky testovacie aktivity počas celého životného cyklu softvéru. Príklady nástrojov, ktoré podporujú manažment testovania a testov zahŕňajú:

- Nástroje pre manažment testovania a nástroje pre manažment životného cyklu aplikácie (ALM)
- Nástroje pre manažment požiadaviek (napr. sledovateľnosť testovaných objektov)

- Nástroje pre manažment defektov
- Nástroje pre konfiguračný manažment
- Nástroje kontinuálnej integrácie (V)

Podporné nástroje pre statické testovanie

Nástroje statického testovania sú spojené s aktivitami a prínosmi opísanými v kapitole 3. Príklady takýchto nástrojov zahŕňajú:

- Nástroje pre revidovanie
- Nástroje pre statickú analýzu (V)

Podporné nástroje pre návrh a implementáciu testov

Nástroje navrhovania testov pomáhajú pri vytváraní udržiavateľných pracovných produktov pri navrhovaní a realizácii testov vrátane testovacích prípadov, testovacích postupov a testovacích dát. Príklady takýchto nástrojov zahŕňajú:

- Nástroje navrhovania testov
- Nástroje testovania založené na modeloch
- Nástroje prípravy testovacích dát
- Nástroje na vývoj riadený akceptačným testovaním alebo správaním (V)
- Nástroje na vývoj riadený testovaním (V)

V niektorých prípadoch môžu nástroje, ktoré podporujú návrh a implementáciu testov, tiež podporovať vykonávanie testov a zaznamenávanie alebo poskytovať svoje výstupy priamo iným nástrojom, ktoré podporujú vykonávanie a zaznamenávanie testov.

Podporné nástroje pre vykonávanie testov a záznamy

Existuje mnoho nástrojov na podporu a zlepšenie činností vykonania testov a zaznamenávania. Príklady týchto nástrojov zahŕňajú:

- Nástroje vykonávania testov (napr. na vykonanie regresných testov)
- Nástroje pokrytia (napr. pokrytie požiadaviek, pokrytie kódu (V))
- Testovacie postroje (V)
- Rámcové nástroje jednotkových testov (V)

Podporné nástroje pre meranie výkonu a dynamickú analýzu

Nástroje na meranie výkonnosti a dynamickú analýzu sú nevyhnutné pri podpore činností v oblasti testovania výkonu a zaťaženia, pretože tieto činnosti sa nedajú manuálne efektívne vykonávať. Príklady týchto nástrojov zahŕňajú:

- Nástroje testovania výkonu
- Monitorovacie nástroje
- Nástroje dynamickej analýzy (V)

Podporné nástroje pre špecializované potreby testovania

Okrem nástrojov podporujúcich testovací proces vo všeobecnosti, existuje mnoho ďalších nástrojov podporujúcich špecifickejšie problémy testovania. Príklady týchto nástrojov zahŕňajú nástroje, ktoré sa zameriavajú na:

- Posúdenie kvality dát
- Konverziu a migráciu dát
- Testovanie použiteľnosti
- Testovanie prístupnosti
- Testovanie lokalizácie
- Testovanie bezpečnosti
- Testovanie prenosnosti (napr. testovanie softvéru na viacerých podporovaných platformách)

6.1.2 Výhody a riziká automatizácie testovania

Samotné nadobudnutie nástroja ešte nezaručuje úspešný výsledok. Každý nový typ nástroja v organizácii môže vyžadovať dodatočnú prácnosť na dosiahnutie skutočných a trvalých výhod. S použitím nástroja sa spájajú možné výhody a príležitosti, ale prináša aj riziká. Tento fakt čiastočne zrejmý pri nástrojoch pre vykonania testov (tiež označované ako automatizácia testovania).

Možné výhody používania nástrojov pre podporu vykonávania testovania zahŕňajú:

- Opakované manuálne činnosti sú redukované (napr. spustenie regresných testov, opätovné zadávanie rovnakých testovacích dát, úlohy nastavenia/rozloženia prostredia, kontrola štandardov kódovania), čo prináša časovú úsporu.
- Väčšia konzistencia a opakovateľnosť (napr. testovacie dáta sú vytvorené koherentným spôsobom, testy vykonávané nástrojom v tom istom poradí s tou istou frekvenciou a konzistentne odvodené testovacie prípady z požiadaviek).
- Objektívne hodnotenie (napr. statické opatrenia, pokrytie).
- Jednoduchší prístup k informáciám o testovaní (štatistiky a grafy pokroku testovania, pomery resp. častosť defektov alebo miera výkonu).

Možné riziká používania nástrojov podpory testovania zahŕňajú:

- Nerealistické očakávania od nástroja (vrátane funkcionality a jednoduchosti používania)
- Podcenenie času, nákladov a prácnosti pri prvotnom zavádzaní nástroja (vrátane školení a externej podpory)
- Podcenenie času a prácnosti, ktoré sú potrebné na dosiahnutie významných a trvalých výhod z nástroja (vrátane potreby zmeny procesov testovania a kontinuálneho zlepšovania spôsobov použitia nástroja)
- Podcenenie prácnosti potrebného na údržbu testovacích produktov vytvorených nástrojom
- Prehnané spoliehanie sa na nástroj (nástroj ako náhrada návrhu testu alebo vykonania testu, alebo použitie automatizovaného testovania v prípade, kde by manuálne testovanie bolo vhodnejšie)
- Zanedbanie riadenia verzií produktov testovania v nástroji

- Zanedbanie vzťahov a problémov medzi dôležitými nástrojmi, ako sú nástroje pre manažment požiadaviek, nástroje pre konfiguračný manažment, nástroje pre manažment incidentov, nástroje pre sledovanie defektov a nástroje od viacerých dodávateľov.
- Riziko ukončenia činnosti dodávateľa nástroja, že skrachuje, ukončenia podpory nástroja, alebo predaja nástroja inému dodávateľovi
- Nízka odozva dodávateľa čo sa týka podpory, nových verzií a opráv defektov
- Riziko ukončenia projektu nástroja s otvoreným zdrojovým kódom
- Neschopnosť nástroja podporovať novú platformu alebo technológiu
- Vlastníctvo nástroja nemusí byť jasné (napr. pre mentoring, aktualizácie atď.)

6.1.3 Osobité úvahy pre nástroje realizácie testov a manažment testovania

Existuje množstvo faktorov, ktoré je potrebné zvážiť pri výbere a integrácii nástrojov na vykonávanie testov a nástrojov pre manažment testovania tak, aby sa zabezpečila hladká a úspešná implementácia do procesov danej spoločnosti

Nástroje vykonávania testov

Nástroje vykonávania testov vykonávajú testované objekty s použitím automatizovaných testovacích skriptov. Tento typ nástroja často vyžaduje veľké úsilie pre dosiahnutie podstatného benefitu.

Zaznamenávanie testov nahrávaním činnosti manuálneho testera vyzerá atraktívne, ale tento prístup prestáva byť efektívny pri veľkom počte automatizovaných testovacích skriptov. Súčasťou každého skriptu sú špecifické činnosti a dáta. To predstavuje priamy výstup zaznamenaného testu. Takýto typ skriptu môže byť nestabilný v prípade výskytu neočakávaných udalostí. Najnovšia generácia nástrojov využíva technológiu „inteligentného“ snímania obrazu a tým robí túto skupinu nástrojov stále použiteľnou. Generované skripty však vyžadujú neustálu údržbu, pretože používateľské rozhranie systému sa v priebehu času vyvíja.

Prístup testovania riadeného dátami oddeľuje testovacie vstupy a očakávané výsledky, zvyčajne do tabuľkového procesora. Tento prístup využíva všeobecný testovací skript, ktorý dokáže načítať a vykonať ten istý test s rôznymi vstupnými dátami. Tester nemusí ovládať skriptovací jazyk a postačuje, ak vytvára dáta pre tieto preddefinované skripty.

V prípade prístupu testovania riadeného kľúčovými slovami všeobecný skript pracuje s kľúčovými slovami, ktoré popisujú činnosti, ktoré majú byť vykonané (taktiež nazývané „akčné slová“). Tieto kľúčové slová následne vyvolajú činnosti, ktoré relevantné testovacie dáta spracujú. Testeri (aj keď nie sú oboznámení so skriptovacím jazykom) môžu navrhovať testy použitím kľúčových slov a súvisiacich dát, ktoré môžu byť prispôsobené na mieru testovanej aplikácii. Ďalšie podrobnosti a príklady prístupov k testovaniu riadeného údajmi a kľúčovými slovami sú uvedené v osnove ISTQB-TAE Advanced Level Test Automation Engineer, Fewster 1999 a Buwalda 2001.

Vyššie uvedené prístupy vyžadujú odborníka, ktorý má skúsenosti s používaním skriptovacieho jazyka (testeri, vývojári alebo špecialisti na automatizáciu testovania). Bez ohľadu na použité skriptovacie techniky sa očakávané výsledky pre každý test musia porovnať so skutočnými výsledkami testu buď dynamicky (v priebehu testu), alebo sú uložené na neskoršie porovnávanie (po vykonaní testu).

Nástroje na testovanie založené na modeli (MBT) umožňujú zachytiť funkčnú špecifikáciu vo forme modelu, ako napríklad diagram aktivít.

Vo všeobecnosti býva navrhnutý tento model návrhármi systémov. Nástroj MBT interpretuje model, tak aby vytvoril špecifikácie testovacieho prípadu, ktoré potom môžu byť uložené v nástroji manažmentu testovania a/alebo vykonané nástrojom vykonávania testov (pozri osnovu ISTQB-MBT Foundation Level Model Based Testing)

Nástroje pre manažment testovania

Nástroje pre manažment testovania musia byť často prepojené s inými nástrojmi alebo tabuľkovými procesormi z rôznych dôvodov, vrátane:

- Aby vytvárali užitočné informácie vo formáte, ktorý vyhovuje potrebám organizácie
- Aby zachovali konzistentnú sledovateľnosť požiadaviek v nástroji pre manažment požiadaviek
- Aby prepojili informácie o verzii testovaných objektov v nástroji pre manažment konfigurácie

Tieto dôvody je obzvlášť potrebné zvážiť pri používaní integrovaného nástroja (napr. manažmentu životného cyklu aplikácií), ktorý obsahuje modul pre manažment testovania (a prípadne aj systém pre manažment defektov), ako aj ďalšie moduly (napr. rozvrh projektu a informácie o rozpočte), ktoré využívajú rôzne skupiny v rámci organizácie.

6.2 Efektívne používanie nástrojov

6.2.1 Hlavné princípy výberu nástrojov

Hlavné úvahy pri výbere nástroja pre organizáciu zahŕňajú:

- Posúdenie zrelosti organizácie, jej silných a slabých stránok
- Identifikácia možností na zlepšenie testovacieho procesu s podporou nástrojov
- Pochopenie technológií používaných testovanými objektmi s cieľom vybrať nástroj, ktorý je kompatibilný s touto technológiou
- Vybudovanie a kontinuálna integrácia nástrojov, ktoré sa už v organizácii používajú, s cieľom zabezpečiť kompatibilitu a integráciu nástrojov
- Posúdenie nástroja voči jednoznačným požiadavkám a objektívnym kritériám
- Zhodnotenie, či nástroj bude k dispozícii na bezplatné skúšobné obdobie (a na ako dlho)
- Vyhodnotenie dodávateľa (vrátane školení, podpory a komerčných aspektov) alebo poskytovateľov podporných služieb v prípade nekomerčných (napr. nástrojov s otvoreným zdrojovým kódom) nástrojov
- Identifikácia interných požiadaviek pre koučing a mentoring pri používaní nástroja
- Vyhodnotenie školení potrebných s ohľadom na skúsenosti s testovaním (a automatizáciu testovania) u osôb, ktoré budú priamo pracovať s nástrojom
- Zváženie výhod a nevýhod rôznych modelov udeľovania licencií (napr. komerčných alebo open-source)
- Vyhodnotenie pomeru nákladov a výnosov, ktorý je založený na konkrétnom biznis prípade (ak je potrebné)

Posledným krokom by malo byť hodnotenie skúšky konceptu s cieľom zistiť, či nástroj funguje efektívne s testovaným softvérom a v rámci súčasnej infraštruktúry, alebo v prípade potreby identifikovať zmeny, ktoré táto infraštruktúra potrebuje na účinné využívanie nástroja.

6.2.2 Pilotné projekty pre zavedenie nástroja do organizácie

Po dokončení výberu nástroja a úspešnej skúšky konceptu, začína zavedenie vybraného nástroja do organizácie vo všeobecnosti pilotným projektom, ktorý má nasledujúce ciele:

- Dozvedieť sa viac podrobností o nástroji, pochopiť jeho silné a slabé stránky
- Zhodnotiť, ako bude nástroj zapadať do existujúcich procesov a praktík a určiť čo bude potrebné zmeniť
- Rozhodnúť o štandardnom spôsobe používania, riadenia, zálohovania a údržby nástroja a testovacích produktov (t. j. dohoda o mennej konvencii pre súbory a testy, výber kódovacích noriem, vytváranie knižníc a definícia modularity testovacích súborov)
- Posúdiť, či budú výnosy dosiahnuté za primerané náklady
- Porozumieť metrikám, ktoré chcete nástrojom zbierať a reportovať, konfigurovanie nástroja na zaistenie toho, aby tieto metriky mohli byť zachytené a reportované

6.2.3 Faktory úspechu pre zavedenie nástroja

Faktory úspechu pre hodnotenie, implementáciu, nasadenie a prebiehajúcu podporu nástrojov v rámci organizácie zahŕňajú

- Postupné zavádzanie nástroja do zvyšku organizácie
- Adaptácia a zlepšovanie procesov tak, aby zodpovedali použitiu nástroja
- Zabezpečovanie tréningov a koučovania/mentorovania pre používateľov nástroja
- Definícia pravidiel používania (napr. interné štandardy pre automatizáciu)
- Implementovanie spôsobu získavania informácií o používaní nástroja z jeho aktuálneho používania
- Monitorovanie používania nástroja a jeho výnosov
- Poskytovanie podpory používateľom pre daný nástroj
- Zbieranie získaných ponaučení od všetkých používateľov

Je tiež dôležité zabezpečiť, aby bol nástroj technicky a organizačne integrovaný do životného cyklu vývoja softvéru, ktorý môže zahŕňať samostatné organizácie zodpovedné za prevádzku a/alebo dodávateľov tretích strán.

Pre skúsenosti a rady ohľadom používania nástrojov realizácie testovania pozri Graham 2012.

7 Referencie

Štandardy

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 Foundation Level Overview 2018

ISTQB-MBT Foundation Level Model-Based Tester Extension Syllabus

ISTQB-AT Foundation Level Agile Tester Extension Syllabus

ISTQB-ATA Advanced Level Test Analyst Syllabus

ISTQB-ATM Advanced Level Test Manager Syllabus

ISTQB-SEC Advanced Level Security Tester Syllaby

ISTQB-TAE Advanced Level Test Automation Engineer Syllabus ISTQB-

ETM Expert Level Test Management Syllabus

ISTQB-EITP Expert Level Improving the Test Process Syllabus

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 Model-Based 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 NY

Ďalšie zdroje (nie priamo odkazované v sylaboch)

Black, R., van Veenendaal, E. and Graham, D. (2012) *Foundations of Software Testing: ISTQB Certification (3e)*, Cengage Learning: London UK

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

Spillner, A., Linz, T., and Schaefer, H. (2014) *Software Testing Foundations (4e)*, Rocky Nook: San Rafael CA

8 Príloha A – podklad učebnej osnovy

História dokumentu

Tento dokument je učebnou osnovou pre medzinárodný Základný certifikát v testovaní softvéru, ktorý je prvou úrovňou medzinárodnej kvalifikácie odsúhlasenou ISTQB (www.istqb.org).

Tento dokument bol pripravený počas rokov 2014–2018 pracovnou skupinou skladajúcou sa z menovaných členov Medzinárodného výboru pre kvalifikáciu softvérového testovania (International Software Testing Qualifications Board - ISTQB®). Najskôr bola verzia dokumentu 2018 revidovaná predstaviteľmi všetkých členských výborov ISTQB a následne predstaviteľmi vybranými z medzinárodnej komunity testovania softvéru.

Ciele kvalifikácie Základný certifikát

- Získať uznanie testovania ako základnej a profesionálnej špecializácie softvérového inžinierstva
- Poskytnúť štandardný framework pre kariérny rozvoj testerov
- Umožniť profesionálne kvalifikovaným testerom, aby boli rešpektovaní zamestnávateľmi, zákazníkmi a kolegami a zároveň zlepšiť postavenie testerov
- Presadzovať jednotné a vhodné testovacie postupy v rámci všetkých disciplín softvérového inžinierstva
- Identifikovať oblasti testovania, ktoré sú podstatné a prínosné pre dané odvetvia
- Umožniť dodávateľom softvéru, aby prijímali certifikovaných testerov a tak získali obchodnú výhodu voči svojej konkurencii zverejnením svojich pravidiel náboru testerov
- Poskytnúť možnosť pre testerov a ďalších, ktorí majú záujem o testovanie, aby získali medzinárodne uznávanú kvalifikáciu v predmete testovania

Ciele medzinárodnej kvalifikácie

- Umožniť porovnanie znalostí testovania v rôznych krajinách
- Umožniť testerom jednoduchšie uplatnenie v iných krajinách
- Umožniť nadnárodným/medzinárodným projektom, aby mali jednotné chápanie problematiky testovania
- Celosvetovo zvyšovať množstvo kvalifikovaných testerov
- Ako medzinárodne založená iniciatíva mať vyšší vplyv a hodnotu ako lokálny prístup v rámci jedného regiónu/krajiny
- Rozvíjať spoločný medzinárodný súbor poznatkov a vedomostí o testovaní pomocou učebnej osnovy a terminológie a zvyšovať úroveň znalostí o testovaní u všetkých zúčastnených.
- Propagovať testovanie ako profesiu v ďalších krajinách
- Umožniť testerom, aby získali uznávanú kvalifikáciu v ich rodnom jazyku
- Umožniť zdieľanie znalostí a zdrojov medzi krajinami
- Poskytovať medzinárodné uznanie testerov a ich kvalifikácie spoluúčasťou množstva krajín

Vstupné požiadavky na túto kvalifikáciu

Vstupné kritérium na získanie Základného certifikátu ISTQB® v testovaní softvéru je záujem kandidátov o testovanie softvéru. Avšak je silne odporúčané, aby kandidáti taktiež:

- Mali prinajmenšom minimálne znalosti z vývoja softvéru alebo testovania softvéru, napríklad šesťmesačnú skúsenosť ako tester systémových alebo akceptačných testov, prípadne ako softvérový vývojár.
- Absolvovali školenie, ktoré je akreditované podľa ISTQB® štandardov (jedným z uznaných lokálnych výborov ISTQB)

Pozadie a história Základného certifikátu v testovaní softvéru

Nezávislá certifikácia softvérových testerov začala vo Veľkej Británii v Skúšobnom výbore informačných systémov (Information Systems Examination Board (ISEB)) Britskej počítačovej spoločnosti (British Computer Society) založením Výboru pre testovanie softvéru (Software testing board) v roku 1998 (www.bcs.org.uk/iseb). V roku 2002 začala spoločnosť ASQF v Nemecku podporovať Nemecký program pre kvalifikáciu testerov (German tester qualification scheme www.asqf.de). Táto učebná osnova je založená na osnovách ISEB a ASQF; zahŕňa reorganizovaný, aktualizovaný obsah a ďalší obsah a dôraz je kladený na témy, ktoré poskytnú najlepšiu praktickú pomoc testerom

Už existujúci Základný certifikát v testovaní softvéru (napr. od ISEB, ASQF alebo lokálnych výborov uznaných ISTQB), ktorý bol udelený pred vydaním tohoto medzinárodného certifikátu, bude považovaný za ekvivalent medzinárodného certifikátu. Základný certifikát neexspiruje a nebude potrebné, aby bol predlžovaný alebo obnovovaný. Dátum vydania sa nachádza na certifikáte.

V rámci každej zúčastnenej krajiny sú miestne aspekty kontrolované lokálnym výborom softvérového testovania uznaným ISTQB. Povinnosti lokálnych výborov sú zadané od ISTQB, ale sú implementované v rámci každej krajiny. V povinnostiach lokálnych výborov sa predpokladá zahrnutie akreditácie poskytovateľov školení a nastavenie skúšok.

9 Príloha B – Študijné ciele /kognitívna úroveň znalostí

Pre tieto osnovy sú použité nasledujúce študijné ciele. Každá téma v učebnej osnove bude preskúšaná podľa jej zodpovedajúceho študijného cieľa.

Úroveň 1: Zapamätať si (K1)

Kandidát pochopí, zapamätá si a spomenie si na výraz, pojem alebo koncept.

Kľúčové slová: Zapamätať si, obnoviť, spomenúť si, rozoznať, vedieť

Príklady:

Pozná definíciu zlyhania ako:

- „Nedodanie služby koncovému požívateľovi alebo inej kľúčovej osobe“, alebo
- „Odchýlka komponentu alebo systému od jeho očakávanej dodávky, služby alebo výsledku“

Úroveň 2: Pochopiť (K2)

Kandidát vie označiť príčiny alebo vysvetlenia pre výroky týkajúce sa témy a vie zhrnúť, porovnať, klasifikovať, rozdeliť a uviesť príklady pre koncepty testovania.

Kľúčové slová: Zhrnúť, zovšeobecniť, abstrahovať, klasifikovať, porovnať, namapovať, odlíšiť, doložiť príkladom, vysvetliť, preložiť, znázorniť, odvodiť, vyvodiť, kategorizovať, vytvoriť model

Príklady:

Vie vysvetliť dôvod, prečo by mali byť testy vytvorené tak skoro, ako je to možné:

- Nájsť defekty keď sú lacnejšie odstrániteľné
- Nájsť najdôležitejšie defekty ako prvé

Dokáže vysvetliť podobnosti a rozdiely medzi integračným a systémovým testovaním:

- Podobnosti: testovanie viac ako jedného komponentu, môžu byť testované nefunkcionálne aspekty
- Rozdiely: integračné testovanie sa zameriava na rozhrania a interakcie a systémové testovanie sa zameriava na aspekty celého systému, ako je proces od začiatku do konca (end-to-end

Úroveň 3: Použiť (K3)

Kandidát vie vybrať správnu aplikáciu konceptu alebo techniky a aplikovať ich v danom kontexte.

Kľúčové slová: Zavádzať, vykonať, používať, sledovať postup, uplatniť postup

Príklady:

- Vie identifikovať hraničné hodnoty pre platné a neplatné sekcie
- Vie vybrať testovacie prípady pre daný diagram prechodu stavov s pokrytím všetkých prechodov

Referencie (Pre úrovne poznania študijných 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: Boston MA

10 Príloha C – Poznámky k vydaniu

ISTQB Foundation Syllabus 2018 je výraznou aktualizáciou a doplnením vydania z roku 2011. Z tohto dôvodu neexistujú žiadne podrobné poznámky k vydaniu podľa kapitol a sekcií. Zhrnutie hlavných zmien je však uvádzané tu. Okrem toho v osobitnom dokumente o vydaní poskytuje ISTQB vysledovateľnosť medzi študijnými cieľmi vo verzii osnovy z roku 2011 a študijnými cieľmi vo verzii osnovy z roku 2018, pričom je zvýraznené, ktoré ciele boli pridané, aktualizované alebo odstránené.

Na začiatku roka 2017 absolvovalo základnú skúšku viac ako 550 000 ľudí vo viac ako 100 krajinách a viac ako 500 000 z nich je certifikovanými testermi po celom svete. S predpokladom, že všetci si prečítali túto osnovu, aby mohli absolvovať skúšku, je to pravdepodobne najviac čítaný softvérový testovací dokument v histórii!

Táto dôležitá aktualizácia bola vytvorená s ohľadom na tento odkaz a s cieľom zlepšiť hodnotu dodávanú zo strany ISTQB ďalším 500 000 ľuďom v globálnej testovacej komunite.

V tejto verzii boli všetky študijné ciele upravené tak, aby sa stali atomickými (rozdrobenými) a umožniť jasnú vysledovateľnosť od každého študijného cieľa do sekcie obsahu (a otázok skúšok), ktoré súvisia s daným študijným cieľom. Zároveň bol záujem vytvoriť jasnú vysledovateľnosť od časti obsahu (a otázok skúšok) späť k súvisiacim študijným cieľom. Okrem toho sa časové rozdelenie kapitol stalo realistickejším oproti vydaniu osnovy z roku 2011, pričom sa použili osvedčené heuristiky a vzorce používané s inými osnovami ISTQB, ktoré sú založené na analýze študijných cieľov, ktoré sa majú zahrnúť do každej kapitoly.

Aj keď ide o základné učebné osnovy, ktoré vyjadrujú best practise a techniky, ktoré vydržali skúšku času, priniesli sme zmeny smerom k modernizácii prezentácie materiálu, najmä pokiaľ ide o metódy vývoja softvéru (napr. Scrum a kontinuálne zavádzanie) a technológie (napr. internet vecí). Aktualizovali sme referenčné štandardy tak, aby boli aktuálnejšie:

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

Okrem toho, keďže portfólio ISTQB sa za posledné desaťročie dramaticky rozrástlo, pridali sme v prípade potreby ďalšie rozsiahle krížové odkazy na súvisiaci materiál v iných osnovách ISTQB. Tiež sme starostlivo revidovali zosúladenie so všetkými osnovami a so slovníkom ISTQB. Cieľom je uľahčiť čítanie, pochopenie, učenie sa a prekladanie tejto verzie so zameraním na zvýšenie praktickej užitočnosti a rovnováhy medzi znalosťami a zručnosťami.

Podrobnú analýzu zmien vykonaných v tejto verzii nájdete v Prehľade učebnej osnovy pre základný stupeň ISTQB 2018.

11 Index