Sertifioitu Testaaja

Jatkotason sertifikaattisisältö Testausasiantuntija

Versio 2012
Käännösversio 2014

International Software Testing Qualifications Board

Tekijänoikeushuomautus
Tämän dokumentin saa kopioida kokonaisuudessaan tai siitä saa tehdä otteita, mikäli lähde mainitaan.

Tekijänoikeus © International Software Testing Qualifications Board (jäljempänä ISTQB®)

Jatkotaso, Testausasiantuntija alatyöryhmä Judy McKay (Chair), Mike Smith, Erik Van Veenendaal; 2010-2012.

© International Software Testing Qualifications Board

Muutoshistoria

Versio	Pvm	Huomautukset
ISEB v1.1	04SEP01	ISEB Practitioner Syllabus
ISTQB 1.2E	SEP03	ISTQB Advanced Level Syllabus from EOQ-SG
V2007	12OCT07	Certified Tester Advanced Level syllabus version 2007
D100626	26JUN10	Incorporation of changes as accepted in 2009, separation of chapters for the separate modules
D101227	27DEC10	Acceptance of changes to format and corrections that have no impact on the meaning of the sentences.
D2011	23OCT11	Change to split syllabus, re-worked LOs and text changes to match LOs. Addition of BOs.
Alpha 2012	09MAR12	Incorporation of all comments from NBs received from October release.
Beta 2012	07APR12	Incorporation of all comments from NBs received from the Alpha release.
Beta 2012	07APR12	Beta Version submitted to GA
Beta 2012	08JUN12	Copy edited version released to NBs
Beta 2012	27JUN12	EWG and Glossary comments incorporated
RC 2012	15AUG12	Release candidate version - final NB edits included
GA 2012	19OCT12	Final edits and cleanup for GA release
Versio 2012, käännös-versio 2014	15.5.2014	Ensimmäinen suomenkielinen versio

© International Software Testing Qualifications Board

Sisällysluettelo

© International Software Testing Qualifications Board

© International Software Testing Qualifications Board

Kiitokset

Tämän dokumentin on tuottanut International Software Testing Qualifications Boardin Jatkotason työryhmän Testausasiantuntija –alatyöryhmä: Judy McKay (Chair), Mike Smith, Erik Van Veenendaal

Ydintiimi kiittää katselmointitiimiä ja kansallisia hallituksia näiden ehdotuksista ja työpanoksesta.

Jatkotason sertifikaattisisällön valmistumishetkellä Jatkotason työryhmään kuuluivat seuraavat jäsenet (aakkosjärjestyksessä):

Graham Bath, Rex Black, Maria Clara Choucair, Debra Friedenberg, Bernard Homès (varapuheenjohtaja), Paul Jorgensen, Judy McKay, Jamie Mitchell, Thomas Mueller, Klaus Olsen, Kenji Onishi, Meile Posthuma, Eric Riou du Cosquer, Jan Sabak, Hans Schaefer, Mike Smith (puheenjohtaja), Geoff Thompson, Erik van Veenendaal, Tsuyoshi Yumoto.

Seuraavat henkilöt osallistuivat tämän sertifikaattisisällön katselmointiin, kommentointiin ja siihen liittyviin äänestyksiin:

Graham Bath, Arne Becher, Rex Black, Piet de Roo, Frans Dijkman, Mats Grindal, Kobi Halperin, Bernard Homès, Maria Jönsson, Junfei Ma, Eli Margolin, Rik Marselis, Don Mills, Gary Mogyorodi, Stefan Mohacsi, Reto Mueller, Thomas Mueller, Ingvar Nordstrom, Tal Pe'er, Raluca Madalina Popescu, Stuart Reid, Jan Sabak, Hans Schaefer, Marco Sogliani, Yaron Tsubery, Hans Weiberg, Paul Weymouth, Chris van Bael, Jurian van der Laar, Stephanie van Dijk, Erik van Veenendaal, Wenqiang Zheng, Debi Zylbermann.

Tämä dokumentti julkaistiin virallisesti ISTQB®:n Yleiskokouksessa 19. lokakuuta 2012.

Suomenkielisen version toteutukseen ja katselmointiin ovat osallistuneet seuraavat henkilöt:
Minna Aalto, Kari Kakkonen, Juha Pomppu, Marko Rytkönen, Laura Selonen.

© International Software Testing Qualifications Board

0. Johdatus tähän sertifikaattisisältöön

0.1 Tämän dokumentin tarkoitus

Tämä sertifikaattisisältö muodostaa pohjan Testausasiantuntijaa koskeville International Software Testing Qualification –sertifioinnin Jatkotason vaatimuksille. ISTQB® on toimittanut sertifikaattisisällön seuraavia tarkoituksia varten:

1. Kansallisille hallituksille käännettäväksi paikalliselle kielelle sekä koulutustarjoajien akkreditointia varten. Kansalliset hallitukset voivat muokata sertifikaattisisältöä tietyn kielen tarpeiden mukaisesti sekä muokata viitteitä vastaamaan paikallisia julkaisuja.
2. Koelautakunnille: Jokaisen sertifikaattisisällön oppimistavoitteita vastaavien tutkintokysymysten tuottamiseksi paikallisella kielellä.
3. Koulutustarjoajille: koulutusmateriaalin tuottamiseen sekä soveltuvien opetustapojen valitsemiseksi.
4. Sertifiointikokelaille: tutkintoon valmistautumista varten (joko osana valmennuskurssia tai itsenäisesti).
5. Kansainväliselle ohjelmisto- ja järjestelmätestaajien yhteisölle: ohjelmisto- ja järjestelmätestauksen ammatin edistämiseksi sekä kirjojen ja artikkeleiden pohjamateriaaliksi.

ISTQB® voi antaa myös muille yhteisöille luvan käyttää tätä sertifiointisisältöä muihin tarkoituksiin edellyttäen, että nämä hakevat ja saavat tarkoitukseen etukäteen kirjallisen suostumuksen.

0.2 Yleiskatsaus

Jatkotaso muodostuu kolmesta erillisestä sertifikaattisisällöstä:

- Testauspäällikkö
- Testausasiantuntija
- Tekninen testausasiantuntija

Dokumentti Jatkotason yleiskatsaus sisältää seuraavat tiedot:

- Jokaisen sertifikaattisisällön tavoitteet liiketoiminnan kannalta
- Jokaisen sertifikaattisisällön yhteenveto
- Sertifikaattisisältöjen väliset yhteydet
- Oppimistasojen (K-tasojen) kuvaus
- Liitteet

0.3 Kuulusteltavat oppimistavoitteet

Oppimistavoitteet tukevat liiketoiminnan tavoitteita ja niitä käytetään laadittaessa koetta Jatkotason Testausasiantuntijan sertifioinnin saavuttamiseksi. Yleisesti ottaen kaikki tämän sertifikaattisisällön osat ovat kuulusteltavissa K1-tasolla. Se tarkoittaa, että kokelas tunnistaa, muistaa ja pystyy palauttamaan mieleensä termin tai käsitteen. K2-, K3- ja K4-tasojen oppimistavoitteet on kuvattu kyseessä olevan luvun alussa.

© International Software Testing Qualifications Board

1. Testausprosessi – 300 min.

Avainsanat

konkreettinen testitapaus, korkean tason testitapaus, looginen testitapaus, lopetuskriteerit, matalan tason testitapaus, testauksen valvonta, testaussuunnittelu, testin suorittaminen, testin toteutus, testisuunnittelu,

Oppimistavoitteet: Testausprosessi

1.2 Testaus ohjelmistokehityksen elinkaaressa

TA-1.2.1 (K2) Selittää, kuinka ja miksi Testausasiantuntijan työpanoksen ajoitus ja määrä vaihtelevat, kun käytetään erilaisia elinkaarimalleja.

1.3 Testauksen seuranta, suunnittelu ja valvonta

TA-1.3.1 (K2) Vetää yhteen Testausasiantuntijan suorittamat tehtävät, jotka tukevat testauksen suunnittelua ja valvontaa.

1.4 Testien analysointi

TA-1.4.1 (K4) Analysoida annettu skenaario (projektin kuvaus ja elinkaarimalli mukaan luettuna) Testausasiantuntijalle analysointi- ja suunnitteluvaiheen aikana kuuluvien tehtävien määrittämiseksi.

1.5 Testaussuunnittelu

TA-1.5.1 (K2) Selittää, miksi sidosryhmien edustajien pitäisi ymmärtää testattavat tilanteet.

TA-1.5,2 (K4) Analysoida projektikuvaus matalan tason (konkreettisten) ja korkean tason (loogisten) testitapauksille soveltuvimman käytön määrittämiseksi.

1.6 Testien toteutus

TA-1.6.1 (K2) Kuvata testien analysoinnin ja suunnittelun tyypilliset lopetusehdot ja selittää, kuinka näiden ehtojen saavuttaminen vaikuttaa testien toteutuksen työmäärään.

1.7 Testin suoritus

TA-1.7.1 (K3) Määrittää annetun skenaarion suoritusaskeleet ja asiat, jotka pitää ottaa huomioon testejä suoritettaessa.

1.8 Lopetusehtojen arviointi ja raportointi

TA-1.8.1 (K2) Selittää, miksi testien suorituksen tilan tarkka raportointi on tärkeää.

1.9 Testauksen päätöstehtävät

TA-1.9.1 (K2) Antaa esimerkkejä tuotoksista, joita Testausasiantuntijan pitäisi luovuttaa testauksen päätöstehtävien aikana.

© International Software Testing Qualifications Board

1.1 Esittely

ISTQB ®:n Perustason sertifikaattisisällössä testauksen perusprosessiin määriteltiin kuuluvaksi seuraavat tehtävät:

- Suunnittelu, monitorointi ja valvonta
- Analysointi ja suunnittelu
- Valmistelu ja suoritus
- Lopetusehtojen arviointi ja raportointi
- Testauksen päätöstehtävät.

Jatkotasolla joitakin näistä tehtävistä käsitellään erillisinä, jotta prosesseista saadaan tarkempaa tietoa ja niitä voidaan optimoida, jotta tehtävät sopivat paremmin ohjelmistokehityksen elinkaareen, sekä tehokkaan testauksen monitoroinnin ja hallinnan helpottamiseksi. Tällä tasolla tehtäväjako on seuraavanlainen:

- Suunnittelu, monitorointi ja valvonta
- Analysointi
- Suunnittelu
- Valmistelu
- Suoritus
- Lopetusehtojen arviointi ja raportointi
- Testauksen päätöstehtävät.

Nämä tehtävät voidaan suorittaa peräkkäin tai jotkin niistä voidaan suorittaa rinnakkain, esim. suunnittelu voidaan tehdä yhtä aikaa valmistelun kanssa (esim. tutkiva testaus). Testausasiantuntija keskittyy tehtävissään ensisijaisesti oikeiden testien ja testitapausten määrittämiseen, suunnitteluun ja suorittamiseen. Vaikka on tärkeää ymmärtää testausprosessin muutkin vaiheet, pääosa Testausasiantuntijan työstä tehdään yleensä testausprojektin analysointi-, suunnittelu-, valmistelu- ja suoritustehtävien aikana.

Jatkotason testaajat kohtaavat monia haasteita esitellessään tässä sertifikaattisisällössä esitettyjä testauksen näkökulmia omassa organisaatiossaan, tiimissään ja omia tehtäviään suorittaessaan. On tärkeää ottaa huomioon ohjelmistokehityksen erilaiset elinkaaret samoin kuin testattavana olevan järjestelmän ominaisuudet, sillä nämä voivat vaikuttaa käytettävään testaustapaan.

1.2 Testaus ohjelmistokehityksen elinkaaressa

Pitkän tähtäimen lähestymistä testaukseen pitäisi pohtia ja se pitäisi suunnitella osana testausstrategiaa. Testausasiantuntijan mukaantulon hetki vaihtelee elinkaaren mukaan ja samoin voivat vaihdella myös osallistumisen määrä, siihen tarvittava aika sekä käytettävissä olevat tiedot ja odotukset. Koska testaus ei tapahdu erillään muusta järjestelmästä ja ympäristöstä, Testausasiantuntijan täytyy olla selvillä siitä, missä vaiheessa mitäkin tietoa toimitetaan muille organisaatioalueille, kuten:

- vaatimusten laatiminen ja hallinta – vaatimuskatselmoinnit
- projektinhallinta – aikataulutus
- kokoonpanon ja muutoksenhallinta – koontien oikeellisuuden testaus, versionhallinta
- ohjelmistokehitys – ennakointi, mitä on tulossa ja milloin
- ohjelmiston ylläpito – vianhallinta, vian läpimenoaika (eli aika vian löytymisestä vian ratkaisuun)
- tekninen tuki – vaihtoehtoisten toimintatapojen tarkka dokumentointi
- teknisen dokumentaation tuottaminen (esim. tietokannan suunnittelukuvaukset) – dokumenttien sisältöön vaikuttaminen samoin kuin dokumenttien tekninen katselmointi

© International Software Testing Qualifications Board

Testaustehtävien täytyy olla linjassa valitun ohjelmistokehitysmallin kanssa, joka voi luonteeltaan olla peräkkäinen, iteratiivinen tai inkrementaalinen. Esimerkiksi peräkkäisessä V-mallissa järjestelmään sovellettava ISTQB®:n perustestausprosessi voidaan linjata seuraavasti:

- Järjestelmätestauksen suunnittelu tapahtuu yhtäaikaisesti projektisuunnittelun kanssa, ja testauksen seuranta jatkuu, kunnes järjestelmätestauksen suoritus ja päätöstehtävät on saatu loppuun.
- Järjestelmätestauksen analysointi ja testien suunnittelu tapahtuvat yhtäaikaisesti vaatimusmäärittelyjen, järjestelmän ja arkkitehtuurin (korkean tason) suunnittelukuvausten ja yksikkötestauksen (matalan tason) suunnittelukuvausten laatimisen kanssa.
- Järjestelmätestausympäristön (esim. testipedit, testikehykset) toteutus voi alkaa järjestelmäsuunnittelun aikana, vaikka valtaosa siitä tapahtuu tyypillisesti samanaikaisesti koodauksen ja yksikkötestauksen kanssa, ja järjestelmätestauksen valmistelutehtävät jatkuvat usein aivan viime päiviin asti ennen järjestelmätestauksen suorituksen alkamista.
- Järjestelmätestauksen suoritus alkaa, kun järjestelmätestauksen aloituskriteerit on kaikki täytetty (tai niistä on luovuttu), mikä tyypillisesti tarkoittaa, että ainakin yksikkötestaus ja usein myös komponentti-integraatiotestaus ovat valmiita. Järjestelmätestauksen suoritus jatkuu, kunnes järjestelmätestauksen päätöskriteerit täyttyvät.
- Järjestelmätestauksen päätöskriteerien arviointia ja testitulosten raportointia tehdään läpi järjestelmätestauksen suorituksen, yleensä sitä tiheämmin ja nopeammin mitä lähemmäksi projektin aikaraja tulee.
- Järjestelmätestauksen päätöstehtävät suoritetaan sen jälkeen, kun päätöskriteerit ovat täyttyneet ja järjestelmätestaus on vahvistettu suoritetuksi, vaikka niitä voidaan joskus lykätä, kunnes hyväksymistestaus on suoritettu ja kaikki projektin tehtävät on saatu valmiiksi.

Iteratiiviset ja inkrementaaliset mallit eivät välttämättä noudata samaa tehtäväjärjestystä ja niissä voidaan jättää pois joitakin tehtäviä. Esimerkiksi iteratiivinen malli voi käyttää vakiotestausprosesseista supistettua mallia joka iteraatiossa. Analysointi ja testien suunnittelu, valmistelu ja testien suoritus sekä lopetusehtojen arviointi ja raportointi voidaan tehdä joka kierroksella, kun taas suunnittelu tehdään projektin alussa ja loppuraportointi projektin lopussa. Ketterässä projektissa on tyypillistä käyttää vähemmän muodollista prosessia ja tehdä paljon tiiviimpää yhteistyötä, mikä mahdollistaa muutosten tekemisen helpommin projektissa. Koska ketterä prosessi on ”kevyt”, testausdokumentaatio on vähemmän kattavaa ja sen sijaan käytetään nopeampia viestintätapoja, kuten päivittäisiä tilannepalavereja (englanniksi ”stand up meeting” eli ”seisova kokous”; palaverit ovat hyvin nopeita, kestoltaan yleensä 10 – 15 minuuttia, joten kenenkään ei tarvitse istua alas ja kaikki osallistuvat aktiivisesti kokoukseen).

Kaikista elinkaarimalleista ketterät projektit vaativat Testausasiantuntijan aikaisinta mukaantuloa. Testausasiantuntijan pitäisi tulla mukaan heti projektin käynnistysvaiheessa ja työskennellä yhdessä toteuttajien kanssa, kun nämä alkavat työskennellä arkkitehtuurin ja suunnitelmien parissa. Katselmoinnit eivät ehkä ole muodollisia, mutta niitä pidetään jatkuvasti ohjelmiston kehittyessä. Osallistumisen odotetaan jatkuvan läpi koko projektin ja siksi Testausasiantuntijan pitäisi olla tiimin käytettävissä. Tämän tiiviin yhteistyön vuoksi ketterän tiimin jäsenet ovat yleensä sitoutuneita yhteen projektiin ja mukana kaikissa kyseisen projektin tehtävissä.

Iteratiiviset/inkrementaaliset mallit vaihtelevat ketterästä lähestymistavasta, jossa odotetaan muutoksia ohjelmiston kehittyessä, iteratiivisiin/inkrementaalisiin kehitysmalleihin, jotka esiintyvät V-mallin sisällä (kutsutaan joskus sulautetuksi iteratiiviseksi). Sulautetussa iteratiivisessa mallissa Testausasiantuntijan pitäisi olla mukana tavanomaisissa suunnittelutehtävissä, mutta hän siirtyy yhä vuorovaikutteisempaan rooliin sitä mukaa, kun ohjelmistoa kehitetään, testataan, muutetaan ja se julkaistaan.

Testausasiantuntijan on tärkeä ymmärtää odotukset mukanaololleen sekä osallistumisen ajoitus, käytetäänpä mitä ohjelmistokehityksen elinkaarimallia hyvänsä. Käytössä on monia yhdistelmämalleja, kuten yllä kuvattu iteratiivisuus V-mallissa. Testausasiantuntijan täytyy usein määrittää itse tehokkain

© International Software Testing Qualifications Board

rooli ja pyrkiä sitä kohti sen sijaan, että hän pitäisi kiinni jostain tietystä mallista ja siinä esitetystä oikeasta mukaantulohetkestä.

1.3 Testauksen suunnittelu, seuranta ja valvonta

Tämä kappale keskittyy testauksen suunnittelun, seurannan ja hallinnan prosesseihin.

1.3.1 Testauksen suunnittelu

Testauksen suunnittelu tapahtuu valtaosaltaan koko testaustyön alussa ja siihen kuuluu kaikkien testauksen mission ja testausstrategiassa mainittujen tavoitteiden saavuttamiseksi tarvittavien tehtävien ja resurssien tunnistaminen ja suunnittelu. Testauksen suunnittelun aikana Testausasiantuntijan on tärkeää miettiä ja suunnitella seuraavia asioita yhdessä Testauspäällikön kanssa:

- Varmista, että testaussuunnitelmat eivät rajoitu vain toiminnalliseen testaukseen. Testaussuunnitelmassa pitäisi ottaa kantaa kaikentyyppisiin testauksiin ja ne pitäisi aikatauluttaa sopivalla tavalla. Esimerkiksi toiminnallisen testauksen lisäksi Testausasiantuntija voi olla vastuussa käytettävyystestauksesta. Tämä testaus pitää myös käsitellä testaussuunnitelmassa.
- Katselmoi testauksen työmääräarviot Testauspäällikön kanssa ja varmista, että testaus-ympäristön hankkimiselle ja sen oikeellisuuden varmistamiselle on varattu riittävästi aikaa.
- Suunnittele kokoonpanon testaus. Jos useita erityyppisiä prosessoreita, käyttöjärjestelmiä, virtuaalikoneita, selaimia ja erilaisia laitteita voidaan yhdistää moniksi erilaisiksi kokoonpanoiksi, varaudu käyttämään testaustekniikoita, jotka tuottavat riittävän kattavuuden näiden yhdistelmien osalta.
- Suunnittele dokumentaation testaus. Käyttäjille toimitetaan sekä ohjelmisto että dokumentaatio. Dokumentaation täytyy olla oikein, jotta se olisi tehokas. Testausasiantuntijan täytyy varata aikaa dokumentaation todentamiseen ja hänen täytyy ehkä työskennellä teknisten kirjoittajien kanssa ja auttaa näitä laatimaan aineisto, jota käytetään näytöstä otettaviin kuvankaappauksiin tai videopätkiin.
- Suunnittele asennustoimenpiteiden testaus. Asennustoimenpiteet samoin kuin varmuuskopiointi- ja palautustoimenpiteet pitää testata riittävästi. Nämä toimenpiteet voivat olla kriittisempiä kuin itse ohjelmisto; jos ohjelmistoa ei voida asentaa, sitä ei voida myöskään käyttää. Suunnittelu voi olla vaikeaa, sillä Testausasiantuntija testaa usein esiasennettua järjestelmää, jossa lopulliset asennustoimenpiteet eivät ole mukana.
- Suunnittele testaus niin, että se on linjassa ohjelmiston elinkaaren kanssa. Tehtävien perättäinen suoritus ei sovi useimpiin aikatauluihin. Monet tehtävät täytyy usein suorittaa (ainakin osittain) samanaikaisesti. Testausasiantuntijan täytyy ottaa huomioon valittu elinkaarimalli sekä odotukset sen suhteen, kuinka paljon hän osallistuu ohjelmiston suunnitteluun, toteutukseen ja käyttöönottoon. Tämä tarkoittaa myös ajan varaamista uudelleen- ja regressiotestausta varten.
- Varaa riittävästi aikaa riskien tunnistamiseen ja analysointiin yhdessä muiden asianomaisten tiimien kanssa. Vaikka Testausasiantuntija ei yleensä ole vastuussa riskienhallintakokousten järjestämisestä, hänen pitäisi kuitenkin osallistua aktiivisesti näihin tehtäviin.

Testauksen pohjamateriaalin, testattavien tilanteiden ja testitapausten välillä voi olla monimutkaisia yhteyksiä, jopa niin, että niiden välillä voi esiintyä ”monen suhde moneen” –tilanteita. Ne on ymmärrettävä, jotta testauksen suunnittelu ja hallinta voidaan tehdä tehokkaasti. Testausasiantuntija on yleensä paras henkilö määrittämään nämä suhteet ja erottelemaan näitä riippuvuuksia niin paljon kuin mahdollista.

© International Software Testing Qualifications Board

1.3.2 Testauksen seuranta ja hallinta

Vaikka testauksen seuranta ja hallinta ovatkin yleensä Testauspäällikön tehtäviä, Testausasiantuntija tuottaa mittaritiedot, jotka tekevät nämä tehtävät mahdollisiksi.

Monenlaista numeerista tietoa pitäisi kerätä läpi ohjelmistokehityksen elinkaaren (esim. valmistuneiden suunnittelutehtävien prosenttiosuus, saavutettu kattavuusprosentti, läpäistyjen ja hylättyjen testitapausten määrä). Joka tilanteessa pitää määritellä lähtötaso (eli vertailukohta), ja edistymistä seurataan ja verrataan kyseiseen tasoon. Testauspäällikkö keskittyy yleensä mittaritietojen yhteenvetämiseen ja niistä raportointiin, kun taas Testausasiantuntija kerää aineiston jokaista mittaria varten. Jokainen suoritettu testitapaus, jokainen kirjoitettu vikaraportti ja jokainen saavutettu tarkistuspiste tulee näkyviin koko projektin mittareissa. On tärkeää, että erilaisiin seurantatyökaluihin viedyt tiedot ovat niin tarkkoja ja oikein kuin mahdollista, jotta mittarit kuvaavat todellista tilannetta.

Tarkat mittaritiedot antavat johtajille mahdollisuuden ohjata projektia (seuranta, monitorointi) ja tarpeen mukaan tehdä muutoksia (hallinta). Esimerkiksi ohjelmiston joltakin alueelta raportoitujen vikojen suuri määrä voi olla merkki siitä, että kyseisellä alueella tarvitaan lisätestausta. Vaatimus- ja riskikattavuustietoja (jäljitettävyys) voidaan käyttää jäljellä olevien töiden priorisointiin ja resurssien sijoitteluun. Perussyyanalyysiä käytetään prosessikehitystä tarvitsevien alueiden määrittämiseen. Jos tallennetut tiedot ovat paikkansapitäviä, projektia voidaan hallita ja tarkat tilatiedot voidaan raportoida sidosryhmille. Tulevat projektit voidaan suunnitella tehokkaammin, kun aikaisemmista projekteista kerätyt tiedot voidaan ottaa huomioon suunnittelussa. Tarkoille tiedoille löytyy suuri joukko erilaisia käyttökohteita. Osa Testausasiantuntijan työtä on varmistaa, että tiedot ovat oikein, ne saadaan oikeaan aikaan, ja että ne ovat objektiivisia.

1.4 Testien analysointi

Testauksen suunnittelun aikana määritetään testausprojektin puitteet. Testausasiantuntija käyttää tätä rajausta

- testauksen pohjamateriaalin analysointiin
- testattavien tilanteiden tunnistamiseen.

Jotta Testausasiantuntija pystyy tekemään testianalyysin tehokkaasti, seuraavien aloituskriteerien pitäisi täyttyä:

- on olemassa dokumentti, joka kuvaa testattavan kohteen ja jota voidaan käyttää testauksen pohjamateriaalina
- dokumentti on läpäissyt katselmoinnit kohtuullisin tuloksin ja sitä on päivitetty katselmoinnin jälkeen tarvittavissa määrin
- käytettävissä on järkevä budjetti ja aikataulu tämän testattavan kohteen jäljellä olevan työn suorittamiseksi.

Testattavat tilanteet tunnistetaan tyypillisesti testauksen pohjamateriaalin ja testattavan kohteen analyysin perusteella. Tilanteissa, joissa dokumentaatio saattaa olla vanhaa tai sitä ei ole, testattavat tilanteet voidaan tunnistaa keskustelemalla oleellisten sidosryhmien kanssa (esim. työpajoissa tai sprintin suunnittelun aikana). Näitä tilanteita käytetään sitten testausstrategiassa ja/tai testaussuunnitelmassa nimettyjen testisuunnittelutekniikoiden kanssa sen päättämiseksi, mitä testataan.

Vaikka testattavat tilanteet riippuvat yleensä testauksen kohteesta, on olemassa joitakin vakioasioita, joita Testausasiantuntijan on pohdittava.

- Testattavien tilanteiden suunnittelussa on yleensä suositeltavaa käyttää vaihtelevia yksityiskohtaisuuden tasoja. Alkuvaiheessa tunnistetaan korkean tason testitapauksia, jotka kohdistuvat yleisiin testauksen kohteisiin, kuten “näytön x toiminnallisuus”. Jatkossa tunnistetaan yhä yksityiskohtaisempia tilanteita testitapausten perustaksi, kuten “näyttö x hylkää tilinumeron, josta puuttuu yksi merkki, jotta se olisi oikean pituinen”. Tämän tyyppisen

© International Software Testing Qualifications Board

hierarkkisen lähestymistavan käyttäminen testattavien tilanteiden määrittelyssä auttaa varmistamaan, että kattavuus on riittävä korkean tason kohteille.
- Jos tuoteriskit on määritetty, on tunnistettava myös jokaisen tuoteriskin käsittelyyn tarvittavat testattavat tilanteet ja jäljitettävä niiden yhteys riskiaiheeseen.

Analysointitehtävien päättyessä Testausasiantuntijan pitäisi tietää, mitä määrättyjä testejä täytyy suunnitella, jotta testausprojektin määrättyjen alueiden tarpeet tulevat täytetyiksi.

1.5 Testisuunnittelu

Testausprosessi jatkuu testaussuunnittelussa määritettyjen rajausten mukaisesti, kun Testausasiantuntija suunnittelee testit, jotka toteutetaan ja suoritetaan. Testisuunnitteluprosessiin kuuluvat seuraavat tehtävät:
- Sen määrittäminen, millä alueilla matalan tason (konkreettiset) tai korkean tason (loogiset) testitapaukset ovat soveliaimpia.
- Tarvittavan testikattavuuden tuottavien testisuunnittelutekniikoiden määrittäminen.
- Tunnistetut testattavat tilanteet toteuttavien testitapausten luominen.

Riskianalyysin ja testisuunnittelun aikana tunnistettuja priorisointikriteereitä pitäisi noudattaa läpi prosessin, analysoinnista ja suunnittelusta toteutukseen ja suoritukseen.

Suunniteltavien testien tyypistä riippuen yksi testisuunnittelun aloituskriteereistä voi olla suunnittelutyön aikana käytettävien työkalujen saatavuus.

Testien suunnittelussa on tärkeää muistaa seuraavat asiat:
- Joidenkin testattavien nimikkeiden testaamiseen on parempi käyttää vain testattavia tilanteita sen sijaan, että edettäisiin pidemmälle skriptattujen testien määrittelyyn. Tällaisessa tilanteessa testattavat tilanteet pitää määritellä niin, että niitä voidaan käyttää ohjaamaan skriptaamatonta testausta.
- Hyväksymis- ja hylkäyskriteerit pitää määritellä selkeästi.
- Testit pitää suunnitella niin, että myös muut testaajat ymmärtävät ne, ei vain niiden laatija. Jos testien laatija ei ole sama henkilö, joka suorittaa ne, toisten testaajien täytyy lukea ja ymmärtää aiemmin suunnitellut testit, jotta he ymmärtävät testauksen tavoitteet ja testin merkityksen.
- Myös muiden sidosryhmien edustajien pitää ymmärtää testit, kuten esimerkiksi toteuttajien, jotka katselmoivat testit, ja auditoijien, joiden täytyy ehkä hyväksyä testit.
- Testit pitää suunnitella kattamaan koko ohjelmiston ja toimijoiden (esim. loppukäyttäjien, toisten järjestelmien) välinen vuorovaikutus, ei pelkästään käyttäjälle näkyvän käyttöliittymän kautta tapahtuva yhteistoiminta. Prosessien välinen viestintä, eräajot ja muut keskeytykset ovat myös vuorovaikutuksessa ohjelmiston kanssa ja saattavat sisältää vikoja, joten Testausasiantuntijan täytyy suunnitella testit näiden riskien pienentämiseksi.
- Testit pitäisi suunnitella testaamaan testattavien kohteiden välisiä eri rajapintoja.

1.5.1 Konkreettiset ja loogiset testitapaukset

Yksi Testausasiantuntijan tehtävistä on määrittää tiettyyn tilanteeseen parhaat testitapaustyypit. Konkreettiset testitapaukset sisältävät kaiken yksityiskohtaisen tiedon ja tarvittavat proseduurit (mukaan luettuna aineistovaatimukset), jotta testaaja voi suorittaa testitapauksen ja todentaa tulokset. Konkreettiset testitapaukset ovat hyödyllisiä, kun vaatimukset ovat hyvin määriteltyjä, testaajilla on vähemmän kokemusta ja kun vaaditaan testitulosten ulkoista todentamista, kuten auditointia. Konkreettisten testitapausten toistettavuus on erinomainen (eli toinen testaaja saa samat tulokset), mutta ne voivat vaatia myös huomattavasti ylläpitoa ja ne saattavat rajoittaa testaajan kekseliäisyyttä suorituksen aikana.

© International Software Testing Qualifications Board

Loogiset testitapaukset tarjoavat suuntaviivat sille, mitä pitäisi testata, mutta antavat Testausasiantuntijalle mahdollisuuden vaihdella testiaineistoa tai jopa testin suorituksen aikana noudatettavaa proseduuria. Loogiset testitapaukset voivat tarjota paremman kattavuuden kuin konkreettiset testitapaukset, koska ne muuttuvat jonkin verran joka kerta, kun ne suoritetaan. Tämä johtaa toisaalta myös toistettavuuden heikkenemiseen. Loogiset testitapaukset sopivat parhaiten käytettäväksi silloin, kun vaatimukset ovat heikosti määriteltyjä, testit suorittavalla Testausasiantuntijalla on kokemusta sekä testauksesta että tuotteesta, ja kun muodollista dokumentointia ei vaadita (esim. auditointeja ei suoriteta). Loogiset testitapaukset voidaan suunnitella vaatimusprosessin aikaisessa vaiheessa, kun vaatimuksia ei ole vielä määritelty tarkasti. Näitä testitapauksia voidaan käyttää konkreettisten testitapauksen kehittämiseen, kun vaatimukset tarkentuvat ja muuttuvat vakaammiksi. Silloin testitapausten luominen tapahtuu vaiheittain edeten loogisesta konkreettiseen, ja ainoastaan konkreettisia testitapauksia käytetään testien suoritukseen.

1.5.2 Testitapausten luominen

Testitapaukset suunnitellaan laajentamalla ja tarkentamalla tunnistettuja testattavia tilanteita testausstrategiassa ja/tai testaussuunnitelmassa mainittuja testisuunnittelutekniikoita (ks. luku 3) käyttämällä. Testitapausten pitäisi olla toistettavia ja todennettavia ja ne pitäisi pystyä jäljittämään takaisinpäin testauksen pohjamateriaaliin (esim. vaatimuksiin) käytettävän testausstrategian määrittämällä tavalla.

Testitapausten suunnitteluun kuuluu seuraavien asioiden määrittäminen:

- tavoite
- esiehdot, kuten vaatimukset projektin testiympäristölle tai lokalisoidulle ympäristölle ja suunnitelmat sen toimitukselle, järjestelmän tila jne.
- vaatimukset testiaineistolle (sekä testitapausten syöteaineisto että aineisto, joka on oltava järjestelmässä, jotta testitapaus voidaan suorittaa)
- odotetut tulokset
- jälkiehdot, kuten tiedot, joihin testin suoritus on vaikuttanut, järjestelmän tila, jatkoprosessoinnin käynnistävät tekijät jne.

Testitapausten laatimisen kustannuksiin sekä suorituksen aikaiseen toistettavuuteen vaikuttava testien yksityiskohtaisuuden taso pitäisi määrittää ennen varsinaista testitapausten luomista. Testitapauksen matalampi yksityiskohtaisuuden taso sallii Testausasiantuntijalle enemmän joustavuutta testitapausten suorituksessa ja antaa mahdollisuuden tutkia mahdollisesti kiinnostavia alueita. Matalampi yksityiskohtaisuuden taso johtaa yleensä kuitenkin myös huonompaan toistettavuuden.

Erityisen haasteellista on usein testin odotettujen tulosten määrittäminen. Niiden laskeminen manuaalisesti on usein työlästä ja virheille altista; jos mahdollista, on suositeltavaa etsiä tai rakentaa automatisoitu testioraakkeli. Tunnistaessaan odotettuja tuloksia testaajat eivät ole kiinnostuneita pelkästään näytölle tulevista tulosteista, vaan myös aineiston ja ympäristön jälkiehdoista. Jos testauksen pohjamateriaali on selkeästi määritetty, oikeiden tulosten määrittämisen pitäisi teoreettisesti olla yksinkertaista. Testauksen pohjamateriaali on kuitenkin usein epämääräistä, ristiriitaista, se ei kata keskeisiä alueita, tai koko aineisto puuttuu. Tällaisessa tilanteessa Testausasiantuntijalla tai hänen käytettävissään täytyy olla asiaosaamista. Monimutkaiset syötteiden ja vasteiden vuorovaikutukset voivat myös tehdä odotettujen tulosten määrittelystä vaikeaa, vaikka testauksen pohjamateriaali olisikin hyvin määritelty; siksi testioraakkeli on välttämätön. Testitapausten suorituksella ilman minkäänlaista keinoa määritellä tulosten oikeellisuutta on hyvin vähän lisäarvoa tai hyötyä, ja usein tuloksena syntyy vääriä vikaraportteja tai väärä luottamus järjestelmään.

Yllä kuvatut tehtävät koskevat kaikkia testaustasoja, vaikka testauksen pohjamateriaali vaihteleekin. Esimerkiksi käyttäjän hyväksymistestit pohjautuvat pääasiassa vaatimusmäärittelyihin, käyttötapauksiin ja määriteltyihin liiketoimintaprosesseihin, kun taas yksikkötestit voivat pohjautua pääasiassa alemman tason suunnittelukuvauksiin, käyttäjätarinoihin ja itse koodiin. On tärkeä

© International Software Testing Qualifications Board

muistaa, että em. tehtävät tapahtuvat läpi kaikkien testaustasojen, vaikka testauksen tavoite voi vaihdella. Esimerkiksi yksikkötestaustasolla toiminnallinen testaus on suunniteltu varmistamaan, että tietty komponentti tuottaa määrätyn toiminnallisuuden niin kuin se on kuvattu kyseisen komponentin yksityiskohtaisissa suunnittelukuvauksissa. Integraatiotestaustasolla toiminnallisessa testauksessa varmistetaan, että komponentit toimivat toistensa kanssa ja tuottavat vuorovaikutuksellaan toimintaa. Järjestelmätestaustasolla testauksen kohteena pitäisi olla päästä-päähän-toiminnallisuus. Testejä analysoitaessa ja suunniteltaessa on tärkeä muistaa testin tavoitetaso samoin kuin sen tavoite. Tämä auttaa määrittämään tarvittavan yksityiskohtaisuuden tason samoin kuin mahdollisesti tarvittavat työkalut (esim. yksikkötestaustasolla tyngät ja ajurit).

Testattavien tilanteiden ja testitapausten laatimisen aikana syntyy tyypillisesti jonkin verran dokumentaatiota, jonka pohjalta syntyy testauksen tuotoksia. Käytännössä tuotosten dokumentoinnin määrä vaihtelee huomattavasti. Tähän saattavat vaikuttaa seuraavat seikat:

- projektiriskit (mitä täytyy/ei täydy dokumentoida)
- “lisäarvo”, jonka dokumentaatio tuottaa projektille
- noudatettavat standardit ja/tai täytettävät säädökset
- käytetty elinkaarimalli (esim. ketterä lähestymistapa tavoittelee “juuri riittävää” dokumentaatiota)
- vaadittu jäljitettävyys testauksen pohjamateriaalista testien analysointiin ja suunnitteluun.

Testauksen tavoitteiden ja rajojen pohjalta testien analysointi ja suunnittelu tarkastelee testauksen kohteen/kohteiden laatuominaisuuksia. ISO 25000 -standardi [ISO25000] (joka korvaa ISO 9126:n) tarjoaa hyödyllisen viitekehyksen. Lisäominaisuuksia voi olla tarpeen ottaa huomioon, kun testataan laitteisto-/ohjelmistojärjestelmiä.

Testien analysointia ja suunnittelua voidaan tehostaa liittämällä ne katselmointeihin ja staattiseen analyysiin. Itse asiassa testien analysointi ja suunnittelu ovat usein eräänlainen staattisen testauksen muoto, koska tämän prosessin aikana voi testauksen pohjadokumentaatiosta löytyä ongelmia. Vaatimusmäärittelyihin pohjautuva testien analysointi ja suunnittelu on erinomainen tapa valmistautua vaatimuskatselmointipalaveriin. Vaatimusten lukeminen ja käyttäminen testien luomiseen edellyttävät vaatimuksen ymmärtämistä ja sitä, että pystytään määrittelemään tapa, jolla arvioidaan, onko vaatimus täytetty. Tämä tehtävä paljastaa usein vaatimuksia, jotka ovat epäselviä, eivät ole testattavia, tai joille ei ole määritelty hyväksymiskriteerejä. Samoin myös muut testauksen tuotokset kuten testitapaukset, riskianalyysit ja testisuunnitelmat pitäisi katselmoida.

Joissain, esimerkiksi ketterää elinkaarimallia noudattavissa projekteissa, vaatimukset voivat olla hyvin suppeasti dokumentoituja. Ne ovat joskus ”käyttäjätarinoiden” muodossa, jolloin ne kuvaavat toiminnallisuuden pieniä mutta todennettavia osia. Käyttäjätarinan pitäisi sisältää hyväksymiskriteerien kuvaus. Jos voidaan osoittaa, että ohjelmisto täyttää hyväksymiskriteerit, sitä pidetään yleensä valmiina integroitavaksi muuhun valmiiseen toiminnallisuuteen, tai se voi olla jo integroitu siihen toiminnallisuuden osoittamiseksi.

Testisuunnittelun aikana voidaan määritellä testiympäristön yksityiskohtaiset vaatimukset, vaikka käytännössä nämä viimeistellään ehkä vasta testien valmisteluvaiheessa. On muistettava, että testiympäristö sisältää paljon muutakin kuin testauksen kohteen ja testimateriaalin. Ympäristövaatimuksiin voivat kuulua esimerkiksi tilat, laitteet, henkilöstö, ohjelmisto, työkalut, oheislaitteet, viestintävälineet, käyttövaltuudet ja kaikki muut testien suorittamiseksi tarvittavat asiat.

Testien analysoinnin ja suunnittelun päätöskriteerit vaihtelevat projektin ominaisuuksien mukaan, mutta kaikkien kahdessa edeltävässä kappaleessa käsiteltyjen seikkojen ottamista mukaan päätöskriteereihin pitäisi miettiä. On tärkeää, että kriteerit ovat mitattavat ja että varmistetaan, että kaikki seuraavissa vaiheissa tarvittavat tiedot ovat saatavilla ja valmistelutehtävät on tehty.

© International Software Testing Qualifications Board

1.6 Testien toteutus

Testien valmistelu tarkoittaa testisuunnitelmien täytäntöönpanoa. Tähän kuuluu automatisoitujen testien luominen, testien (sekä manuaalisten että automatisoitujen) järjestäminen suoritusjärjestykseen, testiaineiston ja -ympäristön viimeistely sekä testausaikataulun laatiminen resurssien allokointi mukaan luettuna, jotta testauksen suoritus voi alkaa. Siihen kuuluu myös tilanteen tarkistus kyseisen testaustason eksplisiittisiä ja implisiittisiä aloituskriteerejä vastaan ja sen varmistaminen, että prosessin edellisten tehtävien päätöskriteerit ovat täyttyneet. Jos kaikkia päätöskriteereitä ei ole noudatettu joko testaustasolla tai testausprosessin jonkin tehtävän osalta, valmistelutehtäviin kohdistuu todennäköisesti aikatauluviivästyksiä, laadullisia puutteita sekä odottamattomia lisätöitä. On tärkeää varmistaa, että kaikki päätöskriteerit on täytetty ennen kuin testien valmistelutyö alkaa.

Suoritusjärjestystä määriteltäessä joudutaan ehkä ottamaan huomioon monia seikkoja. Joissain tapauksissa voi olla järkevää järjestää testitapaukset testijoukoiksi (testitapausten ryhmiksi). Tämä voi auttaa testauksen organisoinnissa niin, että toisiinsa liittyvät testit suoritetaan yhdessä. Jos käytetään riskipohjaista testausstrategiaa, riskien priorisointijärjestys voi määritellä testitapausten suoritusjärjestyksen. Voi olla myös muita tekijöitä, jotka vaikuttavat järjestykseen, kuten oikeiden ihmisten, laitteiston, aineiston ja testattavan toiminnallisuuden saatavuus. Ei ole mitenkään poikkeuksellista, että koodia julkaistaan osissa, ja testaus täytyy koordinoida sen mukaan, missä järjestyksessä ohjelmisto saadaan testattavaksi. Erityisesti inkrementaalisissa elinkaarimalleissa on tärkeää, että Testausasiantuntija koordinoi tehtävät kehitystiimin kanssa sen varmistamiseksi, että ohjelmisto julkaistaan testaukseen testattavassa järjestyksessä. Testauksen valmistelun aikana Testausasiantuntijan pitää viimeistellä ja varmistaa järjestys, jossa manuaaliset ja automatisoidut testit tullaan suorittamaan, ja ottaa erityisesti huomioon rajoitteet, jotka voivat vaatia testien suorittamista määrätyssä järjestyksessä. Riippuvuudet pitää dokumentoida ja tarkistaa.

Testitapausten ja testattavien tilanteiden yksityiskohtaisuus voi vaikuttaa testauksen valmisteluun liittyvän työn yksityiskohtaisuuteen ja monimutkaisuuteen. Joissain tilanteissa on noudatettava erilaisia säädöksiä ja testien pitää osoittaa, että tilanteeseen kohdistuvia standardeja on noudatettu, kuten esimerkiksi Yhdysvaltain liittovaltion ilmailuhallituksen DO-178B/ED 12B [RTCA DO-178B/ED12B).

Kuten yllä on kerrottu, testauksen aikana tarvittavan testiaineiston ja joissain tilanteissa aineistojoukkojen määrä voi olla varsin suuri. Testauksen valmistelun aikana Testausasiantuntija luo syöteaineistoa sekä muuta ympäristöön liittyvää aineistoa, joka voidaan tallentaa tietokantoihin ja muihin tietovarastoihin. Testausasiantuntijat luovat myös aineistoa, jota käytetään aineisto-ohjatuissa automatisoiduissa testeissä sekä manuaalisissa testeissä.

Testauksen valmisteluun liittyy myös testiympäristö(i)stä huolehtiminen. Testiympäristö(t) pitäisi pystyttää valmiiksi tässä vaiheessa ja verifioida ennen testien suoritusta. ”Tarkoitukseen sopiva” testiympäristö on olennainen asia, eli testiympäristön pitäisi mahdollistaa vikojen löytyminen ohjatun testauksen aikana, toimia normaalisti silloin, kun häiriöitä ei esiinny, ja tarpeen vaatiessa kahdentaa tuotanto- tai loppukäyttäjän ympäristö ylemmän tason testauksia varten. Testiympäristön muutokset testauksen aikana voivat olla tarpeellisia riippuen muista odottamattomista muutoksista, testituloksista tai muista huomioon otettavista seikoista. Jos ympäristöjä muutetaan testien suorituksen aikana, on tärkeää arvioida muutosten vaikutus aiemmin suoritettuihin testeihin.

Testaajien on varmistettava testien valmistelun aikana, että tiedetään, ketkä ovat vastuussa testiympäristön luomisesta ja ylläpidosta ja että nämä henkilöt ovat käytettävissä, ja että kaikki testauksen materiaalit ja testausta tukevat työkalut ja muut prosessit ovat valmiita käytettäviksi. Näihin kuuluvat kokoonpanonhallinta, havaintojenhallinta ja testien kirjaaminen ja hallinta. Lisäksi Testausasiantuntijan täytyy todentaa aineiston keräämistoimenpiteet päätöskriteerien arviointia ja testitulosten raportointia varten.

© International Software Testing Qualifications Board

Testien valmisteluun on viisasta käyttää tasapainotettua lähestymistapaa sen mukaan, miten testaussuunnittelun aikana on päätetty. Esimerkiksi riskipohjaiset analyyttiset testausstrategiat yhdistetään usein dynaamisten testausstrategioiden kanssa. Tällaisessa tapauksessa jokin osuus testien valmisteluun varatusta työmäärästä käytetään testaukseen, joka ei seuraa ennalta määrättyjä skriptejä (skriptaamaton testaus).

Skriptaamaton testaus ei saisi olla täysin satunnaista tai tavoitteetonta, sillä tällaisen testauksen kesto ja kattavuus voivat olla ennalta arvaamattomia, ellei niitä rajata ajallisesti ja ohjata testausohjeen avulla. Vuosien saatossa testaajat ovat kehittäneet erilaisia kokemusperusteisia tekniikoita, kuten hyökkäykset, virheenarvaus [Myers79] ja tutkiva testaus. Testien analysointi, suunnittelu ja valmistelu tehdään edelleen, mutta ne tapahtuvat pääasiassa testien suorituksen aikana.

Tällaisia dynaamisia testausstrategioita noudatettaessa jokaisen testin tulokset vaikuttavat seuraavien testien analysointiin, suunnitteluun ja valmisteluun. Vaikka tällaiset strategiat ovat kevyitä ja niiden avulla löydetään usein tehokkaasti vikoja, niihin liittyy myös haittoja. Ne vaativat Testausasiantuntijalta kokemusta, testauksen kestoa voi olla vaikea ennustaa, kattavuutta voi olla vaikeasti seurata ja toistettavuus voidaan menettää ilman hyvää dokumentaatiota tai työkalutukea.

1.7 Testin suoritus

Testien suoritus alkaa, kun testauksen kohde on toimitettu ja suorituksen aloituskriteerit on täytetty (tai niistä on luovuttu). Testit pitäisi suorittaa valmisteluvaiheen aikana määritetyssä järjestyksessä, mutta Testausasiantuntijalla pitää olla riittävästi aikaa, jotta hän voi varmistaa, että testauksen aikana esiin tulevat muut mielenkiintoiset skenaariot ja järjestelmän käyttäytyminen tulevat katetuiksi (tällaisten poikkeamien aikana löydetyt häiriöt pitää kuvata ja kuvaukseen pitää liittää skriptatusta testitapauksesta tehdyt muutokset, jotka tarvitaan häiriön toistamiseksi). Tämä skriptatun ja skriptaamattoman (esim. tutkivan) testauksen yhdistäminen auttaa suojautumaan mahdollisilta aukoilta skriptatun testauksen kattavuudessa ja kiertämään hyönteismyrkkyparadoksin.

Testauksen suoritustehtävien ytimessä on todellisten tulosten vertailu odotettuihin. Testausasiantuntijan täytyy kiinnittää huomiota tähän tehtävään, muutoin kaikki testien suunnitteluun ja valmisteluun käytetty työ saattaa mennä hukkaan, kun häiriöt jäävät huomaamatta (väärä negatiivinen tulos) tai oikea käytös tulkitaan virheellisesti vääräksi (väärä positiivinen tulos). Jos odotetut ja todelliset tulokset eivät täsmää, on syntynyt havainto. Havainnot täytyy tutkia huolellisesti niiden syyn määrittämiseksi (syy voi ehkä olla vika testattavassa kohteessa tai toisaalta jossain muualla) ja havainnon käsittelyä auttavan aineiston keräämiseksi (lisätietoja havaintojenhallinnasta, ks. luku 6).

Testidokumentaatio (testisuunnitelma, testitapaus jne.) pitää käydä huolellisesti läpi sen oikeellisuuden varmistamiseksi, jos testauksessa ilmenee häiriö. Testidokumentti voi olla monestakin syystä virheellinen. Jos dokumentaatio on virheellinen, se pitää korjata ja testi pitää suorittaa uudelleen. Koska muutokset testauksen pohjamateriaalissa ja testattavassa kohteessa voivat aiheuttaa aikaisemmin monta kertaa hyväksytysti suoritetun testitapauksen muuttumisen virheelliseksi, testaajan täytyy pitää mielessä, että tehdyt havainnot voivat johtua myös virheellisestä testistä.

Testin suorituksen aikana testitulokset pitää kirjata sovitulla tavalla. Jos testit suoritetaan, mutta niiden tuloksia ei kirjata, ne voidaan joutua toistamaan oikeiden tulosten tunnistamiseksi, mikä johtaa tehottomuuteen ja viivästyksiin. (Huomaa, että riittävällä tietojen kirjaamisella voidaan kiinnittää huomiota kattavuuteen ja toistettavuuteen liittyviin huolenaiheisiin, jotka koskevat sellaisia testaustekniikoita kuin esim. tutkiva testaus.) Koska testauksen kohde, testausmateriaali ja testiympäristöt voivat kaikki muuttua, pitäisi erityisesti kirjata ylös testauksessa käytetyt versiot samoin kuin erityiset laitteistokokoonpanot. Tulosten kirjaaminen tuottaa kronologiset tiedot testien suoritukseen liittyvistä keskeisistä yksityiskohdista.

© International Software Testing Qualifications Board

Tulosten kirjaaminen koskee sekä yksittäisiä testejä että tehtäviä ja tapahtumia. Jokaisen testin pitäisi olla yksilöivästi nimetty ja sen tila pitää merkitä ylös sitä mukaa, kun testaus etenee. Kaikki tapahtumat, jotka vaikuttavat testin suoritukseen, pitää kirjata ylös. Testikattavuuden mittaamiseksi pitää kirjata riittävät tiedot ja syyt testauksen viivästyksiin ja keskeytyksiin on raportoitava. Lisäksi on kirjattava ylös tiedot, jotka tukevat testauksen hallintaa, edistymisen raportointia, päätöskriteerien mittaamista sekä testausprosessin kehittämistä.

Tietojen kirjaaminen vaihtelee testitason ja strategian mukaan. Jos esimerkiksi tehdään automatisoitua yksikkötestausta, automatisoidut testit tuottavat suurimman osan tallennettavista tiedoista. Jos tehdään manuaalista testausta, Testausasiantuntija kirjaa testien suoritusta koskevat tiedot usein testauksenhallintavälineeseen, jolla seurataan testauksen suoritusta. Kuten testien valmistelun kohdallakin, testien suorituksesta kirjattavaan tietojen määrään voivat joissain tapauksissa vaikuttaa säädöksiin tai auditointeihin liittyvät vaatimukset.

Joissain tapauksissa käyttäjät tai asiakkaat voivat osallistua testien suoritukseen. Tämä voi palvella keinona kasvattaa heidän luottamustaan järjestelmään, mutta silloin lähtökohtaisesti oletetaan, että testit löytävät vain vähän vikoja. Tällainen oletus on usein mahdoton aikaisissa testausvaiheissa, mutta voi olla kelvollinen hyväksymistestauksessa.

Seuraavassa on lueteltu joitakin erityisalueita, joita pitäisi ottaa huomioon testien suorituksen aikana:

- Kiinnitä huomioita “vähäpätöisiin” kummallisuuksiin ja tutki niitä. Huomiot tai tulokset, jotka voivat vaikuttaa epäolennaisilta, ovat usein merkkejä vioista, jotka väijyvät pinnan alla (kuten jäävuoret).
- Tarkista, että tuote ei tee sellaista, mitä sen ei pitäisi tehdä. Testauksen normaalina paino-pisteenä on tarkistaa, että tuote tekee sen, mitä sen pitääkin, mutta Testausasiantuntijan täytyy olla myös varma siitä, että ohjelma ei toimi väärin tekemällä jotain, mitä sen ei pitäisi tehdä (esimerkiksi ylimääräiset ei-toivotut tapahtumat).
- Rakenna testijoukko ja valmistaudu siihen, että se kasvaa ja muuttuu ajan myötä. Koodi kehittyy ja lisätestejä täytyy laatia uusien toiminnallisuuksien kattamiseksi samoin kuin ohjelmiston muiden osien tarkistamiseksi taantumisen (regression) varalta. Suorituksen aikana löydetään usein aukkoja testauksessa. Testijoukon rakentaminen on jatkuva prosessi.
- Tee muistiinpanoja seuraavaa testaustyötä varten. Testaustehtävät eivät pääty siihen, kun ohjelmisto luovutetaan käyttäjälle tai jaellaan markkinoille. Todennäköisesti ohjelmistosta tehdään uusi versio tai julkaisu, joten osaaminen pitäisi tallentaa ja siirtää testaajille, jotka ovat vastuussa seuraavasta testaustyöstä.
- Älä odota, että kaikki manuaaliset testit suoritetaan uudelleen. On epärealistista kuvitella, että kaikki manuaaliset testit testataan uudelleen. Jos epäillään ongelmaa, Testausasiantuntijan pitää tutkia tilanne ja merkitä tapahtumat muistiin ennemmin kuin olettaa, että tilanne jäisi kiinni testitapausten uudelleensuorituksessa.
- Pengo vianhallintavälineen materiaalia lisätestitapauksia varten. Harkitse testitapausten luomista sellaisia vikoja varten, jotka löytyivät skriptaamattoman tai tutkivan testauksen aikana, ja lisää ne regressiotestijoukkoon.
- Etsi viat ennen regressiotestausta. Regressiotestauksen aika on usein rajallinen ja häiriöiden löytäminen regressiotestauksessa voi johtaa aikatauluviivästyksiin. Regressiotestit eivät yleensä löydä suhteellisesti suurta määrää vikoja, pääasiassa siksi, että ne ovat testejä, jotka on jo aikaisemmin suoritettu (esim. saman ohjelmiston aikaisemman version testauksessa), ja viat olisi pitänyt löytää näiden aikaisempien testien aikana. Tämä ei tarkoita, että regressio-testit pitäisi jättää kokonaan pois, vaan tuo esiin vain sen, että uusien vikojen löytämisessä regressiotestaus ei ole niin tehokasta kuin muut testit.

1.8 Lopetusehtojen arviointi ja raportointi

Testausprosessin näkökulmasta testauksen etenemisen seurantaan sisältyy sen varmistaminen, että oikeanlaista tietoa kerätään raportointivaatimusten täyttämiseksi. Tähän kuuluu edistymisen

© International Software Testing Qualifications Board

mittaaminen suhteessa työn valmistumiseen. Kun suunnitteluvaiheessa määritellään päätöskriteerejä, ne voidaan jakaa ”täytyy”- ja ”pitäisi”-kriteereihin. Kriteereissä voidaan esimerkiksi todeta, että ”prioriteetin 1 tai 2 vikoja ei saa olla avoinna”, ja että ”läpäisyprosentin pitäisi olla kaikkien testitapausten osalta 95 %”. Tällöin tilanteen, jossa ”täytyy”-ehto (”ei saa olla”) ei täyty, pitäisi aiheuttaa päätöskriteerien hylkääntyminen, kun taas jos läpäisyprosentti on 93, projekti voi ehkä jatkua seuraavaan vaiheeseen. Päätöskriteerit pitää määritellä selkeästi, jotta niitä voidaan arvioida objektiivisesti.

Testausasiantuntijan vastuulla on tuottaa Testauspäällikölle tietoa, jota tämä käyttää arvioidessaan edistymistä suhteessa päätöskriteerien täyttymiseen, ja varmistaa, että nämä tiedot ovat oikein. Jos esimerkiksi testauksenhallintajärjestelmässä käytetään testitapausten suorituksen tilan ilmaisemiseen arvoja ”hyväksytty”, ”hylätty”, ”hyväksytty ehdollisesti”, on Testausasiantuntijan oltava hyvin selvillä siitä, mitä mikäkin näistä arvoista tarkoittaa, ja sovellettava tätä luokittelua johdonmukaisesti. Tarkoittaako ”hyväksytty ehdollisesti” sitä, että testauksessa löytyi vika, mutta se on sellainen, joka ei vaikuta järjestelmän toiminnallisuuteen? Entä käytettävyysvika, joka saa käyttäjän hämmennyksiin? Jos läpäisyprosentti on ”täytyy”-päätöskriteeri, testitapausten luokittelusta ”hylätyksi” ennemmin kuin ”hyväksytyksi ehdollisesti” tulee kriittinen tekijä. On myös otettava huomioon testitapaukset, jotka on luokiteltu ”hylätyiksi” mutta häiriön syy ei ole vika (esim. testiympäristö oli väärin asennettu). Jos seurattaviin mittareihin tai tilaa kuvaavien arvojen käyttöön liittyy minkäänlaisia epäselvyyksiä, Testausasiantuntijan on selvitettävä asia Testauspäällikön kanssa, jotta tietoja voidaan seurata oikein ja yhdenmukaisesti läpi projektin.

On tavallista, että Testausasiantuntijalta pyydetään tilaraporttia testauskierrosten aikana, ja että häntä pyydetään osallistumaan myös testauksen päätteeksi tehtävän loppuraportin laatimiseen. Tämä voi edellyttää mittaritietojen keräämistä havaintojen- ja testauksenhallintavälineistä samoin kuin koko kattavuuden ja edistymisen arviointia. Testausasiantuntijan pitäisi pystyä käyttämään raportointityökaluja ja tuottamaan Testauspäällikölle tämän pyytämät tiedot, jotta tämä voi koota niistä tarvitsemansa informaation.

1.9 Testauksen päätöstehtävät

Sen jälkeen kun testaus on todettu päättyneeksi, testaustyön avaintuotokset pitäisi tallentaa ja joko siirtää eteenpäin asiaankuuluvalle henkilölle tai arkistoida. Kokonaisuutena näitä tehtäviä kutsutaan testauksen päätöstehtäviksi. On odotettavaa, että Testausasiantuntija on mukana toimittamassa vaihetuotteita niitä tarvitseville henkilöille. Esimerkiksi tunnetuista lykätyistä tai hyväksytyistä vioista pitää viestittää niille, jotka käyttävät järjestelmää ja tukevat sen käyttöä. Testit ja testiympäristöt pitää luovuttaa niille, jotka ovat vastuussa ylläpitotestauksesta. Yksi tuotos voi olla regressiotestijoukko (joko automatisoitu tai manuaalinen). Tiedot testauksen tuotoksista, mukaan luettuna asiaankuuluvat linkit, pitää dokumentoida selkeästi, ja niihin on määriteltävä sopivat käyttöoikeudet.

Testausasiantuntijan voidaan myös odottaa osallistuvan jälkipalaveriin (“mitä tuli opittua”), jossa tärkeät kokemukset (sekä testausprojektin että koko ohjelmistokehityksen elinkaaren ajalta) voidaan kirjata ylös ja laatia suunnitelmat ”hyvien” kokemusten tukemiseksi ja ”huonojen” poistamiseksi tai ainakin hallitsemiseksi. Testausasiantuntija on näissä palavereissa merkittävä tietolähde ja hänen täytyy osallistua niihin erityisesti, mikäli tarkoitus on kerätä prosessikehityksen kannalta kelvollista tietoa. Mikäli ainoastaan Testauspäällikkö osallistuu kokoukseen, Testausasiantuntijan täytyy välittää keskeiset tiedot Testauspäällikölle, jotta projektista esitetään oikeanlainen kuva.

Tulokset, lokit, raportit sekä muut dokumentit ja tuotokset on myös arkistoitava kokoonpanonhallintajärjestelmään. Tämä tehtävä lankeaa usein Testausasiantuntijalle ja se on tärkeä päätöstehtävä erityisesti, mikäli tulevat projektit vaativat näiden tietojen käyttöä.

Vaikka Testauspäällikkö yleensä päättääkin, mitä tietoa arkistoidaan, Testausasiantuntijan pitäisi myös miettiä, mitä tietoa tarvittaisiin, jos projekti käynnistettäisiin uudelleen tulevaisuudessa. Tällaisen

© International Software Testing Qualifications Board

asian miettiminen projektin lopussa voi säästää kuukausien työpanoksen, jos projekti käynnistetään uudelleen myöhemmin tai sen käynnistää toinen tiimi.

© International Software Testing Qualifications Board

2. Testauksen hallinta: Testausasiantuntijan vastuut – 90 min.

Avainsanat
riskianalyysi, riskien hallinta, riskien pienentäminen, riskien tunnistaminen, riskipohjainen testaus, riskitaso, testauksen seuranta, testausstrategia, tuoteriski

Oppimistavoitteet: Testauksenhallinta: Testausasiantuntijan vastuut

2.2 Testauksen edistymisen seuranta ja kontrollointi

TA-2.2.1 (K2) Selittää, minkälaista tietoa täytyy seurata testauksen aikana projektin riittävän seurannan ja hallinnan mahdollistamiseksi.

2.3 Hajautettu, ulkoistettu ja paikallinen ulkoistettu testaus

TA-2.3.1 (K2) Antaa esimerkkejä hyvistä viestintäkäytännöistä noudatettavaksi ympärivuorokautisessa testauksessa.

2.4 Testausasiantuntijan tehtävät riskipohjaisessa testauksessa

TA-2.4.1 (K3) Osallistua riskien tunnistamiseen, suorittaa riskiarviointi ja ehdottaa sopivia riskien pienentämistoimenpiteitä tietyssä projektin tilanteessa.

© International Software Testing Qualifications Board

2.1 Esittely

Vaikka testauksessa onkin monia alueita, joilla Testausasiantuntija toimii ja joista hän tuottaa tietoa Testauspäällikölle, tämä kappale keskittyy testausprosessin niihin tiettyihin alueisiin, joissa Testausasiantuntijalla on keskeinen rooli. On odotettua, että Testauspäällikkö hakee tarvitsemaansa tietoa Testausasiantuntijalta.

2.2 Testauksen edistymisen seuranta ja kontrollointi

Testauksen edistymistä seurataan pääasiassa viidellä eri alueella:

- tuote- (laatu)riskit
- viat
- testit
- kattavuus
- luottamus.

Tuoteriskejä, vikoja, testejä ja kattavuutta voidaan mitata ja niistä voidaan raportoida määrätyllä tavalla projektin tai tehtävän aikana, ja usein tämä on Testausasiantuntijan tehtävä. Vaikka luottamusta voidaankin mitata kyselyiden avulla, siitä raportoidaan yleensä subjektiivisesti. Näitä mittaritietoja tukemaan tarvittavan tiedon kerääminen on osa Testausasiantuntijan päivittäistä työtä. On tärkeää muistaa, että näiden tietojen oikeellisuus on kriittistä, sillä väärät tiedot tuottavat vääristyneitä suuntaustietoja ja voivat johtaa vääriin johtopäätöksiin. Pahimmillaan vääränlaiset tiedot johtavat vääriin hallinnollisiin päätöksiin ja tuottavat vahinkoa testaustiimin uskottavuudelle.

Kun käytetään riskipohjaista testauksen lähestymistapaa, Testausasiantuntijan pitäisi seurata

- mitä riskejä testaus on pienentänyt
- mitkä riskit vaikuttavat pienentymättömiltä.

Riskien pienenemistä seurataan usein työkalulla, joka seuraa myös testien valmistumista (esim. testauksenhallintatyökalut). Tämä edellyttää, että tunnistetut riskit on yhdistetty testattaviin tilanteisiin, jotka on yhdistetty testitapauksiin, jotka pienentävät riskejä, jos testitapaukset suoritetaan ja ne menevät läpi. Tällä tavalla riskien pienentämiseen liittyvät tiedot päivittyvät automaattisesti, kun testitapaukset päivittyvät. Näin voidaan toimia sekä manuaalisessa että automatisoidussa testauksessa.

Vikojen seuraamiseen käytetään yleensä vianhallintatyökalua. Kun viat kirjataan ylös, jokaisesta viasta tallennetaan myös määrättyjä luokittelutietoja. Tätä tietoa käytetään, kun luodaan trendejä ja kaavioita, jotka kertovat testauksen edistymisestä ja ohjelmiston laadusta. Luokittelutiedoista keskustellaan yksityiskohtaisemmin luvussa Vianhallinta. Elinkaari saattaa vaikuttaa vioista laadittavan dokumentaation määrään sekä tapoihin, joilla tiedot tallennetaan.

Testitapausten tilatiedot pitäisi kirjata ylös, kun testausta tehdään. Tämä tapahtuu yleensä testauksenhallintavälineen avulla, mutta se voidaan tarpeen vaatiessa tehdä myös manuaalisesti. Testitapaustiedot voivat sisältää seuraavia asioita:

- testitapauksen laatimisen tila (esim. suunniteltu, katselmoitu)
- testitapauksen suorituksen tila (esim. läpi, hylätty, estetty, ohitettu)
- testitapauksen suoritustiedot (esim. päivämäärä ja aika, testaajan nimi, käytetty aineisto)
- testitapauksen suoritukseen liittyvä materiaali (esim. näytön kuvat, lokitiedot).

Samoin kuin tunnistetut riskit, testitapaukset pitäisi yhdistää vaatimuksiin, joita ne testaavat. Testausasiantuntijan on tärkeä muistaa, että jos testitapaus A on liitetty vaatimukseen A ja se on ainoa siihen vaatimukseen kiinnitetty testitapaus, vaatimus A katsotaan täytetyksi, kun testitapaus A

© International Software Testing Qualifications Board

suoritetaan ja se menee läpi. Tämä voi pitää tai olla pitämättä paikkaansa. Monissa tapauksissa vaatimuksen perusteelliseen testaukseen tarvitaan enemmän testitapauksia, mutta käytännössä aikarajoitteiden vuoksi laaditaan vain osa näistä testeistä. Jos esimerkiksi vaatimuksen toteutuksen perusteelliseen testaukseen tarvittaisiin 20 testitapausta, mutta vain 10 luodaan ja suoritetaan, vaatimuskattavuustiedot näyttävät 100 % kattavuutta, vaikka todellisuudessa saavutettiin vain 50 % kattavuus. Kattavuuden tarkkaa seurantaa samoin kuin itse vaatimusten katselmointitilan seurantaa voidaan käyttää luottamuksen mittauksessa.

Tallennettavan tiedon määrä (ja yksityiskohtaisuuden taso) riippuu monista tekijöistä, mm. ohjelmistokehityksen elinkaarimallista. Esimerkiksi ketterässä projektissa kirjataan tyypillisesti ylös vähemmän tilatietoja tiimin tiiviin yhteistyön ja enemmän kasvokkain tapahtuvan viestinnän vuoksi.

2.3 Hajautettu, ulkoistettu ja paikallinen ulkoistettu testaus

Monissa tapauksissa testauksen parissa ei ole työskentelemässä vain yksittäinen testaustiimi, joka koostuu samaan projektitiimiin kuuluvista työtovereista ja joka työskentelee yhdessä ja samassa paikassa muun projektitiimin kanssa. Jos testausta tehdään useissa paikoissa, sitä voidaan kutsua hajautetuksi testaukseksi. Jos se tapahtuu yhdessä paikassa, sitä voidaan kutsua keskitetyksi. Jos testausta tehdään yhdessä tai useammassa paikassa ja sitä ovat tekemässä henkilöt, jotka eivät ole muiden projektitiimin jäsenten työtovereita eivätkä työskentele samassa paikassa muun projektitiimin kanssa, testausta voidaan kutsua ulkoistetuksi. Jos testausta tekevät henkilöt, jotka työskentelevät samassa paikassa kuin muu projektitiimi mutta eivät ole muiden tiimiläisten työtovereita, testausta voidaan kutsua paikalliseksi ulkoistetuksi testaukseksi.

Kun kyseessä on projekti, jossa osa testaustiimistä on hajallaan useissa eri paikoissa tai jopa useissa eri yrityksissä, Testausasiantuntijan on tärkeää kiinnittää erityistä huomiota tehokkaaseen viestintään ja tiedonsiirtoon. Jotkut organisaatiot käyttävät testauksessa “24 tunnin mallia”, jolloin yhdellä aikavyöhykkeellä oleva testaustiimi siirtää työn toisella aikavyöhykkeellä olevalle tiimille, jotta testaus voi jatkua vuorokauden ympäri. Tämä vaatii erityissuunnittelua työt luovuttavan ja vastaanottavan Testausasiantuntijan kannalta. Hyvä suunnittelu on tärkeää vastuiden ymmärtämisen vuoksi, mutta se on elintärkeää sen varmistamiseksi, että riittävät tiedot ovat saatavilla.

Silloin, kun sanallinen viestintä ei ole mahdollista, kirjallisen viestinnän täytyy riittää. Tämä tarkoittaa, että sähköpostin ja tilaraporttien lisäksi on käytettävä tehokkaasti testauksenhallinta- ja havaintojenhallintatyökaluja. Testauksenhallintaväline voi toimia myös aikataulutusvälineenä ja helppona tapana siirtää työtä ihmisten välillä, mikäli se mahdollistaa testien toimeksiantamisen yksittäisille henkilöille. Tarkasti raportoidut havainnot voidaan tarpeen mukaan ohjata toisten työntekijöiden seurattavaksi. Näiden viestintäkanavien tehokas käyttö on elintärkeää organisaatiolle, joka ei voi luottaa päivittäiseen henkilökohtaiseen kanssakäymiseen.

2.4 Testausasiantuntijan tehtävät riskipohjaisessa testauksessa

2.4.1 Esittely

Testauspäälliköllä on usein kokonaisvastuu riskipohjaisen testausstrategian käyttöönotosta ja hallinnasta. Testauspäällikkö pyytää yleensä Testausasiantuntijaa osallistumaan tehtäviin varmistaakseen, että riskipohjaista lähestymistapaa toteutetaan oikein.

Testausasiantuntijan pitäisi olla aktiivisesti mukana seuraavissa riskipohjaisen testauksen tehtävissä:

- riskien tunnistaminen
- riskien arviointi
- riskien hallinta.

© International Software Testing Qualifications Board

Näitä tehtäviä suoritetaan iteratiivisesti koko projektin elinkaaren ajan esiin nousevien riskien ja muuttuvien prioriteettien hallitsemiseksi sekä riskien tilan säännölliseksi arvioimiseksi ja siitä tiedottamiseksi.

Testausasiantuntijat työskentelevät Testauspäällikön projektille laatiman riskipohjaisen testauksen lähestymistavan puitteissa. Heidän pitäisi tuoda esiin tietämyksensä projektille tyypillisistä liiketoiminta-alueen riskeistä, kuten esimerkiksi turvallisuuteen, liiketoimintaan ja taloudellisiin seikkoihin sekä poliittisiin tekijöihin liittyvät riskit.

2.4.2 Riskien tunnistaminen

Riskien tunnistamisprosessissa löydetään todennäköisimmin suurin mahdollinen määrä merkittäviä riskejä, kun mukaan otetaan laajin mahdollinen joukko eri sidosryhmien edustajia. Koska Testausasiantuntijoilla on usein ainutlaatuista tietoa testattavan järjestelmän tiettyyn liiketoiminta-alueeseen liittyen, he soveltuvat erityisen hyvin haastattelemaan liiketoiminta-alueen asiantuntijoita ja käyttäjiä, tekemään riippumattomia arviointeja, käyttämään riskiluetteloita ja auttamaan niiden käytössä, pitämään riskityöpajoja, vetämään aivoriihiä tulevien ja nykyisten käyttäjien kanssa, laatimaan testauksen tarkistuslistoja sekä hyödyntämään aikaisempia kokemuksia samanlaisista järjestelmistä tai testauksesta. Testausasiantuntijan tulisi työskennellä erityisen tiiviisti käyttäjien ja muiden liiketoiminta-asiantuntijoiden kanssa niiden liiketoimintaan liittyvien riskialueiden määrittämiseksi, joihin testauksen aikana pitäisi paneutua. Testausasiantuntija voi olla myös avuksi erityisesti käyttäjiin ja sidosryhmiin kohdistuvien mahdollisten riskivaikutusten tunnistamisessa.

Projektissa mahdollisesti tunnistettaviin riskeihin kuuluvat esimerkiksi:

- ohjelman toimintaan liittyvät tarkkuusongelmat, esim. väärät laskutoimitukset
- käytettävyysongelmat, esim. riittämättömät näppäimistöoikopolut
- opittavuusasiat, esim. keskeisiin päätöskohtiin liittyvien käyttäjän ohjeiden puuttuminen.

Tiettyjen laatuominaisuuksien testaukseen liittyviä huomioon otettavia asioita käsitellään tämän sertifikaattisisällön luvussa 4.

2.4.3 Riskien arviointi

Siinä missä riskien tunnistamisessa pyritään tunnistamaan niin monta keskeistä riskiä kuin mahdollista, riskien arvioinnissa keskitytään näiden tunnistettujen riskien tutkimiseen. Tämä tarkoittaa jokaisen riskin luokittelua ja joka riskiin liittyvän todennäköisyyden ja vaikutuksen määrittämistä.

Riskitason määrittämiseen kuuluu tyypillisesti riskin toteutumisen todennäköisyyden sekä toteutumisesta seuraavan vaikutuksen arviointi, joka tehdään jokaiselle riskiaiheelle. Riskin toteutumisen todennäköisyydellä tarkoitetaan yleensä todennäköisyyttä, että kyseinen ongelma voi esiintyä testattavassa järjestelmässä ja että se voidaan havaita, kun järjestelmä on tuotantokäytössä. Todennäköisyys on siis toisin sanoen lähtöisin teknisestä riskistä. Tekninen testausasiantuntija auttaa löytämään ja ymmärtämään jokaiseen riskiaiheeseen liittyvät mahdolliset tekniset riskitasot, kun taas Testausasiantuntija auttaa ymmärtämään ongelman toteutumisesta seuraavia mahdollisia liiketoiminnallisia vaikutuksia.

Tapahtuman vaikutus ymmärretään usein tilanteen vakavuutena käyttäjien, asiakkaiden tai muiden sidosryhmien kannalta. Se on siis toisin sanoen lähtöisin liiketoimintariskistä. Testausasiantuntija auttaa tunnistamaan ja arvioimaan jokaisen riskiaiheen mahdollisen liiketoiminta-alueeseen tai käyttäjään kohdistuvan vaikutuksen. Liiketoimintariskeihin vaikuttaviin tekijöihin kuuluvat

- kohteena olevan ominaisuuden käyttötiheys
- liiketoiminnan menetys
- mahdolliset taloudelliset, ekologiset tai sosiaaliset menetykset tai korvausvelvollisuudet
- siviili- tai rikosoikeudelliset rangaistukset

© International Software Testing Qualifications Board

- turvallisuusseikat
- sakot, lisenssimenetykset
- järkevien vaihtoehtoisten toimintatapojen puute
- ominaisuuden näkyvyys
- häiriön näkyvyys, joka johtaa negatiiviseen julkisuuteen ja mahdollisiin imago-vahinkoihin
- asiakkaiden menetys.

Riskeistä saatavilla olevan tiedon perusteella Testausasiantuntijan täytyy määritellä liiketoimintariskien tasot Testauspäällikön laatiman ohjeistuksen mukaisesti. Tasojen luokittelussa voidaan käyttää termejä (esim. matala, keskitaso, korkea) tai numeroita. Mikäli riskiä ei voida objektiivisesti luokitella määritetyllä asteikolla, se ei voi olla todellinen mitattava suure. Todennäköisyyden ja kustannusten/seurausten tarkka mittaaminen on yleensä hyvin vaikeaa, joten riskitaso luokitellaan yleensä laadullisesti.

Laadulliselle arvolle voidaan määrittää numerot, mutta se ei tee siitä todellista mitattavaa suuretta. Esimerkiksi, Testauspäällikkö voi määrätä, että liiketoimintariskit pitää luokitella käyttäen arvoja väliltä 1 – 10, jossa 1 on korkein, ja näin ollen vaikutukseltaan suurin riski. Kun todennäköisyys (teknisen riskin arviointi) ja vaikutus (liiketoimintariskin arviointi) on määritetty, nämä arvot voidaan kertoa keskenään kokonaisriskitason määrittämiseksi jokaiselle riskiaiheelle. Tätä kokonaisriskitasoa käytetään sitten riskien pienentämistoimenpiteiden priorisoinnissa. Jotkut riskipohjaiset testausmallit, kuten PRISMA® [vanVeenendaal12] eivät yhdistä riskiarvoja, mikä mahdollistaa teknisten ja liiketoimintariskien käsittelyn erikseen testauksessa.

2.4.4 Riskien hallinta

Projektin aikana Testausasiantuntijan pitäisi pyrkiä tekemään seuraavat asiat:

- Vähentää tuoteriskejä käyttämällä hyvin suunniteltuja testitapauksia, jotka osoittavat yksiselitteisesti, ovatko testattavat nimikkeet läpäisseet testit vai eivät, sekä osallistumalla ohjelmistotuotteiden, kuten vaatimusten, suunnittelukuvausten ja käyttäjädokumentaation katselmointeihin.
- Toteuttaa testausstrategiassa ja testaussuunnitelmassa kuvattuja sopivia riskienhallintatehtäviä
- Uudelleenarvioida tunnettuja riskejä projektin edetessä kerätyn lisätiedon perusteella ja muokata niiden todennäköisyyttä, vaikutusta tai molempia, mikäli tarpeen.
- Ottaa huomioon testauksen aikana saadun tiedon perusteella tunnistetut uudet riskit.

Kun kyse on tuote-(laatu)riskeistä, testaus on tapa niiden hallitsemiseen. Löytäessään vikoja testaaja pienentää riskejä tuottamalla tietoa vioista ja luomalla tilaisuuksia niiden käsittelyyn ennen julkaisua. Jos testaajat eivät löydä vikoja, testaus vähentää riskejä varmistamalla, että määrätyissä olosuhteissa (eli testatussa tilanteessa) järjestelmä toimii oikein. Testausasiantuntija auttaa määrittämään riskienhallintaan liittyviä vaihtoehtoja tutkimalla tilaisuuksia tarkan testiaineiston keräämiseen, luomalla ja testaamalla realistisia käyttäjätarinoita sekä suorittamalla tai ohjaamalla käytettävyystutkimuksia.

2.4.4.1 Testien priorisointi

Riskitasoa käytetään myös testien priorisointiin. Testausasiantuntija voi päätellä, että kirjanpitojärjestelmän tapahtumien tarkkuuden käsittelyyn liittyy korkean tason riski. Tämän seurauksena testaaja voi riskin pienentämiseksi työskennellä muiden liiketoiminta-alueen asiantuntijoiden kanssa ja kerätä kattavan joukon esimerkkiaineistoa, joka voidaan käsitellä ja todentaa oikeellisuuden varmistamiseksi. Yhtä lailla Testausasiantuntija voi päätellä, että käytettävyysseikat muodostavat merkittävän riskin uudelle tuotteelle. Sen sijaan, että Testaus-asiantuntija odottaisi hyväksymistestauksesta mahdollisesti löytyviä ongelmia, hän voi priorisoida käytettävyystestauksen suoritettavaksi aikaisin integrointitestaustasolla, mikä auttaa tunnistamaan ja ratkaisemaan käytettävyysongelmia testauksen aikaisessa vaiheessa. Tätä priorisointia täytyy harkita

© International Software Testing Qualifications Board

suunnitteluvaiheessa niin aikaisin kuin mahdollista, jotta tarvittava testaus sopii aikatauluun tarvittavaan aikaan.

Joissain tapauksissa kaikki korkeimman tason riskien testit suoritetaan ennen alemman tason riskien testejä ja testit suoritetaan tiukasti riskijärjestyksessä (kutsutaan usein “syvyysjärjestykseksi”); toisissa tapauksissa käytetään näytteenottolähestymistapaa, jolloin kaikkien tunnistettujen riskien joukosta valitaan joukko testejä käyttämällä riskiä painottamassa valintaa varmistamalla samalla kuitenkin, että jokainen riski tulee katettua ainakin kerran (kutsutaan usein ”leveysjärjestykseksi”).

Käytetäänpä riskipohjaisessa testauksessa ensimmäisenä sitten syvyys- tai leveysjärjestystä, on mahdollista, että testaukseen varattu aika loppuu ennen kuin kaikki testit on suoritettu. Riskipohjainen testaus luo testaajalle mahdollisuuden raportoida johdolle käyttämällä kyseisellä hetkellä jäljellä olevaa riskitasoa, ja se antaa johdolle mahdollisuuden päättää, laajennetaanko testausta vai siirretäänkö jäljellä olevat riskit käyttäjien, asiakkaan, help deskin/teknisen tuen ja/tai operaattoreiden hoidettavaksi.

2.4.4.2 Testauksen sopeuttaminen seuraavia testikierroksia varten

Riskiarviointi ei ole vain kerran testauksen alussa suoritettava tehtävä: se on jatkuva prosessi. Jokaisen suunnitellun testikierroksen osalta pitäisi suorittaa riskianalyysi, jotta voidaan ottaa huomioon mm. seuraavat seikat:

- uudet tai merkittävästi muuttuneet tuoteriskit
- testauksen aikana löydetyt epävakaat tai vioille alttiit alueet
- korjatuista vioista aiheutuneet riskit
- tyypilliset testauksen aikana löydetyt viat
- alitestatut alueet (alhainen testikattavuus).

Jos testaukselle annetaan lisäaikaa, voi olla mahdollista laajentaa riskikattavuutta matalan riskin alueille.

© International Software Testing Qualifications Board

3. Testaustekniikat – 825 min

Avainsanat

arvoalueanalyysi, ekvivalenssiositus, kokemusperusteinen tekniikka, kombinatorinen testaus, käyttäjätarinatestaus, käyttötapaustestaus, luokittelupuumenetelmä, määrittelypohjainen tekniikka, ortogonaalinen matriisi, ortogonaaliseen matriisiin perustuva testaus, päätöstaulutestaus, raja-arvoanalyysi (BVA), syy-seuraus-kaaviotestaus, syötteiden pareittainen testaus, tarkistuslistoihin pohjautuva testaus, testausohje, tilasiirtymätestaus, tutkiva testaus, vaatimuksiin perustuva testaus, vikaluokitusjärjestelmä, vikaperusteiset tekniikat, virheenarvaus

Oppimistavoitteet: Testaustekniikat

3.2 Määrittelypohjaiset tekniikat

TA-3.2.1 (K2) Selittää, kuinka syy-seuraus-kaavioita käytetään

TA-3.2.2 (K3) Kirjoittaa annetun kuvauksen pohjalta testitapauksia käyttämällä ekvivalenssiositus-testisuunnittelutekniikkaa määrätyn kattavuustason saavuttamiseksi.

TA-3.2.3 (K3) Kirjoittaa annetun kuvauksen pohjalta testitapauksia käyttämällä raja-arvoanalyysi-testisuunnittelutekniikkaa määrätyn kattavuustason saavuttamiseksi.

TA-3.2.4 (K3) Kirjoittaa annetun kuvauksen pohjalta testitapauksia käyttämällä päätöstaulutestaus-testisuunnittelutekniikkaa määrätyn kattavuustason saavuttamiseksi.

TA-3.2.5 (K3) Kirjoittaa annetun kuvauksen pohjalta testitapauksia käyttämällä tilasiirtymätestaus-testisuunnittelutekniikkaa määrätyn kattavuustason saavuttamiseksi.

TA-3.2.6 (K3) Kirjoittaa annetun kuvauksen pohjalta testitapauksia käyttämällä pareittainen testaus-testisuunnittelutekniikkaa määrätyn kattavuustason saavuttamiseksi.

TA-3.2.7 (K3) Kirjoittaa annetun kuvauksen pohjalta testitapauksia käyttämällä luokittelupuu-testisuunnittelutekniikkaa määrätyn kattavuustason saavuttamiseksi.

TA-3.2.8 (K3) Kirjoittaa annetun kuvauksen pohjalta testitapauksia käyttämällä käyttötapaustestaus-testisuunnittelutekniikkaa määrätyn kattavuustason saavuttamiseksi.

TA-3.2.9 (K2) Selittää, kuinka käyttäjätarinoita käytetään ketterässä projektissa ohjaamaan testausta.

TA-3.2.10 (K3) Kirjoittaa annetun kuvauksen pohjalta testitapauksia käyttämällä arvoalueanalyysi-testisuunnittelutekniikkaa määrätyn kattavuustason saavuttamiseksi.

TA-3.2.11 (K4) Analysoida järjestelmää tai sen vaatimuskuvauksia todennäköisten vikatyyppien löytämiseksi sekä sopivien määrittelypohjaisten tekniikoiden valitsemiseksi.

3.3 Vikaperusteiset tekniikat

TA-3.3.1 (K2) Kuvata vikaperusteisten tekniikoiden soveltaminen ja erottaa niiden käyttö määrittelypohjaisista tekniikoista.

TA-3.3.2 (K4) Analysoida annettua vikaluokittelujärjestelmää tiettyyn käyttötilanteeseen soveltuvuuden kannalta käyttämällä perustana hyvän luokittelujärjestelmän kriteereitä.

3.4 Kokemusperusteiset tekniikat

TA-3.4.1 (K2) Selittää kokemuspohjaisten tekniikoiden pääperiaatteet sekä niiden hyödyt ja haitat verrattuna määrittelypohjaisiin ja vikaperusteisiin tekniikoihin.

TA-3.4.2 (K3) Määrittää tietyssä tilanteessa käytettävät tutkivan testauksen testit ja selittää, kuinka tulokset voidaan raportoida.

TA-3.4.3 (K4) Määrittää, mitä määrittelypohjaisia, vikaperusteisia tai kokemusperusteisia tekniikoita pitäisi käyttää määrättyjen tavoitteiden saavuttamiseksi tietyssä projektin tilanteessa.

© International Software Testing Qualifications Board

3.1 Esittely

Tässä luvussa käsitellyt testisuunnittelutekniikat on jaettu seuraaviin luokkiin:

- Määrittelypohjaiset (tai käyttäytymiseen pohjautuvat tai mustalaatikkotekniikat)
- Vikaperusteiset
- Kokemusperusteiset

Nämä tekniikat täydentävät toisiaan ja niitä voidaan käyttää sopivaksi katsotulla tavalla minkä tahansa testaustehtävän aikana huolimatta siitä, millä testaustasolla testausta ollaan tekemässä.

On huomattava, että näihin kaikkiin kolmeen luokkaan kuuluvia tekniikoita voidaan käyttää sekä toiminnallisten että ei-toiminnallisten laatuominaisuuksien testaamiseen. Ei-toiminnallisten ominaisuuksien testausta käsitellään seuraavassa luvussa.

Näissä kappaleissa käsitellyt testaustekniikat voivat keskittyä pääasiassa optimaalisen testiaineiston määrittämiseen (esim. ekvivalenssiositus) tai testijaksojen luomiseen (esim. tilamallit). On tavallista, että testaustekniikoita yhdistellään kokonaisten testitapausten luomiseksi.

3.2 Määrittelypohjaiset tekniikat

Määrittelypohjaisia tekniikoita käytetään testattavien tilanteiden testauksessa tarvittavien testitapausten määrittämiseen komponentin tai järjestelmän pohjamateriaalin analyysin perusteella ilman, että kiinnitetään huomiota komponentin tai järjestelmän sisäiseen rakenteeseen.

Määrittelypohjaisten tekniikoiden tyypillisiin piirteisiin kuuluvat seuraavat:

- Malleja, esim. tilasiirtymäkaaviot ja päätöstaulut, luodaan käytettävän tekniikan mukaisesti testisuunnittelun aikana
- Testattavat tilanteet johdetaan järjestelmällisesti näistä malleista.

Jotkut tekniikat tuottavat myös kattavuuskriteereitä, joita voidaan käyttää testisuunnitteluun ja testien suoritukseen liittyvien tehtävien mittaamiseen. Kattavuuskriteerien täydellinen täyttäminen ei tarkoita, että koko testijoukko olisi täydellinen, vaan se kertoo enemmänkin, että kyseinen malli ei enää tuo esiin lisätestejä kattavuuden kasvattamiseksi kyseistä tekniikkaa käyttämällä.

Määrittelypohjaiset tekniikat perustuvat yleensä järjestelmän vaatimuskuvauksiin. Koska vaatimuskuvausten pitäisi määritellä, kuinka järjestelmän tulisi käyttäytyä erityisesti toiminnallisuuden osalta, testien luominen vaatimusten perusteella on usein osa järjestelmän käyttäytymisen testausta. Joissakin tapauksissa ei ehkä ole dokumentoituja vaatimuksia vaan on olemassa epäsuoria vaatimuksia, kuten esimerkiksi vanhan järjestelmän toiminnallisuuden korvaaminen.

Määrittelypohjaisia tekniikoita on paljon. Näiden tekniikoiden kohteena ovat erityyppiset ohjelmistot ja käyttötilanteet. Seuraavissa kappaleissa kuvataan kunkin tekniikan käyttötilanteet, rajoitteita ja vaikeuksia, joita Testausasiantuntija saattaa kohdata, tapa, jolla testikattavuus mitataan, sekä vikatyypit, joihin kyseinen tekniikka kohdistuu.

3.2.1 Ekvivalenssiositus

Ekvivalenssiositusta käytetään vähentämään syötteiden, tulosten, ohjelman sisäisten ja aikariippuvaisten arvojen tehokkaaseen testaukseen tarvittavien testitapausten määrää. Ositusta käytetään, kun luodaan ekvivalenssiluokkia (kutsutaan myös ekvivalenssipartitioiksi), jotka muodostuvat arvojoukoista, jonka kaikki arvot käsitellään samalla tavalla. Oletuksena on, että valitsemalla joukosta yksi arvojen edustaja katetaan kaikki saman joukon arvot.

Sovellettavuus

© International Software Testing Qualifications Board

Tätä tekniikkaa voidaan käyttää kaikilla testaustasoilla, ja se on käyttökelpoinen silloin, kun oletetaan, että kaikkia testattavan arvojoukon jäseniä käsitellään samalla tavalla, ja kun sovelluksen käyttämät arvojoukot eivät ole keskenään tekemisissä toistensa kanssa. Arvojoukot jaetaan kelvollisiin ja epäkelpoihin (joukkoihin, jotka sisältävät arvoja, joiden pitäisi testattavan ohjelman näkökulmasta olla epäkelpoja). Tämä tekniikka on parhaimmillaan silloin, kun sitä käytetään yhdessä raja-arvoanalyysin kanssa, joka laajentaa testattavat arvot sisältämään myös niitä, jotka ovat arvojoukkojen rajoilla. Tätä tekniikkaa käytetään yleisesti uuden koonnin tai julkaisun aloitustestauksessa, sillä se osoittaa nopeasti, toimiiko perustoiminnallisuus.

Rajoitukset/vaikeudet
Jos oletus on väärä ja arvojoukon arvoja ei käsitellä täsmälleen samalla tavalla, vikoja saattaa jäädä huomaamatta tällä tekniikalla. On myös tärkeää valita luokat huolellisesti. On esimerkiksi parempi testata sekä positiivisia että negatiivisia lukuja hyväksyvä syöttökenttä kahtena eri luokkana, yksi positiivisille luvuille ja toinen negatiivisille luvuille, koska niitä todennäköisesti käsitellään eri tavalla. Riippuen siitä, onko nolla sallittu arvo vai ei, siitä saattaa muodostua myös oma luokka. Testausasiantuntijan on tärkeä ymmärtää taustalla oleva prosessointi, jotta hän pystyy määrittelemään parhaimmat luokat arvoille.

Kattavuus
Kattavuus määritellään laskemalla testattujen luokkien määrä ja jakamalla se kaikkien tunnistettujen luokkien määrällä. Useiden arvojen käyttäminen yksittäisen luokan testauksessa ei kasvata kattavuusprosenttia.

Vikatyypit
Tämä tekniikka löytää toiminnallisia vikoja, jotka liittyvät erilaisten tietoarvojen käsittelyyn.

3.2.2 Raja-arvoanalyysi

Raja-arvoanalyysiä käytetään, kun testataan arvoja, jotka ovat järjestettyjen ekvivalenssiluokkien rajoilla. Raja-arvoanalyysiin on kaksi lähestymistapaa: kaksiarvo- ja kolmiarvotestaus. Kaksiarvotestauksessa käytetään raja-arvoa (rajalla olevaa arvoa) sekä arvoa, joka on juuri (pienimmän mahdollisen askeleen) rajan ulkopuolella. Jos esimerkiksi luokkaan kuuluvat arvot yhdestä kymmeneen, ja askel on puoli yksikköä, ylärajalla testattavat kaksi arvoa ovat 10 ja 10,5. Alarajalla testattavat arvot ovat 1 ja 0,5. Rajat määritellään kyseisen ekvivalenssiluokan minimi- ja maksimiarvojen perusteella.

Kolmiarvotestauksessa käytetään arvoja, jotka ovat rajalla sekä sitä ennen ja sen jälkeen. Edellisessä esimerkissä ylärajan testeihin kuuluisivat 9,5, 10 ja 10,5. Alarajalla testattavat arvot olisivat 1,5, 1 ja 0,5. Päätöksen kaksi- tai kolmiarvoanalyysin käyttämisestä pitäisi perustua testattavaan kohteeseen liittyviin riskeihin, jolloin kolmiarvolähestymistapaa käytetään kohteisiin, joissa riskit ovat suuremmat.

Sovellettavuus
Tätä tekniikkaa voidaan käyttää kaikilla testaustasoilla, ja se sopii käytettäväksi silloin, kun on olemassa järjestettyjä ekvivalenssiluokkia. Järjestystä vaaditaan, koska puhutaan arvoista, jotka ovat rajalla ja sen ulkopuolella. Esimerkiksi numeroalue on järjestetty luokka. Luokka, joka sisältää suorakaiteen muotoisia esineitä, ei ole järjestetty luokka, eikä sillä ole raja-arvoja. Numeroalueiden lisäksi raja-arvoanalyysiä voidaan soveltaa seuraaviin kohteisiin:
- ei-numeeristen muuttujien numeeriset attribuutit (esim. pituus)
- silmukat, myös käyttötapausten sisältämät
- tallennetut tietorakenteet
- fyysiset kohteet (mukaan luettuna muisti)
- aikariippuvat toiminnot.

Rajoitukset/vaikeudet

© International Software Testing Qualifications Board

Koska tämän tekniikan tarkkuus riippuu ekvivalenssiluokkien oikeasta tunnistamisesta, siihen kohdistuvat samat rajoitukset ja vaikeudet. Testausasiantuntijan pitää myös huomata kelvollisten ja epäkelpojen arvojen askeleet, jotta hän pystyy määrittämään testattavat arvot oikein. Vain järjestettyjä luokkia voidaan käyttää raja-arvoanalyysiin, mutta sitä voidaan tehdä muillakin kuin kelvollisilla syötteillä. Kun esimerkiksi testataan, paljonko soluja laskentataulukossa pystytään käsittelemään, on olemassa luokka, joka sisältää arvot suurimpaan sallittuun solujen määrään maksimiarvo (raja) mukaan luettuna, sekä toinen luokka, joka alkaa arvosta, joka on yksi yli sallitun maksimiarvon (yli rajan).

Kattavuus
Kattavuus määritetään laskemalla testattujen rajaehtojen määrä ja jakamalla se tunnistettujen rajaehtojen määrällä (käyttämällä joko kaksi- tai kolmiarvomenetelmää). Tämä tuottaa tulokseksi raja-arvotestauksen kattavuusprosentin.

Vikatyypit
Raja-arvoanalyysin avulla löydetään luotettavasti rajoja, jotka on sijoitettu väärin tai jotka ovat jääneet huomaamatta, ja sen avulla voidaan löytää ylimääräisiä (turhia) rajoja. Tällä tekniikalla löydetään vikoja, jotka liittyvät raja-arvojen käsittelyyn, erityisesti pienempi-kuin- ja suurempi-kuin-logiikkaan liittyviä virheitä (eli rajasiirtymiä). Sitä voidaan käyttää myös löytämään ei-toiminnallisia vikoja, esimerkiksi liittyen kuormituksen rajoihin (esim. järjestelmän pitää pystyä käsittelemään 10.000 yhtäaikaista käyttäjää).

3.2.3 Päätöstaulut

Päätöstauluja käytetään ehtoyhdistelmien välisen toiminnan testaamiseen. Päätöstaulut tarjoavat selkeän menetelmän, jolla voidaan todentaa, että kaikki oleelliset ehtoyhdistelmät on testattu ja että testattava ohjelmisto käsittelee kaikki mahdolliset yhdistelmät. Päätöstaulutestauksen tavoitteena on varmistaa, että kaikki ehtojen, suhteiden ja rajoitteiden yhdistelmät tulevat testatuiksi. Päätöstauluista saattaa tulla hyvin suuria, kun yritetään testata kaikki mahdolliset yhdistelmät. Menetelmää, jolla harkitusti vähennetään yhdistelmien määrää kaikista mahdollisista niihin, jotka ovat “kiinnostavia”, kutsutaan supistetuksi päätöstaulutestaukseksi. Kun tätä tekniikkaa käytetään, yhdistelmien määrää vähennetään poistamalla sellaisia ehtoyhdistelmiä, jotka eivät vaikuta lopputulokseen, niin että jäljelle jäävät yhdistelmät tuottavat erilaisia lopputuloksia. Testeistä poistetaan tarpeettomat (tuplat) tai sellaiset, joiden ehtoyhdistelmät eivät ole mahdollisia. Päätös siitä, käytetäänkö täyttä vai supistettua päätöstaulua, pohjautuu yleensä riskeihin. [Copeland03].

Sovellettavuus
Tätä tekniikkaa käytetään tyypillisesti integrointi-, järjestelmä- ja hyväksymistestaustasoilla. Sitä voidaan myös ohjelmakoodista riippuen käyttää yksikkötestaustasolla, mikäli komponentti sisältää useiden päätösten yhdistelmiä. Tämä tekniikka on erityisen hyödyllinen silloin, kun vaatimukset on esitetty joko vuokaavion muodossa tai liiketoimintasäännöt sisältävänä taulukkona. Päätöstaulut ovat myös vaatimusmäärittelytekniikka ja jotkut vaatimusmäärittelyt voivatkin olla jo valmiiksi tässä muodossa. Silloinkin, kun vaatimuksia ei ole esitetty taulukko- tai vuokaaviomuodossa, ehto-yhdistelmät löytyvät yleensä kuvaustekstistä. Kun päätöstaulua suunnitellaan, on tärkeää miettiä sekä kuvattuja ehtoyhdistelmiä että niitä, joita ei ole yksiselitteisesti kuvattu mutta jotka ovat olemassa. Kelvollisen päätöstaulun suunnittelemiseksi testaajan täytyy pystyä päättelemään kaikki odotetut lopputulokset kaikille ehtoyhdistelmille joko määrityksistä tai testioraakkelin avulla. Päätöstaulua voidaan käyttää hyvänä testisuunnitteluvälineenä vain silloin, kun kaikki toisiinsa vaikuttavat ehdot on otettu huomioon.

Rajoitukset/vaikeudet
Kaikkien toisiinsa vaikuttavien yhdistelmien löytäminen voi olla haastavaa erityisesti silloin, jos vaatimukset eivät ole selkeästi määriteltyjä tai niitä ei ole. Ei ole poikkeuksellista, että määritellään joukko ehtoja ja todetaan, että odotettu lopputulos on tuntematon.

© International Software Testing Qualifications Board

Kattavuus
Päätöstaulun vähimmäiskattavuus tarkoittaa, että jokaista saraketta kohtaan on yksi testitapaus. Tämä edellyttää, että mukana ei ole yhdistelmäehtoja ja että kaikki mahdolliset ehtoyhdistelmät on kirjattu sarakkeisiin. Kun määritellään testejä päätöstaulun pohjalta, on myös tärkeää miettiä mahdollisia rajaehtoja, joita pitäisi testata. Nämä rajaehdot saattavat johtaa siihen, että ohjelmiston riittävään testaukseen tarvittavien testitapausten määrä kasvaa. Raja-arvoanalyysi ja ekvivalenssiositus täydentävät päätöstaulutekniikkaa.

Vikatyypit
Tyypillisiin vikoihin kuuluu tiettyjen ehtoyhdistelmien vääränlainen käsittely, joka johtaa odottamattomiin lopputuloksiin. Määrittelykuvauksista voi löytyä vikoja päätöstaulun luomisen aikana. Tyypillisimmät vikatyypit ovat puutteita (ei ole kerrottu, mitä tietyssä tilanteessa pitäisi todella tapahtua) ja ristiriitoja. Testaus voi löytää myös ongelmia liittyen ehtoyhdistelmiin, joita ei käsitellä joko lainkaan tai kunnolla.

3.2.4 Syy-seuraus-kaaviotestaus

Syy-seuraus-kaavioita voidaan luoda mistä tahansa materiaalista, joka kuvaa ohjelman toiminnallisen logiikan (eli säännöt), kuten käyttäjätarinat tai vuokaaviot. Ne voivat olla hyödyllisiä, jos halutaan saada graafinen yleisnäkymä ohjelman loogisesta rakenteesta, ja niitä käytetään tyypillisesti perustana päätöstauluja luotaessa. Päätösten kirjaaminen syy-seuraus-kaavioihin ja/tai päätöstauluihin mahdollistaa ohjelmalogiikan kattamisen järjestelmällisesti.

Sovellettavuus
Syy-seuraus-kaaviot sopivat samoihin tilanteisiin ja samoille testaustasoille kuin päätöstaulut. Erityisesti syy-seuraus-kaaviot tuovat esiin

- ehtoyhdistelmiä, jotka aiheuttavat seurauksen (syy-yhteys)
- ehtoyhdistelmiä, jotka poissulkevat tuloksen (ei – “not”)
- useiden ehtojen ryhmiä, joissa kaikkien ehtojen täytyy saada arvo ”tosi” tuloksen aikaansaamiseksi (ja – ”and”)
- vaihtoehtoisia ehtoja, jotka voivat saada arvon ”tosi” aiheuttaakseen tietyn tuloksen (tai – ”or”).

Nämä suhteet on helpompi nähdä syy-seuraus-kaaviosta kuin sanallisesta kuvauksesta.

Rajoitukset/vaikeudet
Syy-seuraus-kaaviotestauksen oppiminen vaatii enemmän aikaa ja työtä verrattuna muihin testisuunnittelutekniikoihin. Se vaatii myös työvälinetukea. Syy-seuraus-kaavioissa käytetään erityistä notaatiota, joka sekä kaavion tekijän että lukijan on ymmärrettävä.

Kattavuus
Vähimmäiskattavuuden saavuttamiseksi on testattava jokainen mahdollinen syy-seuraus-polku, mukaan luettuna yhdistelmäehdot. Syy-seuraus-kaaviot tarjoavat keinon määritellä rajoitteita sekä aineiston että logiikkavuon suhteen.

Vikatyypit
Nämä kaaviot löytävät samantyyppisiä yhdistelmiin liittyviä vikoja kuin päätöstaulut. Lisäksi kaavioiden laatiminen auttaa määrittämään testauksen pohjamateriaalilta vaadittavan tarkkuustason, mikä auttaa parantamaan pohjamateriaalin tarkkuutta ja laatua sekä auttaa testaajia tunnistamaan puuttuvia vaatimuksia.

© International Software Testing Qualifications Board

3.2.5 Tilasiirtymätestaus

Tilasiirtymätestausta käytetään, kun testataan ohjelmiston kykyä siirtyä määrättyyn tilaan ja poistua siitä kelvollisten ja epäkelpojen siirtymien kautta. Tapahtumat saavat ohjelmiston siirtymään tilasta toiseen ja suorittamaan toimintoja. Tapahtumien valintaa voivat ohjata ehdot (joita joskus kutsutaan vahtiehdoiksi (guard condition) tai siirtymävahdeiksi (transition guard)), jotka vaikuttavat siihen, mikä siirtymäpolku suoritetaan. Esimerkiksi sisäänkirjautuminen kelvollisella käyttäjätunnus/salasana-yhdistelmällä johtaa eri siirtymään kuin sisäänkirjautuminen virheellisellä salasanalla.

Tilasiirtymät kuvataan joko tilasiirtymäkaaviossa, jossa näkyvät kaikki tilojen väliset kelvolliset siirtymät graafisessa muodossa, tai tilataulussa, jossa esitetään kaikki mahdolliset siirtymät, sekä kelvolliset että epäkelvot.

Sovellettavuus
Tilasiirtymätestaus soveltuu kaikkien sellaisten ohjelmistojen testaukseen, joihin liittyy määriteltyjä tiloja sekä tapahtumia, jotka aiheuttavat siirtymiä tilojen välillä (esim. vaihtuvat näytöt). Tilasiirtymätestausta voidaan käyttää kaikilla testaustasoilla. Sulautetut järjestelmät, verkko-ohjelmistot ja tapahtumapohjaiset ohjelmistot ovat hyviä ehdokkaita tämän tyyppiseen testaukseen. Valvontajärjestelmät, esim. liikennevalovalvonta, ovat myös hyviä ehdokkaita tällaiseen testaukseen.

Rajoitukset/vaikeudet
Tilataulun tai tilakaavion määrittämisessä kaikkein vaikeinta on usein tilojen määrittäminen. Mikäli ohjelmistossa on käyttöliittymä, käyttäjälle näytettäviä eri näyttöjä käytetään usein tilojen määrittämiseen. Sulautettujen järjestelmien kohdalla tilat voivat olla riippuvaisia laitteiston kohtaamista tiloista.

Itse tilojen lisäksi tilasiirtymätestauksen perusyksikkö on yksittäinen siirtymä, jota kutsutaan myös 0-switchiksi. Testaamalla pelkästään kaikki siirtymät voidaan löytää jonkinlaisia tilasiirtymävikoja, mutta enemmän vikoja voidaan löytää testaamalla siirtymien jaksoja. Kahden peräkkäisen siirtymän jaksoa kutsutaan 1-switchiksi, kolmen peräkkäisen siirtymän jaksoa 2-switchiksi, ja niin edelleen. (Näistä siirtymistä käytetään joskus myös nimitystä N - 1 switch, jossa N on luku, joka kertoo läpi kuljettavien siirtymien määrän. Esimerkiksi yksittäinen siirtymä (0-switch), olisi 1 – 1 switch. [Bath08])

Kattavuus
Kuten muidenkin testaustekniikoiden kohdalla, kattavuuteen liittyy tasohierarkia. Pienimmän hyväksyttävän kattavuuden saavuttamiseksi on käytävä läpi jokainen tila ja jokainen siirtymä. 100 % siirtymäkattavuus (käytetään myös nimitystä 100 % 0-switch-kattavuus tai 100 % looginen haarakattavuus) takaa, että jokainen tila ja siirtymä on käyty läpi, mikäli järjestelmän suunnittelussa tai tilasiirtymämallissa (kaavio tai taulukko) ei ole virheitä. Tilojen ja siirtymien välisistä suhteista riippuen voi olla tarpeen käydä jokin siirtymä läpi useampaan kertaan, jotta jokin toinen siirtymä päästään käymään läpi yhden kerran.

Termi “n-switch-kattavuus” liittyy katettujen siirtymien määrään. Esimerkiksi 100 % 1-switch-kattavuuden saavuttaminen edellyttää, että jokainen kelvollinen kahden peräkkäisen siirtymän jakso on testattu ainakin kerran. Tämä testaus voi tuoda esiin häiriöitä, joita 100 % 0-switch-kattavuudella ei huomattaisi.

“Meno-paluu-kattavuus” liittyy tilanteeseen, jossa siirtymäjaksot muodostavat silmukoita. 100 % meno-paluu-kattavuus saavutetaan, kun kaikki silmukat mistä tahansa tilasta takaisin samaan tilaan on testattu. Tämä testaus pitää suorittaa kaikille silmukoihin kuuluville tiloille.

Kaikkien näiden lähestymistapojen osalta vieläkin korkeampi kattavuus pyrkii ottamaan mukaan kaikki epäkelvot siirtymät. Kattavuusvaatimuksissa ja tilasiirtymäjoukkojen kattamissuunnitelmissa on määriteltävä, otetaanko epäkelvot siirtymät mukaan.

© International Software Testing Qualifications Board

Vikatyypit
Tyypillisiin vikoihin kuuluvat senhetkisen tilan vääränlainen käsittely, joka on seurausta edellisessä tilassa tapahtuneesta käsittelystä, vääränlaiset tai ei-tuetut siirtymät, tilat, joista ei pääse pois, sekä tarve tiloille tai siirtymille, joita ei ole. Määrittelykuvauksista voi löytyä vikoja tilakonemallin luomisen aikana. Tyypillisimmät vikatyypit ovat puutteita (ei ole kerrottu, mitä tietyssä tilanteessa pitäisi todella tapahtua) ja ristiriitoja.

3.2.6 Kombinatoriset testaustekniikat

Kombinatorista testausta käytetään, kun testataan ohjelmistoa, joka käyttää useita parametreja, joista jokainen voi saada useita arvoja, mikä johtaa useampiin yhdistelmiin kuin on järkevää tai mahdollista testata käytettävissä olevan ajan puitteissa. Parametrien täytyy olla riippumattomia ja yhteensopivia siinä mielessä, että minkä tahansa tekijän mikä tahansa vaihtoehto voidaan yhdistää minkä tahansa tekijän minkä tahansa vaihtoehdon kanssa. Luokittelupuut mahdollistavat joidenkin yhdistelmien poissulkemisen, jos jotkut vaihtoehdot ovat yhteensopimattomia. Tämä ei kuitenkaan tarkoita, että yhdistetyt tekijät eivät vaikuttaisi toisiinsa; ne voivat hyvinkin vaikuttaa, mutta niiden pitää tehdä se hyväksyttävillä tavoilla.

Kombinatorinen testaus tarjoaa keinon tunnistaa näistä yhdistelmistä sopiva osajoukko, jolla voidaan saavuttaa ennalta määritetty kattavuustaso. Testausasiantuntija voi valita yhdistelmiin mukaan otettavien tekijöiden määrän, kuten yksittäiset tekijät, parit, kolmikot tai enemmän [Copeland03]. Tarjolla on lukuisia työkaluja, jotka auttavat Testausasiantuntijaa tässä tehtävässä (ks. esimerkiksi www.pairwise.org). Nämä työkalut edellyttävät, että parametrit ja niiden arvot esitetään luettelona (syötteiden pareittainen testaus ja ortogonaaliseen matriisin perustuva testaus) tai graafisessa muodossa (luokittelupuut) [Grochtmann94]. Syötteiden pareittainen testaus on menetelmä, jolla testataan muuttujapareja eri yhdistelmissä. Ortogonaaliset matriisit ovat ennalta määriteltyjä matemaattisesti tarkkoja taulukkoja, joiden avulla Testausasiantuntija voi korvata taulukossa käytetyt muuttujat testattavilla kohteilla ja saada aikaan joukon yhdistelmiä, jotka saavuttavat tietyn kattavuustason, kun ne testataan [Koomen06]. Luokittelupuutyökalut auttavat Testausasiantuntijaa määrittämään testattavien yhdistelmien koon (eli käytetäänkö kahden arvon yhdistelmiä, kolmen arvon yhdistelmiä jne.).

Sovellettavuus
Ongelma, joka syntyy liian monesta parametriarvojen yhdistelmästä, näkyy ainakin kahdessa testaukseen liittyvässä tilanteessa. Jotkut testitapaukset sisältävät useita parametreja, joilla on useita mahdollisia arvoja, kuten esimerkiksi näyttö, jossa on useita syöttökenttiä. Tässä tapauksessa parametriarvojen yhdistelmät muodostavat testitapausten syöteaineiston. Lisäksi joitakin järjestelmiä voidaan konfiguroida monen eri ulottuvuuden suhteen, mikä johtaa mahdollisesti laajaan kokoonpanojoukkoon. Molemmissa näissä tilanteissa kombinatorista testausta voidaan käyttää tunnistamaan yhdistelmien osajoukko, joka on kooltaan järkevä.

Kun kyseessä ovat parametrit, jotka voivat saada useita arvoja, voidaan ensiksi käyttää ekvivalenssiositusta tai jotain muuta valintamenetelmää jokaiseen parametriin erikseen, ja näin vähentää joka parametrin arvoja, ennen kuin käytetään kombinatorista testausta pienentämään syntyvien yhdistelmien joukkoa.

Näitä tekniikoita käytetään yleensä integrointi-, järjestelmä- ja järjestelmäintegrointitestaustasoilla.

Rajoitukset/vaikeudet
Suurin rajoitus näiden tekniikoiden kohdalla on oletus, että muutaman testin tulokset edustavat kaikkia testejä ja että nuo muutamat testit edustavat odotettua käyttöä. Jos joidenkin muuttujien välillä tapahtuu odottamatonta yhteistoimintaa, se saattaa jäädä huomaamatta tämän tyyppisessä

© International Software Testing Qualifications Board

testauksessa, jos tuota kyseistä yhdistelmää ei testata. Nämä tekniikat saattavat olla vaikeasti selitettävissä ei-teknisille kuulijoille, sillä he eivät ehkä ymmärrä testien vähentämisen logiikkaa.

Parametrien ja niiden käyttämien arvojen tunnistaminen on joskus vaikeaa. Pienimmän yhdistelmäjoukon löytäminen tietyn kattavuustason saavuttamiseksi on vaikeaa käsityönä. Työkaluja käytetään yleensä löytämään minimijoukko yhdistelmiä. Jotkut työkalut mahdollistavat joidenkin (ali-)yhdistelmien pakolla sisällyttämisen tai poisjättämisen lopullisesta yhdistelmien joukosta. Testausasiantuntija voi käyttää tätä ominaisuutta painottaakseen tai vähentääkseen toimialueosaamiseen tai tuotteen käyttötietoihin perustuvien tekijöiden merkitystä.

Kattavuus
Kattavuudelle on monia tasoja. Alinta kattavuuden tasoa kutsutaan yksinkertaiseksi tai yksittäiskattavuudeksi. Se edellyttää, että joka parametrin jokainen arvo esiintyy ainakin yhdessä valituista yhdistelmistä. Seuraavaa kattavuuden tasoa kutsutaan kaksinkertaiseksi tai syöteparien kattavuudeksi. Se edellyttää, että minkä tahansa kahden parametrin arvojen jokainen pari esiintyy ainakin yhdessä yhdistelmässä. Tämä ajatus voidaan yleistää n-kertaiseksi kattavuudeksi, joka edellyttää, että jokainen n:n parametrin arvojen aliyhdistelmä esiintyy valittujen yhdistelmien joukossa. Mitä suurempi n:n arvo, sitä enemmän yhdistelmiä tarvitaan 100 % kattavuuden saavuttamiseen. Vähimmäiskattavuus näillä tekniikoilla on, että jokaista välineen tuottamaa yhdistelmää kohtaan on ainakin yksi testitapaus.

Vikatyypit
Yleisimmät tällä testaustyypillä löydetyt viat liittyvät useiden parametrien yhdistettyihin arvoihin.

3.2.7 Käyttötapaustestaus

Käyttötapaustestaus tuottaa tapahtumia käyttäviä, skenaariopohjaisia testejä, jotka pyrkivät emuloimaan järjestelmän käyttöä. Käyttötapausten määrittelyn pohjana on toimijan ja järjestelmän välinen toiminta, joka saavuttaa jonkin tavoitteen. Toimijat voivat olla käyttäjiä tai ulkoisia järjestelmiä.

Sovellettavuus
Käyttötapaustestausta tehdään yleensä järjestelmä- ja hyväksymistestaustasoilla. Sitä voidaan tehdä integroinnin tasosta riippuen integrointitestaustasolla ja komponenttien toiminnasta riippuen jopa yksikkötestauksessa. Käyttötapaukset muodostavat usein myös pohjan suorituskykytestaukselle, koska ne kuvaavat järjestelmän todellista käyttöä. Käyttötapauksissa kuvatut skenaariot voidaan toteuttaa käyttämällä virtuaalikäyttäjiä, jotta järjestelmään voidaan kohdistaa realistinen kuormitus.

Rajoitukset/vaikeudet
Jotta käyttötapaukset olisivat kelvollisia, niiden täytyy välittää tietoa todellisista käyttäjätapahtumista. Tämän tiedon pitäisi tulla käyttäjältä tai käyttäjän edustajalta. Käyttötapauksen arvo vähenee, jos käyttötapaus ei kuvaa todenmukaisia käyttäjän toimenpiteitä. Vaihtoehtojen polkujen (virtojen) tarkka kuvaaminen on tärkeää, jotta saavutettaisiin perusteellinen testikattavuus. Käyttötapausten pitäisi toimia ohjeina, mutta ei täysin kattavana kuvauksena sille, mitä pitäisi testata, sillä ne eivät ehkä anna täysin selvää kuvausta koko vaatimusjoukosta. Testauksen tarkkuuden parantamiseksi ja itse käyttötapauksen oikeellisuuden todentamiseksi voi olla myös hyödyllistä laatia käyttötapauksen kuvauksesta muita malleja, kuten vuokaavioita.

Kattavuus
Käyttötapauksen vähimmäiskattavuuden saavuttamiseksi tarvitaan yksi testitapaus (positiivista) pääpolkua varten ja yksi testitapaus kutakin vaihtoehtoista polkua tai vuota varten. Vaihtoehtoisiin polkuihin kuuluva poikkeukset ja vikatilannepolut. Vaihtoehtoiset polut kuvataan joskus pääpolun laajennuksina. Kattavuusprosentti määritetään laskemalla testattujen polkujen määrä ja jakamalla se pääpolun ja vaihtoehtoisten polkujen yhteismäärällä.

© International Software Testing Qualifications Board

Vikatyypit
Vikoihin kuuluvat määritettyjen skenaarioiden vääränlainen käsittely, puuttuvat vaihtoehtoisten polkujen käsittelyt, virheellinen esitettyjen ehtojen käsittely ja oudot tai väärät virheraportit.

3.2.8 Käyttäjätarinatestaus

Joissain ketterissä menetelmissä, kuten Scrumissa, vaatimukset kuvataan käyttäjätarinoina, jotka kuvaavat pieniä toiminnallisia kokonaisuuksia, jotka voidaan suunnitella, toteuttaa, testata ja esitellä yhden iteraatiokierroksen aikana [Cohn04]. Nämä käyttäjätarinat sisältävät toteutettavan toiminnallisuuden kuvauksen, mahdolliset ei-toiminnalliset vaatimukset, sekä myös hyväksymiskriteerit, joiden täytyy täyttyä, jotta käyttäjätarinaa voidaan pitää loppuunkäsiteltynä.

Sovellettavuus
Käyttäjätarinoita käytetään pääsiassa ketterissä ja vastaavissa iteratiivisissa ja inkrementaalisissa ympäristöissä. Niitä käytetään sekä toiminnalliseen että ei-toiminnalliseen testaukseen. Käyttäjätarinoita käytetään testauksen kaikilla tasoilla, ja oletuksena on, että toteuttaja esittelee käyttäjätarinan perusteella toteutetun toiminnallisuuden ennen kuin koodi luovutetaan tiimin jäsenille seuraavan tason testausta varten (esim. integrointi- tai suorituskykytestaus).

Rajoitukset/vaikeudet
Koska tarinat sisältävät toiminnallisuuden pieniä palasia, voi olla tarpeen laatia ajureita ja tynkiä, jotta toimitettu toiminnallisuuden osa päästään testaamaan kunnolla. Tämä vaatii yleensä ohjelmointitaitoja sekä testausta tukevien, kuten API-testaustyökalujen, käyttötaitoa. Tynkien ja ajureiden luominen on yleensä toteuttajan vastuulla, vaikka Tekninen testausasiantuntija voi osallistua myös koodin toteuttamiseen ja API-testaustyökalujen käyttöön. Tynkien ja ajureiden tarve minimoidaan, jos käytetään jatkuvan integraation mallia, kuten useimmissa ketterissä projekteissa on tapana.

Kattavuus
Käyttäjätarinan minimikattavuus tarkoittaa sen todentamista, että jokainen määritetty hyväksymiskriteeri on täytetty.

Vikatyypit
Viat ovat yleensä toiminnallisia, eli ohjelmisto ei pysty suorittamaan määrättyä toiminnallisuutta. Vikoja esiintyy myös, kun uuden tarinan toiminnallisuutta integroidaan jo olemassa olevaan toiminnallisuuteen. Koska tarinoita voidaan laatia toisistaan erillään, voi esiin nousta suorituskykyyn, rajapintoihin ja virheenkäsittelyyn liittyviä ongelmia. Testausasiantuntijan on tärkeää suorittaa sekä toimitetun yksittäisen toiminnallisuuden testaus että integrointitestaus aina, kun uusi tarina julkaistaan testattavaksi.

3.2.9 Arvoalueanalyysi

Arvoalue tarkoittaa määriteltyä arvojen joukkoa. Joukko voidaan määritellä yksittäisen muuttujan arvojen alueena (yksiulotteinen arvoalue, esim. ”iältään yli 24- ja alle 66-vuotiaat miehet”), tai toistensa kanssa tekemisissä olevien muuttujien arvojen alueina (moniulotteinen arvoalue, esim. ”iältään yli 24- ja alle 66-vuotiaat miehet, joiden paino on yli 69 ja alle 90 kiloa”). Moniulotteisen arvoalueen jokaisen testitapauksen täytyy sisältää jokaiselle mukaan otetulle muuttujalle soveltuvat arvot.

Yksiulotteisen arvoalueen arvoalueanalyysissä käytetään tyypillisesti ekvivalenssiluokitusta ja raja-arvoanalyysiä. Kun luokat on määritelty, Testausasiantuntija valitsee jokaisesta luokasta arvot, jotka edustavat arvoja luokan sisältä (IN), ulkopuolelta (OUT), luokan rajalta (ON) ja juuri rajan vierestä (OFF). Nämä arvot määrittelemällä jokainen luokka testataan myös sen rajaehtojen osalta. [Black07]

Moniulotteisten arvoalueiden kohdalla näitä menetelmiä käytettäessä testitapausten määrä nousee eksponentiaalisesti mukana olevien muuttujien määrän perusteella, kun taas arvoalueteorian käyttöön perustuva lähestymistapa johtaa lineaariseen kasvuun. Koska muodollinen lähestymistapa käyttää

© International Software Testing Qualifications Board

myös vikateoriaa (vikamallia), joka puuttuu ekvivalenssiluokituksesta ja raja-arvoanalyysistä, sen pienempi testijoukko löytää moniulotteisista arvoalueista myös vikoja, jotka laajemmalta, heuristiselta testijoukolta jäisivät todennäköisesti huomaamatta. Kun käsitellään moniulotteisia arvoalueita, testausmalli voidaan rakentaa päätöstauluna (tai “arvoaluematriisina”). Testitapausten arvojen tunnistaminen moniulotteisille arvoalueille, joissa on yli kolme ulottuvuutta, vaatii yleensä laskennallista tukea.

Sovellettavuus
Arvoalueanalyysi yhdistää tekniikoita, joita käytetään päätöstauluissa, ekvivalenssiluokituksessa ja raja-arvoanalyysissä, jotta pystytään luomaan pienempiä testijoukkoja, jotka silti kattavat tärkeät ja todennäköisesti häiriöitä aiheuttavat alueet. Sitä käytetään usein tilanteissa, joissa päätöstaulut olisivat liian hankalia käsiteltäväksi mahdollisesti toistensa kanssa tekemisissä olevien muuttujien määrän vuoksi. Arvoalueanalyysia voidaan käyttää kaikilla testaustasoilla, mutta useimmin sitä käytetään integrointi- ja järjestelmätestaustasoilla.

Rajoitukset/vaikeudet
Perusteellinen arvoalueanalyysi vaatii ohjelmiston vahvaa tuntemusta, jotta eri arvoalueet ja niiden väliset mahdolliset vuorovaikutukset voidaan tunnistaa. Jos arvoalue jää tunnistamatta, testaus voi jäädä merkittävästi puutteelliseksi, mutta on todennäköistä, että arvoalue löydetään, koska OFF- ja OUT-muuttujat voivat osua tunnistamattomalle arvoalueelle. Arvoalueanalyysi on vahva tekniikka käytettäväksi, kun työskennellään toteuttajan kanssa ja ollaan määrittämässä testattavia alueita.

Kattavuus
Arvoalueanalyysin minimikattavuuden saavuttamiseksi tarvitaan testitapaus joka arvoalueen jokaiselle IN-, OUT-, ON- ja OFF-arvolle. Mikäli arvojen kohdalla esiintyy päällekkäisyyksiä (esimerkiksi yhden arvoalueen OUT-arvo on toisen arvoalueen IN-arvo), ei testejä tarvitse kahdentaa. Tämän vuoksi todellinen tarvittavien testien määrä on usein vähemmän kuin neljä arvoaluetta kohti.

Vikatyypit
Vikoihin kuuluvat ongelmat arvoalueen toiminnallisuudessa, raja-arvojen käsittelyssä, muuttujien välisessä yhteistoiminnassa sekä virheenkäsittelyssä (erityisesti, kun on kyse arvoista, jotka eivät ole kelvollisella arvoalueella).

3.2.10 Tekniikoiden yhdistäminen

Joskus tekniikoita yhdistellään testitapausten luomiseksi. Esimerkiksi päätöstaulujen avulla tunnistetut ehdot voidaan luokitella ekvivalenssiosituksen avulla, jotta löydetään erilaisia tapoja, joilla ehdot voidaan täyttää. Tällöin testitapaukset kattavat kaikki ehtoyhdistelmät, mutta lisäksi luokiteltuja ehtoja varten luodaan lisätestitapauksia ekvivalenssiluokkien kattamiseksi. Kun Testausasiantuntija valitsee käytettäviä tekniikoita, hänen on harkittava tekniikan soveltuvuutta, rajoituksia ja haasteita sekä testaukselle kattavuuden kannalta asetettuja tavoitteita ja löydettäväksi tarkoitettuja vikoja. Tilanteeseen ei välttämättä ole yhtä “parasta” tekniikkaa. Tekniikoiden yhdistelmät tarjoavat usein parhaimman kattavuuden edellyttäen, että tekniikoiden oikeanlaiseen käyttöön on käytettävissä riittävästi aikaa ja osaamista.

3.3 Vikaperusteiset tekniikat

3.3.1 Vikaperusteisten tekniikoiden käyttö

Vikaperusteisessa testisuunnittelutekniikassa testisuunnittelun pohjana käytetään etsittävien vikojen tyyppiä ja testitapaukset suunnitellaan järjestelmällisesti sen perusteella, mitä vikatyypistä tiedetään. Toisin kuin määrittelypohjaiset tekniikat, joissa testit suunnitellaan määrittelykuvausten pohjalta, vikaperusteinen testaus käyttää testien suunnitteluun virheluokitteluja (eli ryhmiteltyjä listoja), jotka

© International Software Testing Qualifications Board

voivat olla täysin riippumattomia testattavasta ohjelmistosta. Luokitteluihin voi kuulua luetteloita vikatyypeistä, alkuperäissyistä, häiriöiden oireista ja muista vikoihin liittyvistä tiedoista. Vikaperusteinen testaus voi myös hyödyntää testauksen kohdistamisessa tunnistettujen riskien ja riskitilanteiden luetteloita. Tämä tekniikka antaa testaajalle mahdollisuuden kohdistaa testaus tietyntyyppisiin vikoihin tai käydä järjestelmällisesti läpi tunnetut ja tavanomaiset viat. Testausasiantuntija käyttää luokittelutietoja määritellessään testauksen tavoitetta, joka on tietyntyyppisen vikatyypin löytäminen. Tämän tiedon perusteella Testausasiantuntija luo testitapaukset ja testattavat tilanteet, jotka saavat virheen paljastumaan, mikäli sellainen on olemassa.

Sovellettavuus
Vikaperusteista testausta voidaan käyttää kaikilla testaustasoilla, mutta tyypillisimmin sitä käytetään järjestelmätestauksen aikana. Vioille on olemassa vakioluokitteluja, jotka soveltuvat monentyyppisiin ohjelmistoihin. Tämä tuoteriippumaton testaustyyppi auttaa nostamaan teollisuusstandardien osaamista määrätyntyyppisten testien laatimiseksi. Teollisuudenalakohtaisia luokitteluja käyttämällä vikojen ilmentymisestä kerätyt mittaritiedot voidaan jäljittää yli projektirajojen ja jopa läpi organisaation.

Rajoitukset/vaikeudet
Vikaluokitteluja on olemassa monia ja ne voivat keskittyä määrätyntyyppiseen testaukseen, kuten esimerkiksi käytettävyyteen. Jos luokitteluja on käytettävissä, on tärkeää valita luokittelu, joka soveltuu testattavana olevaan ohjelmistoon. Voi olla, että esimerkiksi uutta, innovatiivista ohjelmistoa varten ei ole saatavilla luokitteluja. Jotkut organisaatiot ovat laatineet omia luokittelujaan todennäköisistä tai usein havaituista vioista. Mitä tahansa luokittelua käytetäänkin, on tärkeää määritellä odotettu kattavuus ennen testauksen aloittamista.

Kattavuus
Tämä tekniikka tuottaa kattavuuskriteereitä, joita käytetään määrittämään, milloin kaikki hyödylliset testitapaukset on tunnistettu. Käytännön kannalta vikaperusteisten tekniikoiden kattavuuskriteerit ovat yleensä epäjärjestelmällisempiä kuin määrittelypohjaisten tekniikoiden, sillä kattavuudelle annetaan vain yleisiä sääntöjä ja tarkka päätös hyödyllisen kattavuuden rajasta on harkinnan varassa. Kuten muidenkin tekniikoiden kohdalla, kattavuuskriteerit eivät tarkoita sitä, että koko testijoukko on tyhjentävä, vaan että tarkastelun kohteena olevat viat eivät enää luo pohjaa uusille hyödyllisille testeille tätä tekniikkaa käyttämällä.

Vikatyypit
Löydettävien vikojen tyypit riippuvat yleensä käytettävästä luokittelusta. Jos käytetään käyttöliittymäluokittelua, pääosa löydetyistä vioista liittyy todennäköisesti testattavaan käyttöliittymään, mutta muita vikoja voidaan löytää testauksen sivutuotteena.

3.3.2 Vikaluokittelut

Vikaluokittelut ovat vikatyyppien ryhmiteltyjä listoja. Nämä listat voivat olla hyvin yleisellä tasolla ja niitä voidaan käyttää karkean tason ohjeena, tai ne voivat olla hyvinkin yksityiskohtaisia. Esimerkiksi käyttöliittymävikojen luokittelu voi sisältää yleisiä asioita, kuten toiminnallisuus, vikojenkäsittely, grafiikan näyttö ja suorituskyky. Yksityiskohtainen luokittelu voi sisältää luettelon kaikista mahdollisista käyttöliittymän objekteista (erityisesti graafisen käyttöliittymän kohdalla) ja määritellä näiden objektien vääränlaisen käsittelyn, kuten esimerkiksi:

- Tekstikenttä
  - Kelvollista tietoa ei hyväksytä
  - Epäkelpo tieto hyväksytään
  - Syötteen pituutta ei todenneta
  - Erikoismerkkejä ei löydetä
  - Virheilmoitukset eivät ole informatiivisia
  - Käyttäjä ei pysty korjaamaan virheellistä tietoa
  - Sääntöjä ei noudateta

© International Software Testing Qualifications Board

- Päivämääräkenttä
  - Kelvollista päivämäärää ei hyväksytä
  - Epäkelpoa päivämäärää ei hylätä
  - Päivämääräalueita ei todenneta
  - Tiedon tarkkuutta ei käsitellä oikein (esim. hh:mm:ss)
  - Käyttäjä ei pysty korjaamaan virheellistä tietoa
  - Sääntöjä ei noudateta (esim. lopetuspäivän pitää olla suurempi kuin aloituspäivän)

Saatavilla on monenlaisia luokitteluja, jotka vaihtelevat kaupallisesti hankittavista muodollisista luokitteluista organisaatiokohtaisiin määrättyyn tarkoitukseen suunniteltuihin. Sisäisesti laadittuja vikaluokitteluja voidaan käyttää myös tiettyjen organisaatiossa tyypillisesti löydettyjen vikojen paljastamiseen. Kun luodaan uutta tai muokataan jo olemassa olevaa luokittelua, on tärkeää ensin määritellä luokittelun päämäärät tai tavoitteet. Tavoitteena voi esimerkiksi olla tuotantojärjestelmissä havaittujen käyttöliittymäongelmien tai syötekenttien käsittelyyn liittyvien ongelmien tunnistaminen.

Luokittelun laatimiseksi

1. Laadi tavoite ja määrittele tavoiteltu yksityiskohtaisuuden taso.
2. Valitse perustana käytettävä luokittelu.
3. Määrittele organisaation sisäisten ja/tai ulkopuolelta saatujen kokemusten perusteella arvot ja tyypilliset viat.

Mitä yksityiskohtaisempi luokittelu on, sitä enemmän aikaa kuluu sen laatimiseen ja ylläpitoon, mutta sen tuloksena on testitulosten parempi toistettavuus. Yksityiskohtaisissa luokitteluissa voi olla päällekkäisyyksiä, mutta niiden avulla testaustiimi voi jakaa testauksen ilman, että tiedonkulku tai kattavuus kärsivät.

Kun sopiva luokittelu on valittu, sitä voidaan käyttää testattavien tilanteiden ja testitapausten luomiseen. Riskipohjainen luokittelu voi auttaa testausta keskittymään tiettyyn riskialueeseen. Luokitteluita voidaan käyttää myös ei-toiminnallisilla alueilla, kuten käytettävyys, suorituskyky jne. Luokitteluita on saatavilla eri julkaisuista, IEEE:ltä ja Internetistä.

3.4 Kokemusperusteiset tekniikat

Kokemuspohjaiset testit hyödyntävät testaajien osaamista ja intuitiota samoin kuin heidän kokemuksiaan samanlaisista sovelluksista tai teknologioista. Nämä testit löytävät tehokkaasti vikoja mutta eivät ole yhtä soveltuvia saavuttamaan määrättyjä testikattavuustasoja tai tuottamaan uudelleenkäytettäviä testiproseduureja kuin muut tekniikat. Kokemuspohjainen testaus voi olla hyvä vaihtoehto järjestelmällisemmille lähestymistavoille, mikäli järjestelmän dokumentaatio on heikkoa, testauksella on tiukat aikarajat tai testaustiimillä on laaja kokemus testattavasta järjestelmästä. Kokemuspohjainen testaus ei välttämättä sovi sellaisten järjestelmien testaamiseen, joilta vaaditaan yksityiskohtaista dokumentaatiota, suurta testien toistettavuutta tai että testikattavuus voidaan arvioida tarkasti.

Testaajat käyttävät yleensä kokemuspohjaista testausta, kun käytetään dynaamista ja heuristista lähestymistapaa, ja testaus on enemmän tapahtumiin reagoivaa kuin ennalta suunnitellut testauksen lähestymistavat. Lisäksi testien suoritus ja arviointi tapahtuvat yhtä aikaa. Jotkut kokemuspohjaiseen testaukseen liittyvät systemaattiset lähestymistavat eivät ole täysin dynaamisia, eli testejä ei luoda kokonaan samaan aikaan kuin testaaja suorittaa testit.

On huomattava, että vaikka tässä esiteltyjen tekniikoiden kohdalla esitetäänkin joitain ideoita testikattavuuden suhteen, kokemuspohjaisilla tekniikoilla ei ole muodollisia kattavuuskriteereitä.

© International Software Testing Qualifications Board

3.4.1 Virheenarvaus

Virheenarvaus-tekniikkaa käyttäessään Testausasiantuntija käyttää kokemustaan arvatakseen mahdollisia virheitä, joita on voitu tehdä koodin suunnittelussa ja toteutuksessa. Kun odotetut virheet on tunnistettu, Testausasiantuntija määrittää parhaat tavat, joilla virheistä seuranneet viat voidaan paljastaa. Jos Testausasiantuntija esimerkiksi odottaa, että ohjelmistossa esiintyy häiriöitä, kun syötetään virheellinen salasana, suunnitellaan testit, joilla syötetään joukko erilaisia arvoja salasana-kenttään sen todentamiseksi, onko todella tehty virhe, joka on johtanut vikaan, joka voidaan havaita häiriönä testien suorittamisen aikana.

Sen lisäksi, että virheenarvausta käytetään testaustekniikkana, se on myös hyödyllistä riskianalyysin yhteydessä mahdollisten häiriötilojen tunnistamisessa. [Myers79]

Sovellettavuus
Virheenarvausta tehdään pääasiassa integrointi- ja järjestelmätestauksessa, mutta sitä voidaan käyttää millä testaustasolla tahansa. Tätä tekniikkaa käytetään usein muiden tekniikoiden kanssa ja se auttaa laajentamaan olemassa olevien testitapausten ulottuvuutta. Virheenarvausta voidaan käyttää myös tehokkaasti, kun testataan ohjelmiston uutta julkaisua tavallisten erehdysten ja virheiden löytämiseksi ennen perusteellisemman ja skriptatun testauksen aloittamista. Tarkistuslistat ja virheluettelot voivat olla hyödyllisiä testauksen suunnittelussa.

Rajoitukset/vaikeudet
Kattavuutta on vaikea arvioida ja se vaihtelee suuresti riippuen Testausasiantuntijan taidoista ja kokemuksesta. Parhaiten tekniikkaa sopii käyttämään kokenut testaaja, jolle testattavana olevalle koodille tyypilliset virhetyypit ovat tuttuja. Virheenarvausta käytetään yleisesti, mutta testaus jää usein dokumentoimatta ja siksi se on vaikeammin toistettavaa kuin muut testausmuodot.

Kattavuus
Luokitteluja käytettäessä kattavuuden määräävät aineistoon liittyvät tyypilliset viat ja vikatyypit. Ilman luokitteluja kattavuutta rajoittavat testaajan kokemus ja osaaminen sekä käytettävissä oleva aika. Tästä tekniikasta saatavat hyödyt vaihtelevat riippuen siitä, kuinka hyvin testaaja pystyy kohdentamaan testauksen ongelmallisiin alueisiin.

Vikatyypit
Tyypillisiä vikoja ovat yleensä kyseisessä luokittelussa määritellyt tai Testausasiantuntijan “arvaamat” viat, jotka ovat voineet jäädä löytymättä määrittelypohjaisessa testauksessa.

3.4.2 Tarkistuslistoihin pohjautuva testaus

Tarkistuslistoihin pohjautuvaa testaustekniikkaa käyttäessään kokenut Testausasiantuntija käyttää apuna karkean tason yleistä listaa asioista, jotka pitää huomata, tarkistaa tai muistaa, tai joukkoa sääntöjä tai kriteereitä, joita vastaan tuote pitää todentaa. Nämä tarkistuslistat on laadittu standardien, kokemuksen ja muiden huomioon otettavien seikkojen perusteella. Esimerkki tarkistuslistapohjaisesta testistä on käyttöliittymästandardien tarkistuslista, jota käytetään sovelluksen testauksen pohjana.

Sovellettavuus
Tarkistuslistoihin pohjautuva testaus on tehokkaimmillaan projekteissa, joissa kokenut testaustiimi tuntee hyvin testattavan ohjelmiston tai tarkistuslistan kattaman alueen (esim. Testausasiantuntija pystyy menestyksekkäästi käyttämään käyttöliittymän tarkistuslistaa, jos hänellä on kokemusta käyttöliittymätestauksesta mutta ei välttämättä kyseisestä testattavasta ohjelmistosta). Koska tarkistuslistat ovat hyvin karkealla tasolla ja niistä puuttuvat yksityiskohtaiset askeleet, jotka tyypillisesti ovat olemassa testitapauksissa ja testiproseduureissa, testaajan osaamista käytetään täyttämään väliin jäävät aukot. Koska yksityiskohtaiset askeleet puuttuvat, tarkistuslistojen ylläpito on helppoa ja niitä voidaan käyttää monissa samanlaisissa julkaisuissa. Tarkistuslistoja voidaan käyttää kaikilla testaustasoilla. Niitä käytetään myös regressiotestauksessa ja aloitustestauksessa.

© International Software Testing Qualifications Board

Rajoitukset/vaikeudet
Tarkistuslistojen sisällön yksityiskohtaisuuden puute voi vaikuttaa testitulosten toistettavuuteen. On mahdollista, että eri testaajat tulkitsevat tarkistuslistoja eri tavalla ja noudattavat eri lähestymistapoja tarkistuslistan kohtien läpikäynnissä. Tämä voi tuottaa erilaisia tuloksia, vaikka käytössä on sama tarkistuslista. Tämä voi johtaa laajempaan kattavuuteen, mutta joskus toistettavuuden kustannuksella. Tarkistuslistojen käyttö voi myös aiheuttaa liikaa luottamusta saavutettuun kattavuuden tasoon, sillä varsinainen testaus riippuu testaajan päätöksistä. Tarkistuslistoja voidaan laatia yksityiskohtaisemmista testitapauksista ja niillä on taipumus kasvaa ajan myötä. Listoja täytyy ylläpitää sen varmistamiseksi, että ne kattavat kaikkein tärkeimmät testattavan ohjelmiston piirteet.

Kattavuus
Kattavuuden taso on yhtä hyvä kuin itse tarkistuslista, mutta listan karkean luonteen vuoksi tulokset vaihtelevat sen mukaan, kuka Testausasiantuntija tarkistuslistaa käyttää.

Vikatyypit
Tyypillisiin tällä tekniikalla löydettäviin vikoihin kuuluvat häiriöt, jotka ovat tulosta aineiston, askeljärjestyksen tai yleisen työnkulun muutoksista testauksen aikana. Tarkistuslistojen käyttö auttaa pitämään testausnäkökulman tuoreena, koska uusia aineiston ja prosessien yhdistelmiä sallitaan testauksen aikana.

3.4.3 Tutkiva testaus

Tutkivassa testauksessa testaaja samanaikaisesti kerää tietoja tuotteesta ja sen vioista, suunnittelee tehtävän testaustyön, laatii ja suorittaa testit ja raportoi tulokset. Testaaja muokkaa dynaamisesti testauksen tavoitteita suorituksen aikana ja laatii vain kevyen dokumentaation. [Whittaker09]

Sovellettavuus
Hyvä tutkiva testaus on suunniteltua, interaktiivista ja luovaa. Se vaatii vain vähän dokumentaatiota testattavasta järjestelmästä ja sitä käytetään usein tilanteissa, joissa dokumentaatiota ei ole saatavilla tai se ei riitä muihin testaustekniikoihin. Tutkivaa testausta käytetään usein muun testauksen lisänä ja lisätestitapausten luonnin pohjana.

Rajoitukset/vaikeudet
Tutkivan testauksen hallinnointi ja aikataulutus voi olla hankalaa. Kattavuus voi olla epätasaista ja toistettavuus on vaikeaa. Yksi tapa hallita tutkivaa testausta on käyttää testausohjeita määrittelemään testaussession aikana katettavat alueet ja aikalaatikoita määrittämään testaukselle käytettävissä oleva aika. Testausistunnon tai -istuntojoukon päätteeksi testauspäällikkö voi pitää jälkipalaverin, jossa kerätään testien tulokset ja määritetään seuraavien istuntojen testausohjeet. Jälkipalavereita on vaikea laajentaa isoille testaustiimeille tai isoihin projekteihin sopiviksi.

Toinen testausistuntoihin liittyvä vaikeus on niiden tarkka seuranta testauksenhallintajärjestelmässä. Joskus se hoidetaan laatimalla testitapauksia, jotka todellisuudessa ovat tutkivan testauksen istuntoja. Tämä mahdollistaa tutkivaan testaukseen määritetyn ajan ja sille suunnitellun kattavuuden seurannan muiden testaustöiden ohella.

Koska tutkivassa testauksessa toistettavuus voi olla vaikeaa, se saattaa aiheuttaa ongelmia myös, kun on tarpeen toistaa testiaskeleet häiriön uudelleentuottamiseksi. Jotkut organisaatiot käyttävät testiautomaatiotyökalun nauhoita/toista-ominaisuutta tutkivaa testausta tekevän testaajan toimenpiteiden tallentamiseen. Tämä tuottaa täydellisen nauhoituksen kaikista tutkivan testauksen istunnon (tai minkä tahansa kokemuspohjaisen testausistunnon) aikana tehdyistä toimenpiteistä. Voi olla työlästä kahlata läpi kaikki yksityiskohdat häiriön todellisen syyn löytämiseksi, mutta ainakin olemassa on tallenne kaikista tilanteeseen liittyvistä askelista.

© International Software Testing Qualifications Board

Kattavuus
Tehtävien, tavoitteiden ja tuotosten määrittämiseksi voidaan luoda testausohjeita. Tutkivan testauksen istunnot suunnitellaan sen jälkeen kyseisten tavoitteiden saavuttamiseksi. Testausohje voi myös määrittää, mihin testaustyöt keskitetään, mikä kuuluu testausistunnossa käsiteltävään alueeseen ja mikä ei, ja mitkä resurssit pitäisi kiinnittää suunniteltujen testien loppuun saattamiseksi. Istunnossa voidaan keskittyä määrätyn tyyppisiin vikoihin ja muihin mahdollisesti ongelmallisiin alueisiin, joita voidaan tarkastella ilman muodollista skriptattua testausta.

Vikatyypit
Tutkivan testauksen avulla tyypillisesti löytyvät viat ovat skenaariopohjaisia ongelmia, jotka ovat jääneet huomaamatta skriptatun toiminnallisen testauksen aikana, toiminnallisten rajojen väliin jääviä ongelmia, sekä työnkulkuihin liittyviä ongelmia. Joskus myös suorituskyky- ja tietoturvaongelmia paljastuu tutkivan testauksen aikana.

3.4.4 Parhaan tekniikan käyttäminen

Vika- ja kokemusperusteiset tekniikat vaativat vioista ja muusta testauskokemuksesta peräisin olevan tiedon soveltamista testauksen kohdentamiseksi, jotta löydetään enemmän vikoja. Ne vaihtelevat “pikatesteistä”, joissa testaajalla ei ole ennalta suunniteltuja toimenpiteitä suoritettavanaan, ennalta suunniteltujen testien kautta skriptattuihin testausistuntoihin. Vika- ja kokemusperusteiset tekniikat ovat miltei aina hyödyllisiä, mutta erityisesti niistä on arvoa seuraavissa tilanteissa:

- Määrittelyjä ei ole saatavilla
- Testattavan järjestelmän dokumentaatio on heikkolaatuinen
- Yksityiskohtaisten testien suunnitteluun ja laatimiseen ei ole riittävästi aikaa
- Testaajilla on asiantuntemusta liiketoiminta-alueesta ja/tai teknologiasta
- Skriptatusta testauksesta poikkeamisen tarkoituksena on maksimoida testikattavuus
- Toiminnalliset häiriöt tullaan analysoimaan.

Vika- ja kokemusperusteiset tekniikat ovat myös hyödyllisiä, kun niitä käytetään yhdessä määrittelypohjaisten tekniikoiden kanssa, sillä ne täyttävät testikattavuuteen syntyvät aukot, jotka ovat peräisin määrittelypohjaisten tekniikoiden perusheikkouksista. Kuten määrittelypohjaisten tekniikoidenkin kohdalla, kaikkiin tilanteisiin ei ole yhtä ainoaa täydellistä tekniikkaa. Testausasiantuntijan on tärkeää ymmärtää jokaisen tekniikan edut ja haitat ja pystyä valitsemaan testaustekniikoiden joukosta tilanteeseen parhaiten sopiva tekniikka ottaen huomioon projektin tyyppi, aikataulu, tietojen saatavuus, testaajan osaaminen ja muut seikat, jotka voivat vaikuttaa valintaan.

© International Software Testing Qualifications Board

4. Ohjelmiston laatuominaisuuksien testaaminen - 120 min.

Avainsanat

esteettömyyden testaus, heuristinen arviointi, houkuttelevuus, käytettävyystestaus, käyttökelpoisuus, opittavuus, soveltuvuustestaus, SUMI, tarkkuustestaus, WAMMI yhteentoimivuustestaus, ymmärrettävyys

Oppimistavoitteet: Ohjelmiston laatuominaisuuksien testaus

4.2 Liiketoiminta-alueen testauksen laatuominaisuudet

TA-4.2.1 (K2) Selittää esimerkkien avulla, mitkä testaustekniikat soveltuvat tarkkuuden, soveltuvuuden, yhteentoimivuuden ja yhdenmukaisuuden testaukseen.

TA-4.2.2 (K2) Määritellä tarkkuuden, soveltuvuuden ja yhteentoimivuuden kannalta tyypilliset vikatyypit, joihin testaus kohdistetaan.

TA-4.2.3 (K2) Määritellä tarkkuuden, soveltuvuuden ja yhteentoimivuuden kannalta, missä vaiheessa elinkaaren aikana nämä ominaisuudet pitäisi testata.

TA-4.2.4 (K4) Määritellä tietyn projektin tilanteen näkökulmasta mitkä lähestymistavat olisivat soveliaita, kun pyritään todentamaan ja kelpuuttamaan sekä käytettävyysvaatimusten toteuttaminen että käyttäjän odotusten täyttyminen.

© International Software Testing Qualifications Board

4.1 Esittely

Siinä missä edellisessä luvussa on kuvattu erilaisia testaajan käytettävissä olevia tekniikoita, tässä luvussa käsitellään kyseisten tekniikoiden käyttöä, kun arvioidaan pääominaisuuksia, joilla kuvataan ohjelmistosovellusten tai järjestelmien laatua.

Tässä sertifikaattisisällössä käsitellään laatuominaisuuksia, joita Testausasiantuntija voi arvioida. Ominaisuuksia, joiden arviointi on Teknisen testausasiantuntijan tehtävä, käsitellään Teknisen testausasiantuntijan Jatkotason sertifikaattisisällössä. Ominaisuuksien kuvausta ohjaamaan käytetään ISO 9126:n sisältämää tuotteen laatuominaisuuksien kuvausta. Muitakin standardeja, kuten ISO 25000-sarja [ISO25000] (joka on korvannut ISO 9126:n), voidaan käyttää. ISO laatuominaisuudet on jaettu tuotteen laatuominaisuuksiin (attribuutteihin), joista jokaisella voi olla aliominaisuuksia (ali-attribuutteja). Ne on esitetty alla olevassa taulukossa, johon on myös merkitty, mitkä laatuominaisuudet/aliominaisuudet käsitellään Testausasiantuntija- ja Tekninen testausasiantuntija-sertifikaattisisällöissä.

Ominaisuus	**Aliominaisuus**	**Testaus-asiantuntija**	**Tekninen testausasiantuntija**
Toiminnallisuus	Tarkkuus, soveltuvuus, yhteentoimivuus, yhdenmukaisuus	X	
	Tietoturva		X
Luotettavuus	Kypsyys (vakaus), vikasietoisuus, toipuvuus, yhdenmukaisuus		X
Käytettävyys	Ymmärrettävyys, opittavuus, käyttö-kelpoisuus, viehättävyys, yhdenmukaisuus	X	
Toiminnallinen tehokkuus	Suorituskyky (aikakäyttäytyminen), resurssien käyttö, yhdenmukaisuus		X
Ylläpidettävyys	Analysoitavuus, muutettavuus, vakaus, testattavuus, yhdenmukaisuus		X
Siirrettävyys	Sovitettavuus, asennettavuus, yhdessä-toimivuus, korvattavuus, yhdenmukaisuus		X

Testausasiantuntijan tulisi keskittyä toiminnallisuuteen ja käytettävyyteen liittyviin ohjelmiston laatuominaisuuksiin. Testausasiantuntijan pitäisi suorittaa myös esteettömyyden testausta. Vaikka sitä ei ole listattu aliominaisuutena, sitä pidetään usein osana käytettävyystestausta. Muiden laatu-ominaisuuksien testauksen katsotaan yleensä kuuluvan Teknisen testausasiantuntijan vastuulle. Vaikka tämä työnjako voi vaihdella organisaatioittain, näissä ISTQB sertifikaattisisällöissä noudatetaan yllä esitettyä.

Aliominaisuus ”yhdenmukaisuus” esiintyy jokaisen laatuominaisuuden kohdalla. Tiettyjen turvallisuuskriittisten tai muuten säädeltyjen ympäristöjen kohdalla jokaisen laatuominaisuuden täytyy ehkä olla yhdenmukainen määrättyjen standardien ja säädösten kanssa (esim. toiminnallisuuden yhdenmukaisuus voi tarkoittaa, että toiminnallisuus noudattaa tiettyä standardia, kuten että käytetään tiettyä viestintäprotokollaa, jotta tietoja voidaan lähettää/ottaa vastaan mikrosirulta). Koska nämä standardit voivat vaihdella laajasti teollisuudenalan mukaan, niihin ei paneuduta syvällisemmin tässä yhteydessä. Jos Testausasiantuntija työskentelee ympäristössä, johon yhdenmukaisuusvaatimukset vaikuttavat, on tärkeää ymmärtää kyseiset vaatimukset ja varmistua siitä, että sekä testaus että testausdokumentaatio tulevat täyttämään ne.

Kaikkien tässä kappaleessa käsiteltävien laatuominaisuuksien ja aliominaisuuksien osalta on tunnistettava tyypilliset riskit sopivan testausstrategian laatimiseksi ja dokumentoimiseksi. Laatuominaisuuksien testaus vaatii, että erityistä huomiota kiinnitetään elinkaaren ajankohtaan, tarvittaviin työkaluihin, ohjelmiston ja dokumentaation saatavuuteen sekä tekniseen

© International Software Testing Qualifications Board

asiantuntemukseen. Mikäli jokaisen ominaisuuden ja sen yksilöllisten testausvaatimusten käsittelemiseksi ei laadita strategiaa, testausaikatauluun ei ehkä varata testaajalle riittävästi aikaa testauksen suunnitteluun ja käynnistämiseen sekä testien suoritukseen. Osa tästä testauksesta, esim. käytettävyystestaus, voi vaatia erityisasiantuntijoiden käyttöä, laajaa suunnittelua, erityisesti sitä varten tarkoitettua laboratoriota, erityistyökaluja, erikoistunutta testausosaamista ja useimmissa tapauksissa huomattavan paljon aikaa. Joissakin tapauksissa käytettävyystestauksen voi suorittaa erillinen käytettävyysryhmä tai käytettävyyden asiantuntija.

Laatuominaisuuksien ja aliominaisuuksien testaus täytyy sisällyttää testauksen kokonaisaikatauluun ja siihen pitää varata riittävästi resursseja. Jokaisella näistä alueista on omat erityistarpeensa, jokaisella niistä on tietyt kohteet ja ne saattavat tapahtua eri aikoihin ohjelmistokehityksen elinkaaren aikana, kuten edempänä todetaan.

Vaikka Testausasiantuntija ei ehkä ole vastuussa teknisempää lähestymistä vaativista laatuominaisuuksista, on tärkeää, että hän on tietoinen muista ominaisuuksista ja ymmärtää niiden testauksen kannalta päällekkäiset alueet. Esimerkiksi tuote, joka ei läpäise suorituskykytestejä, ei todennäköisesti läpäise myöskään käytettävyystestausta, jos se on liian hidas, jotta käyttäjä voi käyttää sitä tehokkaasti. Yhtä lailla tuote, jossa on yhteentoimivuusongelmia joidenkin komponenttien kanssa, ei todennäköisesti ole valmis siirrettävyystestaukseen, koska perusongelmilla on yleensä taipumus jäädä piiloon ympäristön muuttuessa.

4.2 Liiketoiminta-alueen testauksen laatuominaisuudet

Testausasiantuntijalle testauksen pääpainopiste on toiminnallisessa testauksessa. Toiminnallinen testaus keskittyy siihen, “mitä” tuote tekee. Toiminnallisen testauksen pohjamateriaalina ovat yleensä vaatimus- tai määrittelydokumentaatio, määrätty toimialueen asiantuntemus tai esiin tuotu tarve. Toiminnalliset testit vaihtelevat sen mukaan, millä testaustasolla ne suoritetaan, ja myös ohjelmistokehityksen elinkaari voi vaikuttaa niihin. Esimerkiksi integrointitestauksen aikana tehty toiminnallinen testi testaa keskenään vuorovaikutuksessa olevien moduuleiden toiminnallisuutta, joka tuottaa yhden määritellyn toiminnon. Järjestelmätestaustasolla toiminnallisiin testeihin kuuluu sovelluksen toiminnallisuuden testaus yhtenä kokonaisuutena. Järjestelmistä koostuvien järjestelmien kohdalla testaus keskittyy ensisijassa testaamaan integroidut järjestelmät läpi päästä päähän. Ketterässä kehitysympäristössä toiminnallinen testaus rajoittuu yleensä toiminnallisuuteen, joka on saatavilla tietyn iteraation tai sprintin aikana, vaikka iteraation regressiotestaus saattaakin kattaa kaiken julkaistun toiminnallisuuden.

Toiminnallisen testauksen aikana käytetään laajaa joukkoa erilaisia testaustekniikoita (ks. luku 3). Toiminnalliset testit voi suorittaa testaukseen keskittynyt henkilö (testaaja), toimialueen asiantuntija tai toteuttaja (yleensä yksikkötestaustasolla).

Tässä kappaleessa käsitellyn toiminnallisen testauksen lisäksi on kaksi laatuominaisuutta eli ei-toiminnallisen testaukseen (keskittyy siihen, ”kuinka” tuote saa toiminnallisuuden aikaan) kuuluvaa piirrettä, jotka kuuluvat Testausasiantuntijan vastuualueeseen. Nämä kaksi ei-toiminnallista ominaisuutta ovat käytettävyys ja esteettömyys.

Tässä luvussa käsitellään seuraavia laatuominaisuuksia:

- toiminnallisen laadun aliominaisuudet
  - tarkkuus
  - soveltuvuus
  - yhteentoimivuus
- ei-toiminnalliset laatuominaisuudet
  - käytettävyys
  - esteettömyys

© International Software Testing Qualifications Board

4.2.1 Tarkkuustestaus

Toiminnallisessa tarkkuustestauksessa testataan, kuinka hyvin sovellus noudattaa määritettyjä tai epäsuorasti kerrottuja vaatimuksia, ja siihen voi kuulua myös laskennallisen tarkkuuden testaus. Tarkkuustestaus käyttää monia luvussa 3 selitettyjä testaustekniikoita ja se hyödyntää usein myös määrittelyjä tai vanhaa järjestelmää testioraakkelina. Tarkkuustestausta voidaan tehdä missä tahansa elinkaaren vaiheessa ja se kohdistuu tietojen tai tilanteiden vääränlaiseen käsittelyyn.

4.2.2 Soveltuvuustestaus

Soveltuvuustestauksessa arvioidaan ja kelpuutetaan toimintojoukon soveltuvuus sille tarkoitettuihin määrättyihin tehtäviin. Tämä testaus voi perustua käyttötapauksiin. Soveltuvuustestausta tehdään yleensä järjestelmätestauksen aikana, mutta sitä voidaan tehdä myös integrointitestauksen myöhemmissä vaiheissa. Testauksen aikana löydetyt viat kertovat siitä, että järjestelmä ei pysty täyttämään käyttäjän tarpeita hyväksyttävällä tavalla.

4.2.3 Yhteentoimivuustestaus

Yhteentoimivuustestaus testaa, kuinka hyvin kaksi tai useampia järjestelmiä tai komponentteja voi vaihtaa tietoja keskenään ja tämän jälkeen käyttää näitä tietoja. Testauksen tulee kattaa kaikki aiotut kohdeympäristöt (mukaan luettuna eri laitteistot, ohjelmistot, väliohjelmistot, käyttöjärjestelmät jne.) sen varmistamiseksi, että tiedonvaihto toimii oikein. Käytännössä tämä voi olla järkevää vain suhteellisen pienessä joukossa erilaisia ympäristöjä. Siinä tapauksessa yhteentoimivuustestaus voidaan rajata ympäristöjä edustavaan valikoituun ympäristöjoukkoon. Yhteentoimivuustestien määrittely edellyttää, että aiotut kohdeympäristöt on tunnistettu ja konfiguroitu, ja että ne ovat testaustiimin käytettävissä. Nämä ympäristöt testataan sitten käyttämällä valikoituja toiminnallisuustestejä, jotka käyvät läpi ympäristössä olevat eri tiedonsiirtokohdat..

Yhteentoimivuus liittyy siihen, kuinka eri ohjelmistojärjestelmät toimivat toistensa kanssa. Ohjelmisto, jolla on hyvät yhteentoimivuusominaisuudet, voidaan integroida monen muun eri järjestelmän kanssa ilman, että tarvitaan suuria muutoksia. Muutosten ja niiden toteuttamiseen tarvittavan työn määrää voidaan käyttää yhteentoimivuuden mittarina.

Ohjelmiston yhteentoimivuuden testaus voi esimerkiksi keskittyä seuraaviin suunnittelun piirteisiin:

- Teollisuudenalan yleisten viestintästandardien, kuten XML:n käyttö
- Kyky tunnistaa automaattisesti muiden järjestelmien viestintätarpeet, joiden kanssa järjestelmä on vuorovaikutuksessa, ja tarpeen mukaan sopeutua niihin.

Yhteentoimivuustestaus voi olla erityisen merkittävää organisaatioille, jotka kehittävät kaupallisia valmisohjelmistoja ja työkaluja, sekä järjestelmien järjestelmiä kehittäville organisaatioille.

Tämän tyyppinen testaus tehdään komponentti-integraatio- ja järjestelmätestauksen aikana, ja se keskittyy järjestelmän vuorovaikutukseen sen ympäristön kanssa. Järjestelmäintegraatiotasolla tämän tyyppinen testaus tehdään sen määrittämiseksi, kuinka hyvin valmiiksi toteutettu järjestelmä toimii muiden järjestelmien kanssa. Koska järjestelmät voivat olla toistensa kanssa vuorovaikutuksessa monella eri tasolla, Testausasiantuntijan täytyy ymmärtää nämä vuorovaikutustilanteet ja pystyä luomaan tilanteita, jotka toteuttavat nämä erilaiset vuorovaikutussuhteet. Jos esimerkiksi kaksi järjestelmää vaihtaa keskenään tietoja, Testausasiantuntijan pitää pystyä luomaan tarpeellinen aineisto ja toimenpiteet, joita tarvitaan tietojen vaihtamiseksi. On tärkeää muistaa, että kaikkea vuorovaikutusta ei ehkä ole kuvattu tarkasti vaatimuskuvauksissa. Sen sijaan monet näistä tilanteista voi olla määritelty vain järjestelmän arkkitehtuuri- ja suunnittelukuvauksissa. Testausasiantuntijan pitää pystyä ja olla valmis käymään läpi näitä dokumentteja, jotta järjestelmien ja järjestelmän ja sen ympäristön väliset tiedonvaihtokohdat voidaan tunnistaa ja varmistaa, että kaikki tulevat testatuiksi. Päätöstaulut, tilasiirtymäkaaviot, käyttötapaukset ja kombinatorinen testaus ovat kaikki sopivia

© International Software Testing Qualifications Board

tekniikoita yhteentoimivuuden testaukseen. Tyypillisiin löydettäviin vikoihin kuuluu vääränlainen tiedonvaihto komponenttien välillä.

4.2.4 Käytettävyystestaus

On tärkeää ymmärtää, miksi käyttäjillä saattaa olla vaikeuksia järjestelmän käyttämisessä. Tämän ymmärryksen saamiseksi on ensin tarpeen käsittää, että termi ”käyttäjä” voi koskea hyvin laajaa joukkoa erityyppisiä ihmisiä, lähtien IT-asiantuntijoista lapsiin ja ihmisiin, joiden toiminta on jollain lailla rajoittunutta.

Jotkut kansalliset järjestöt (esim. British Royal National Institute for the Blind) suosittavat, että verkkosivujen pitäisi olla vammaisten, sokeiden, heikkonäköisten, liikuntarajoitteiden, kuurojen ja ymmärrykseltään rajoittuneiden henkilöiden saavutettavissa. Sen varmistaminen, että edellä mainittujen käyttäjäryhmien edustajat voivat käyttää sovelluksia ja web-sivustoja, voi myös auttaa parantamaan käytettävyyttä kaikkien muiden kohdalta. Saavutettavuudesta keskustellaan enemmän edempänä.

Käytettävyystestaus testaa, kuinka helposti käyttäjä pystyy käyttämään järjestelmää tai oppii sen käytön saavuttaakseen tietyn tavoitteen tietyssä tilanteessa. Käytettävyystestaus kohdistuu mittaamaan seuraavia ominaisuuksia:

- tehokkuus – ohjelmiston kyky luoda käyttäjille mahdollisuudet saavuttaa asetetut tavoitteet oikein ja kokonaan määrätyssä käyttötilanteessa
- toiminnallinen tehokkuus – ohjelmiston kyky luoda käyttäjille mahdollisuudet käyttää sopiva määrä resursseja suhteessa saavutettuun tehokkuuteen määrätyssä käyttötilanteessa
- tyytyväisyys – ohjelmiston kyky tehdä käyttäjät tyytyväisiksi määrätyssä käyttötilanteessa.

Mitattaviin ominaisuuksiin kuuluvat:

- ymmärrettävyys – ohjelmiston ominaisuudet, jotka vaikuttavat käyttäjältä vaadittavaan panostukseen, jotta hän tunnistaa loogisen käsitteen ja sen sovellettavuuden
- opittavuus – ohjelmiston ominaisuudet, jotka vaikuttavat käyttäjältä vaadittavaan panostukseen, jotta hän oppii järjestelmän käytön
- käyttökelpoisuus – ohjelmiston ominaisuudet, jotka vaikuttavat käyttäjältä vaadittavaan panostukseen, jotta hän voi suorittaa tehtävät sisäisesti ja ulkoisesti tehokkaasti.
- houkuttelevuus – missä määrin käyttäjät pitävät ohjelmistosta.

Käytettävyystestaus tehdään yleensä kahdessa vaiheessa:

- Formatiivinen käytettävyystestaus – testaus, jota tehdään iteratiivisesti suunnittelu- ja prototyyppivaiheen aikana ja jolla pyritään ohjaamaan suunnittelua tunnistamalla käytettävyyteen liittyviä suunnitteluvikoja.
- Summatiivinen käytettävyystestaus – testaus, joka tehdään toteutuksen jälkeen valmiin komponentin tai järjestelmän käytettävyyden mittaamiseksi ja siihen liittyvien ongelmien tunnistamiseksi.

Käytettävyystestaajan taitoihin pitäisi kuulua asiantuntemusta tai osaamista seuraavilta alueilta:

- sosiologia
- psykologia
- kansallisten standardien sisältö (mukaan luettuna esteettömyysstandardit)
- ergonomia.

4.2.4.1 Käytettävyystestien suorittaminen

Valmiin toteutuksen kelpuuttaminen pitäisi tehdä olosuhteissa, jotka ovat niin lähellä kuin mahdollista niitä olosuhteita, joissa järjestelmää tullaan käyttämään. Tämä voi tarkoittaa, että rakennetaan käytettävyyslaboratorio, jossa on videokameroita, lavastettu toimisto, seurantaikkunoita, käyttäjiä jne., jotta toteuttajat voivat seurata todellisen järjestelmän vaikutuksia oikeisiin ihmisiin. Muodollinen

© International Software Testing Qualifications Board

käytettävyystestaus vaatii usein “käyttäjien” (jotka voivat olla oikeita käyttäjiä tai käyttäjien edustajia) jonkinasteista esivalmistelua, joko antamalla heille suoritusaskelkuvaukset tai ohjeet, joita heidän pitää seurata. Muissa vapaamuotoisissa testeissä käyttäjä voi kokeilla ohjelmiston käyttöä, jotta havainnoijat voivat päätellä, kuinka helppoa tai vaikeaa käyttäjän on saada selville, kuinka suorittaa hänelle annetut tehtävät.

Testausasiantuntija voi suorittaa monet käytettävyystesteistä osana muita testejä, esimerkiksi toiminnallisen järjestelmätestauksen aikana. Käytettävyysohjeet voivat auttaa, jotta käytettävyys-vikojen löytämiseen ja raportointiin on yhtenäinen lähestymistapa kaikissa elinkaaren vaiheissa. Ilman käytettävyysohjeita voi olla vaikea päätellä, mitä on “ei-hyväksyttävä” käytettävyys. Onko esimerkiksi kohtuutonta, että käyttäjän täytyy napsauttaa hiirellä 10 kertaa, jotta hän pystyy kirjautumaan sisään sovellukseen? Ilman selkeitä ohjeita Testausasiantuntijan voi olla vaikea puolustaa vikaraportteja, jotka toteuttaja haluaa sulkea, koska ohjelmisto toimii ”kuten suunniteltu”. On hyvin tärkeää, että todennettavat käytettävyysmäärittelyt kuvataan mukaan vaatimuksiin, samoin kuin että on olemassa käytettävyysohjeet, joita sovelletaan kaikkiin samankaltaisiin projekteihin. Ohjeessa pitäisi käsitellä mm. seuraavia asioita: ohjeiden saatavuus, kehotteiden selkeys, tehtävän toteuttamiseksi tarvittavien hiiren napsautusten määrä, virheilmoitukset, toiminnan ilmaisimet (jonkinlainen ilmaisin, joka kertoo käyttäjälle, että järjestelmä käsittelee tietoa eikä voi sillä hetkellä ottaa vastaan lisää syötteitä), näyttöjen ulkoasun ohjeistus, värien ja äänten käyttö sekä muut tekijät, jotka vaikuttavat käyttäjän kokemukseen.

4.2.4.2 Käytettävyystestien määrittely

Käytettävyystestauksen pääasialliset tekniikat ovat:

- Tarkastus, arviointi ja katselmointi
- Prototyyppien dynaaminen käyttäminen
- Varsinaisen toteutuksen todentaminen ja kelpuutus
- Tutkimusten ja kyselyiden suorittaminen

Tarkastus, arviointi vai katselmointi
Vaatimusmäärittelyiden ja suunnitteluiden tarkastus tai katselmointi käytettävyysnäkökulmasta, joka kasvattaa käyttäjien osallistumisen tasoa, voi olla kustannustehokasta, sillä sen avulla löydetään ongelmia aikaisin. Heuristista arviointia (käyttöliittymäsuunnitelmien käytettävyyden systemaattista tarkastusta) voidaan käyttää paljastamaan suunnitelmien käytettävyysongelmia, jotta niihin voitaan puuttua osana iteratiivista suunnitteluprosessia. Tällöin pieni joukko arvioijia tutkii käyttöliittymää ja arvioi, kuinka hyvin se noudattaa yleisesti tunnustettuja käytettävyysperiaatteita (”heuristiikkoja”). Katselmoinnit ovat tehokkaampia, kun käyttöliittymä on näkyvämpi. Esimerkiksi kuvaruutukaappauksia on yleensä helpompi ymmärtää ja tulkita kuin sanallista kuvausta jonkin tietyn näytön toiminnallisuudesta. Visualisointi on tärkeää dokumentaation riittävässä käytettävyyskatselmoinnissa.

Prototyyppien dynaaminen käyttäminen
Kun prototyyppejä kehitetään, Testausasiantuntijan pitäisi käyttää niitä ja auttaa toteuttajia kehittämään prototyyppiä ottamalla käyttäjiltä tulleet palautteet huomioon suunnittelussa. Tällä tavalla prototyyppejä voidaan kehittää tarkemmiksi ja käyttäjä saa realistisemman kuvan siitä, miltä lopullinen tuote tulee näyttämään ja tuntumaan.

Varsinaisen toteutuksen todentaminen ja kelpuutus
Vaatimukset määrittelevät ohjelmiston käytettävyysominaisuudet (esim. tietyn tehtävän suorittamiseksi tarvittavien hiiren napsautusten määrä), ja testitapaukset pitäisi luoda sen todentamiseksi, että nämä ominaisuudet on otettu huomioon ohjelman toteutuksessa.

Varsinaisen toteutuksen kelpuuttamiseksi tarvittavat toiminnallisen järjestelmätestauksen testit voidaan laatia käytettävyystestiskenaarioina. Nämä skenaariot mittaavat määrättyjä käytettävyyden ominaisuuksia, kuten opittavuutta tai käyttökelpoisuutta ennemmin kuin toiminnallisia tuloksia.

© International Software Testing Qualifications Board

Käytettävyyden testiskenaariot voidaan laatia testaamaan erityisesti syntaksia ja semantiikkaa. Syntaksi tarkoittaa liittymän rakennetta tai kielioppia (esim. mitä voidaan syöttää syötekenttään) kun taas semantiikka kuvaa liittymän merkityksen ja käyttötarkoituksen (esim. käyttäjälle annettujen järjestelmäviestien ja tulosten järkevyys ja merkityksellisyys).

Mustalaatikkotekniikoita (esim. kappaleessa 3.2 kuvatut), erityisesti käyttötapauksia, jotka voidaan kuvata joko suorasanaisena tekstinä tai käyttämällä UML-notaatiota (Unified Modeling Language), hyödynnetään joskus käytettävyystestauksessa.

Käytettävyyden testiskenaarioihin tulee sisällyttää myös käyttäjän ohjeet, ajankäyttö, joka on varattava testiä edeltäviin ja sen jälkeisiin haastatteluihin ohjeiden antamiseksi ja palautteen keräämiseksi, sekä sovitut menettelytavat testi-istuntojen läpiviemiseksi. Nämä menettelytavat sisältävät kuvauksen siitä, kuinka testi suoritetaan, mikä on sen ajoitus, miten muistiinpanot ja istunnon tapahtumat kirjataan, sekä mitä haastattelu- ja kyselymenetelmiä käytetään.

Tutkimusten ja kyselyiden suorittaminen
Mittauksia ja kyselyitä voidaan käyttää, kun kerätään huomioita ja palautetta käyttäjän toiminnasta järjestelmän kanssa. Standardoidut ja julkisesti saatavilla olevat mittausmenetelmät kuten SUMI (Software Usability Measurement Inventory) ja WAMMI (Website Analysis and MeasureMent Inventory) mahdollistavat vertailun aiemmin tehtyjä käytettävyysmittauksia vastaan. Lisäksi SUMI tuottaa konkreettisia käytettävyysmittaritietoja, ja niitä voidaan käyttää osana valmistumis-/hyväksymiskriteereitä.

4.2.5 Esteettömyyden testaus

On tärkeää miettiä myös ohjelmiston esteettömyyttä sellaisten henkilöiden kannalta, joilla on erityisiä tarpeita tai rajoitteita sen käytön suhteen. Tähän kuuluvat myös eri tavoin vammautuneet henkilöt. Esteettömyyden testauksessa pitäisi ottaa huomioon siihen liittyvät standardit, kuten Web Content Accessibility Guidelines, ja lainsäädännöt, kuten Disability Discrimination Acts (Iso-Britannia, Australia) ja Section 508 (USA). Esteettömyyttä, kuten käytettävyyttäkin, pitää miettiä suunnitteluvaiheessa. Testaus tapahtuu usein integrointitasoilla ja se jatkuu läpi järjestelmätestauksen hyväksymistestaustasolle asti. Vian tunnistamiseen päädytään yleensä, kun ohjelmisto ei täytä siihen kohdistuvia säädöksiä tai sille määriteltyjä standardeja.

© International Software Testing Qualifications Board

5. Katselmoinnit - 165 min.

Avainsanat
Ei ole

Oppimistavoitteet: Katselmoinnit

5.1 Esittely
TA-5.1.1 (K2) Selittää, miksi katselmointeihin valmistautuminen on tärkeää Testausasiantuntijalle.

5.2 Tarkistuslistojen käyttö katselmoinneissa
TA-5.2.1 (K4) Analysoida käyttötapausta tai käyttöliittymää ja tunnistaa sertifikaattisisällössä kuvatun tarkistuslistan mukaisia ongelmia.
TA-5.2.2 (K4) Analysoida vaatimusmäärittelyä tai käyttäjätarinaa ja tunnistaa sertifikaattisisällössä kuvatun tarkistuslistan mukaisia ongelmia.

© International Software Testing Qualifications Board

5.1 Esittely

Onnistunut katselmointiprosessi vaatii suunnittelua, osallistumista ja jälkiseurantaa. Testausasiantuntijoiden täytyy osallistua aktiivisesti katselmointiprosessiin ja tuoda esiin ainutlaatuiset näkemyksensä. Heillä pitäisi olla muodollinen katselmointikoulutus, jotta he ymmärtäisivät paremmin kulloisenkin roolinsa katselmointiprosessissa. Kaikkien katselmointiin osallistuvien täytyy olla sitoutuneita hyvin toteutetusta katselmoinnista saataviin hyötyihin. Mikäli katselmoinnit tehdään kunnolla, ne voivat olla suurin ja kustannustehokkain yksittäinen tuotettuun laatuun myötävaikuttava tekijä.

Käytettävästä katselmointityypistä riippumatta Testausasiantuntijalle on annettava riittävästi aikaa valmistautumiseen. Tähän kuuluu vaihetuotteen katselmointiin kuluva aika, ristiviitattujen dokumenttien ja niiden yhdenmukaisuuden varmistamiseen tarvittava aika sekä aika, jota tarvitaan mahdollisten vaihetuotteesta puuttuvien asioiden tunnistamiseksi. Ilman riittävää valmistautumisaikaa Testausasiantuntijan tehtävä voi rajoittua vain jo dokumentissa olevan tiedon muokkaukseen sen sijaan, että hän voisi osallistua tehokkaaseen katselmointiin, mikä merkitsisi katselmointitiimin ajan tehokkainta käyttöä ja tuottaisi parhaimman mahdollisen palautteen. Hyvään katselmointiin kuuluu sen ymmärtäminen, mitä on kirjoitettu, puuttuvien asioiden määrittäminen ja sen todentaminen, että kuvattu tuote on yhdenmukainen jo toteutettujen tai parhaillaan toteutettavana olevien tuotteiden kanssa. Esimerkiksi integrointitestaussuunnitelmaa katselmoidessaan Testausasiantuntijan on myös pohdittava integroinnin kohteita. Millä ehdoilla ne ovat valmiita integrointiin? Onko riippuvuuksia, jotka täytyy dokumentoida? Onko saatavilla aineistoa integraatiokohtien testaamiseksi? Katselmointi ei rajoitu pelkästään katselmoitavaan vaihetuotteeseen; sen on otettava myös huomioon tuotteen vuorovaikutus järjestelmän muiden osien kanssa.

Katselmoitavan tuotteen laatija voi helposti tuntea olevansa kritisoitavana. Testausasiantuntijan tulisi lähestyä kaikkia katselmointikommentteja ajatuksella, että tavoitteena on tehdä yhteistyötä katselmoitavan materiaalin laatijan kanssa ja näin saada aikaan paras mahdollinen tuote. Tätä lähestymistapaa noudattamalla kommenteista tulee rakentavia ja ne kohdistuvat itse vaihetuotteeseen eivätkä sen laatijaan. Jos esimerkiksi lause on tulkinnanvarainen, on parempi sanoa ”En ymmärrä, mitä minun pitäisi testata, jotta saisin todennettua, että tämä vaatimus on toteutettu oikein. Voitko auttaa minua ymmärtämään tämän paremmin?” sen sijaan, että sanoisi ”Tämä vaatimus on tulkinnanvarainen eikä kukaan pysty ymmärtämään sitä”. Katselmoinneissa Testausasiantuntijan tehtävänä on varmistaa, että vaihetuotteessa kerrotut tiedot tukevat riittävästi testaustyötä. Jos tietoa ei löydy, tieto on epäselvää tai se ei ole riittävän yksityiskohtaista, kyseessä on todennäköisesti vika, joka laatijan pitää korjata. Kommentit otetaan paremmin vastaan ja kokouksesta tulee tuloksekkaampi, jos kriittisyyden sijaan lähestytään asiaa positiivisesti.

5.2 Tarkistuslistojen käyttö katselmoinneissa

Tarkistuslistoja käytetään katselmoinneissa muistuttamaan osallistujia tiettyjen seikkojen tarkistamisesta katselmoinnin aikana. Tarkistuslistat voivat myös auttaa tekemään katselmoinnista vähemmän henkilöön kohdistuvan, esim. ”Tämä on sama tarkistuslista, jota käytämme kaikissa katselmoinnissa, eikä kohteena ole pelkästään sinun vaihetuotteesi”. Tarkistuslistat voivat olla yleisluonteisia ja kaikissa katselmoinneissa käytettäviä tai ne voivat keskittyä määrättyihin laatuominaisuuksiin, alueisiin tai asiakirjatyyppeihin. Esimerkiksi yleisen tason tarkistuslista voi todentaa dokumentin yleisiä ominaisuuksia, kuten että sillä on yksilöivä tunnus, siinä ei ole ”Kesken”-merkintöjä, se on muotoiltu oikein, ja muita vastaavia asioita. Vaatimuskuvausten yksityiskohtainen tarkistuslista voi sisältää kohdat, joilla tarkistetaan termien “pitää” ja “pitäisi” oikea käyttö, varmistetaan jokaisen määritetyn vaatimuksen testattavuus ja niin edelleen. Vaatimusten muoto voi myös ilmaista käytettävän tarkistuslistan tyypin. Sanalliseen muotoon laaditulla vaatimusdokumentilla on erilaiset katselmointikriteerit kuin kaavioihin pohjautuvalla.

© International Software Testing Qualifications Board

Tarkistuslista voi myös kohdistua ohjelmoijan/arkkitehdin tai testaajan tiettyyn osaamisalueeseen. Testausasiantuntijan tapauksessa testaajan osaamisalueiden tarkistuslista olisi soveliain. Nämä tarkistuslistat voivat sisältää alempana kuvattuja kohtia.

Vaatimuksiin, käyttötapauksiin ja käyttäjätarinoihin käytetyt tarkistuslistat kohdistuvat yleensä eri alueisiin kuin ne, joita käytetään ohjelmakoodiin tai arkkitehtuuriin. Vaatimuksiin kohdistetut tarkistuslista voivat sisältää seuraavia kohtia:

- jokaisen vaatimuksen testattavuus
- jokaisen vaatimuksen hyväksymiskriteerit
- käyttötapauksen kutsurakenteen saatavuus, mikäli sopii tilanteeseen
- jokaisen vaatimuksen/käyttötapauksen/käyttäjätarinan yksilöllinen nimeäminen
- jokaisen vaatimuksen/käyttötapauksen/käyttäjätarinan versiointi
- jokaisen vaatimuksen jäljitettävyys liiketoiminnan/markkinoinnin vaatimuksista.
- vaatimusten ja käyttötapausten välinen jäljitettävyys.

Yllä mainitut ovat vain esimerkkejä. On tärkeää muistaa, että mikäli vaatimus ei ole testattava eli se on määritelty niin, että Testausasiantuntija ei voi päätellä, kuinka se pitää testata, vaatimuksessa on vika. Esimerkiksi vaatimusta ”Ohjelmiston pitäisi olla hyvin käyttäjäystävällinen” ei voi testata. Miten Testausasiantuntija voi päätellä, onko ohjelmisto käyttäjäystävällinen, saati sitten hyvin käyttäjäystävällinen? Jos sen sijaan vaatimus kuuluisi: “Ohjelmiston täytyy noudattaa käytettävyysstandardit sisältävässä dokumentissa määritettyjä käytettävyysstandardeja”, ja jos käytettävyysstandardit sisältävä dokumentti todella on olemassa, vaatimus olisi testattava. Tämä vaatimus on myös korkean tason vaatimus, koska se koskee jokaista käyttöliittymän osaa. Tässä tapauksessa tästä yhdestä vaatimuksesta voisi helposti syntyä monta yksittäistä testitapausta, jos kyse ei ole merkitykseltään vähäpätöisen sovelluksen testauksesta. Jäljitettävyys tästä vaatimuksesta tai kenties käytettävyysstandardista testitapauksiin on myös kriittistä, koska käyttöliittymämäärittelyjen mahdollisesti muuttuessa kaikki testitapaukset pitää katselmoida ja niitä on tarpeen mukaan päivitettävä.

Vaatimus ei myöskään ole testattava, mikäli testaaja ei pysty päättelemään, menikö testi läpi vai ei, tai hän ei pysty laatimaan testiä, joka voi mennä läpi tai hylkääntyä. Esimerkiksi “Järjestelmän pitää olla käytettävissä 100 % ajasta, 24 tuntia vuorokaudessa, 7 päivää viikossa, 365 (tai 366) päivää vuodessa” ei ole testattava.

.

Käyttötapausten katselmointiin käytettävä yksinkertainen tarkistuslista voi sisältää seuraavia kysymyksiä:

- Onko pääpolku (skenaario) selvästi määritelty?
- Onko vaihtoehtoiset polut (skenaariot) tunnistettu ja ovatko ne täydellisiä virheenkäsittely mukaan lukien?
- Onko käyttöliittymän viestit määritelty?
- Onko olemassa vain yksi pääpolku (näin pitäisi olla, muuten kyseessä on useampia käyttötapauksia)?
- Onko jokainen polku testattava?

Käyttöliittymän käytettävyyden liittyvä yksinkertainen tarkistuslista voi sisältää seuraavia asioita:

- Onko jokainen kenttä ja sen tarkoitus määritelty?
- Onko kaikki virheilmoitukset määritelty?
- Onko kaikki käyttäjäkehotteet määritelty ja ovatko ne yhdenmukaisia?
- Onko kenttien sarkainjärjestys määritelty?
- Onko hiiren toiminnoille vaihtoehtoisia näppäinkomentoja?
- Onko käyttäjän toiminnoille määritelty “oikopolku”-näppäinyhdistelmiä (esim. leikkaa ja liitä)?
- Onko kenttien välillä riippuvuuksia (esim. että tietyn päivämäärän pitää olla myöhäisempi kuin toisen päivämäärän)?

© International Software Testing Qualifications Board

- Onko näytöstä tehty layout-suunnitelma?
- Vastaako näytön asettelu määriteltyjä vaatimuksia?
- Onko käyttäjää varten olemassa ilmaisin, joka tulee näkyviin, kun järjestelmä prosessoi tietoa?
- Täyttääkö näyttö hiiren napsautuksille asetetut minimivaatimukset (jos ne on määritelty)?
- Seuraako navigointi käyttötapauksen tietoja käyttäjän kannalta loogisesti?
- Täyttääkö näyttö mahdolliset opittavuusvaatimukset?
- Onko käyttäjän saatavilla ohjetekstejä?
- Onko käyttäjän saatavilla vihjetekstejä?
- Pitääkö käyttäjä tätä “miellyttävänä” (subjektiivinen näkemys)?
- Onko värien käyttö yhdenmukaista muiden sovellusten ja organisaation standardien kanssa?
- Onko ääniefektejä käytetty sopivasti ja ovatko ne säädettävissä?
- Täyttääkö näyttö lokalisointivaatimukset?
- Voiko käyttäjä päätellä, mitä tehdä seuraavaksi (ymmärrettävyys) (subjektiivinen näkemys)?
- Pystyykö käyttäjä muistamaan, mitä tehdä seuraavaksi (opittavuus) (subjektiivinen näkemys)?

Ketterässä projektissa vaatimukset kuvataan yleensä käyttäjätarinoiden muodossa. Nämä tarinat edustavat pieniä todennettavan toiminnallisuuden osia. Siinä missä käyttötapaus kuvaa käyttäjän toimintoja, jotka kulkevat läpi toiminnallisuuden monia alueita, käyttäjätarina on rajatumpi ja sitä määrittää yleensä sen kehittämiseen tarvittava aika. Käyttäjätarinan tarkistuslista voi sisältää seuraavia kohtia:

- Sopiiko tarina kohdeiteraatioon/sprinttiin?
- Onko hyväksymiskriteerit määritelty ja ovatko ne testattavissa?
- Onko toiminnallisuus selkeästi määritelty?
- Onko tämän ja muiden tarinoiden välillä riippuvuuksia?
- Onko tarina priorisoitu?
- Sisältääkö tarina yhden kohteen tai toiminnallisuuden?

Mikäli tarina kuvaa uutta käyttöliittymää, on yleisen tason käyttäjätarinoiden tarkistuslistan (kuten yllä kuvattu) ja yksityiskohtaisen käyttöliittymän tarkistuslistan käyttö tietysti suotavaa.

Tarkistuslistaa voidaan räätälöidä seuraavien seikkojen perusteella:

- organisaatio (esim. otetaan huomioon organisaation politiikat, standardit, toimintatavat)
- projekti/kehitystyön kohde (esim. painopiste, tekniset standardit, riskit)
- katselmoinnin kohde (esim. koodikatselmoinnit voidaan räätälöidä tietylle ohjelmointikielelle sopivaksi).

Hyvät tarkistuslistat paljastavat ongelmia ja auttavat myös synnyttämään keskustelua muista asioista, joihin ei ehkä ole suoraan otettu kantaa tarkistuslistassa. Usean tarkistuslistan yhdistelmän käyttö on tehokas tapa varmistaa, että katselmoinnit tuottavat tuotoksen, jonka laatu on paras mahdollinen. Esimerkiksi Perustason sertifikaattisisällössä kuvattujen vakiotarkistuslistojen käyttäminen sekä esimerkiksi yllä mainitun tyyppisten organisaatiokohtaisten tarkistuslistojen kehittäminen auttavat Testausasiantuntijaa toimimaan katselmoinneissa tehokkaasti.

Lisätietoja katselmoinneista ja tarkastuksista, ks. [Gilb03] ja [Wiegers03].

© International Software Testing Qualifications Board

6. Vianhallinta – 120 min.

Avainsanat
perussyyanalyysi, vaiheen vikarajaustehokkuus, vikaluokitusjärjestelmä

Oppimistavoitteet: Vianhallinta

6.2 Milloin vika voidaan havaita?
TA-6.2.1 (K2) Selittää, kuinka vaiheen vikarajaustehokkuus voi vähentää kustannuksia.

6.3 Vikaraportin kentät
TA-6.3.1 (K2) Selittää, mitä tietoja voidaan tarvita, kun ollaan raportoimassa ei-toiminnallista vikaa.

6.4 Vikojen luokittelu
TA-6.4.1 (K4) Tunnistaa, kerätä ja tallentaa määrättyyn vikaan liittyvää luokittelutietoa.

6.5 Perussyyanalyysi
TA-6.5.1 (K2) Selittää perussyyanalyysin tarkoitus.

© International Software Testing Qualifications Board

6.1 Esittely

Testausasiantuntija arvioi järjestelmän käyttäytymistä liiketoiminnan ja käyttäjän tarpeiden kannalta, esim. tietääkö käyttäjä, mitä hänen pitää tehdä, kun ohjelma antaa tietynlaisen viestin tai käyttäytyy tietyllä tavalla. Vertaamalla todellista tulosta odotettuun Testausasiantuntija päättelee, käyttäytyykö ohjelma oikein. Poikkeama (kutsutaan myös havainnoksi) on odottamaton tapahtuma, joka vaatii lisäselvitystä. Poikkeama voi olla vian aiheuttama häiriö. Poikkeama voi johtaa vikaraportin laatimiseen, mutta ei aina. Vika on todellinen ongelma, joka pitäisi selvittää.

6.2 Milloin vika voidaan havaita?

Vika voidaan havaita staattisessa testauksessa ja vian oireet eli häiriö voidaan löytää dynaamisen testauksen avulla. Jokaisessa ohjelmistokehityksen elinkaaren vaiheessa pitäisi olla olemassa menetelmät mahdollisten häiriöiden löytämiseksi ja poistamiseksi. Esimerkiksi toteutusvaiheessa pitäisi käyttää koodin ja suunnitelmien katselmointeja vikojen löytämiseksi. Dynaamisen testauksen aikana testitapauksia käytetään häiriöiden löytämiseen.

Mitä aikaisemmin vika löydetään ja korjataan, sitä matalammat järjestelmän laatukustannukset kokonaisuudessaan ovat. Esimerkiksi staattinen testaus voi paljastaa vikoja ennen kuin dynaaminen testaus on mahdollista. Tämä on yksi syy siihen, miksi staattinen testaus on kustannustehokas lähestymistapa korkealaatuisen ohjelmiston tuottamiseen.

Vianhallintajärjestelmän pitäisi antaa Testausasiantuntijalle mahdollisuus kirjata ylös elinkaaren vaihe, jossa vika syntyi, ja vaihe, jossa se löydettiin. Jos nämä kaksi vaihetta ovat samat, on saavutettu täydellinen vaiheen vikarajaustehokkuus. Tämä tarkoittaa, että vika syntyi ja löydettiin saman vaiheen aikana eikä se “karannut” myöhempään vaiheeseen. Esimerkki tästä on virheellinen vaatimus, joka tunnistetaan vaatimuskatselmoinnissa ja korjataan siellä. Tämä ei ole pelkästään tehokasta vaatimuskatselmointien käyttöä, vaan se estää myös vikaa aiheuttamasta lisätyötä, mikä olisi paljon kalliimpaa organisaatiolle. Jos virheellinen vaatimus “karkaa” vaatimuskatselmoinnista ja päätyy toteuttajan toteuttamaksi, Testausasiantuntijan testaamaksi, ja jää kiinni vasta käyttäjän suorittamassa hyväksymistestauksessa, kaikki kyseiseen vaatimukseen käytetty työ oli turhaa (puhumattakaan siitä, että käyttäjä on ehkä menettänyt luottamuksensa järjestelmään).

Vaiheen vikarajaustehokkuus on tehokas tapa vähentää vikojen aiheuttamia kustannuksia.

6.3 Vikaraportin kentät

Vikaraportin eri kenttien (parametrien) tarkoituksena on tuottaa riittävästi tietoa, jotta raportin pohjalta voidaan ryhtyä toimenpiteisiin. Toimenpidekelpoinen raportti on

- täydellinen – kaikki tarvittava tieto löytyy raportista
- tiivis – raportissa ei ole epäoleellista tietoa
- tarkka – raportin sisältämä tieto on oikein ja ilmaisee selkeästi odotetun ja todellisen tuloksen sekä tarvittavat askeleet tilanteen toistamiseksi
- objektiivinen – raportti on ammattimaisesti kirjoitettu tosiasioiden kuvaus.

Raporttiin kirjoitettava tieto pitäisi jakaa tietokenttiin. Mitä tarkemmin kentät on määritelty, sitä helpompaa on yksittäisten vikojen raportointi sekä trendiraporttien ja muiden yhteenvetoraporttien tuottaminen. Mikäli kenttään voidaan syöttää arvo määrättyjen vaihtoehtojen joukosta, käytettävissä olevat arvot sisältävän alasvetovalikon käyttäminen voi vähentää vian raportointiin tarvittavaa aikaa. Alasvetovalikot ovat tehokkaita vain silloin, kun vaihtoehtojen määrä on rajattu eikä käyttäjän tarvitse rullata läpi pitkän listan löytääkseen oikean vaihtoehdon. Eri tyyppiset vikaraportit vaativat erilaista tietoa ja havaintojenhallintavälineen pitäisi olla riittävän joustava, jotta se osaa pyytää tietoja oikeisiin

© International Software Testing Qualifications Board

kenttiin vikatyypin mukaan. Tiedot pitäisi raportoida määrättyihin kenttiin, joissa ihannetilanteessa myös tarkistetaan niiden oikeellisuus syöttövirheiden välttämiseksi ja tehokkaan raportoinnin varmistamiseksi.

Vikaraportteja laaditaan toiminnallisen ja ei-toiminnallisen testauksen aikana ilmenneistä häiriöistä. Vikaraportin sisältämän tiedon tarkoituksena pitäisi aina olla ongelman löytymishetkellä vallinneen tilanteen kuvaaminen, sen toistamiseksi tarvittavat askeleet ja aineisto sekä odotetut ja todelliset tulokset mukaan luettuna. Ei-toiminnallisten vikojen raportteihin voidaan tarvita enemmän yksityiskohtia ympäristöön, muihin suoritusparametreihin (esim. kuorman koko), askelten järjestykseen ja odotettuihin tuloksiin liittyen. Käytettävyyteen liittyvää häiriötä raportoitaessa on tärkeää kuvata, mitä käyttäjä odotti ohjelmiston tekevän. Jos esimerkiksi käytettävyysstandardin mukaan toimenpide pitäisi pystyä suorittamaan alle neljällä hiiren napsautuksella, vikaraportissa pitää kertoa, montako napsautusta tarvittiin verrattuna vertailtavaan standardiin. Tilanteissa, joissa standardia ei ole saatavilla ja vaatimukset eivät kata ohjelmiston ei-toiminnallisia laatuominaisuuksia, testaaja voi käyttää ”järkevän henkilön” testiä määrittämään, että käytettävyys ei ole hyväksyttävä. Tällaisessa tilanteessa ”järkevältä henkilöltä” odotetut ominaisuudet pitää selkeästi ilmaista vikaraportissa. Koska ei-toiminnalliset vaatimukset puuttuvat joskus vaatimusdokumentaatiosta, ei-toiminnallisten häiriöiden dokumentoinnissa on testaajalle enemmän haasteita, kun dokumentoidaan ”odotettua” käyttäytymistä verrattuna ”todelliseen” käyttäytymiseen.

Vaikka vikaraportin kirjoittamisen tyypillisenä tavoitteena on saada aikaan vian korjaus, vikatietoja tarvitaan myös tukemaan oikeaa luokittelua, riskianalyysiä ja prosessikehitystä.

6.4 Vikojen luokittelu

Vikaraportti voidaan luokitella monella tasolla sen elinkaaren aikana. Asianmukainen vikojen luokittelu on keskeinen osa kunnollista vikaraportointia. Luokitteluja käytetään vikojen ryhmittelyyn, testauksen tehokkuuden arviointiin, kehityselinkaaren tehokkuuden arviointiin, ja mielenkiintoisten suuntausten määrittelyyn.

Vastalöydetyn vian tyypillisiin luokittelutietoihin kuuluvat:

- projektin tehtävä, jolloin vika löydettiin – esim. katselmointi, auditointi, tarkastus, toteutus, testaus.
- projektin vaihe, jolloin vika syntyi (jos tiedetään) – esim. vaatimukset, suunnittelu, tekninen suunnittelu, toteutus
- projektin vaihe, jolloin vika löydettiin – esim. vaatimukset, suunnittelu, tekninen suunnittelu, toteutus, koodikatselmointi, yksikkötestaus, integrointitestaus, järjestelmätestaus, hyväksymistestaus
- arvioitu vian syy – esim. vaatimukset, suunnittelu, rajapinta, toteutus, aineisto
- toistettavuus – esim. kerran, epäsäännöllisesti, toistettava
- oireet – esim. kaatuminen, jumiutuminen, käyttöliittymävirhe, järjestelmävirhe, suorituskyky.

Tutkimisen jälkeen vian jatkoluokittelu voi olla mahdollista:

- alkuperäissyy – virhe, joka tehtiin ja jonka tuloksena vika syntyi, esim. prosessi, toteutusvirhe, käyttäjän virhe, testausvirhe, kokoonpanoon liittyvä virhe, aineistoon liittyvä virhe, alihankkijan tuottama ohjelmisto, ulkoinen ohjelmisto-ongelma, dokumentaatiovirhe
- lähde – vaihetuote, jossa virhe tehtiin, esim. vaatimukset, suunnittelu, tekninen suunnittelu, arkkitehtuuri, tietokantasuunnittelu, käyttäjän dokumentaatio, testidokumentaatio
- tyyppi – esim. logiikkaongelma, laskennallinen ongelma, ajoitusongelma, aineiston käsittely, lisäys.

Lisää luokittelutietoja voi olla saatavilla sen jälkeen, kun vika on korjattu (tai korjausta on lykätty tai vikaa ei ole saatu vahvistettua), kuten:

© International Software Testing Qualifications Board

- lopputulos – esim. koodimuutos, dokumentaation muutos, lykätty, ei vika, kaksoiskappale
- korjaavat toimenpiteet – esim. vaatimuskatselmointi, koodikatselmointi, yksikkötestaus, kokoonpanon dokumentaatio, aineiston valmistelu, ei muutoksia.

Näiden luokittelujen lisäksi vikoja luokitellaan säännöllisesti niiden vakavuuden ja kiireellisyyden mukaan. Lisäksi, projektista riippuen, voi olla järkevää luokitella vikoja vian turvallisuusvaikutuksen tai sen projektin aikatauluun, kustannuksiin, riskeihin tai laatuun aiheuttaman vaikutuksen mukaan. Näitä luokitteluja voidaan käyttää tukena, kun sovitaan siitä, kuinka nopeasti korjaus toteutetaan.

Luokittelun viimeinen kohta on lopputulos. Viat luokitellaan usein yhteen niiden lopputuloksen mukaan, esim. korjattu/verifioitu, suljettu/ei vika, lykätty, avoin/selvittämätön. Tätä luokittelua käytetään yleensä läpi projektin, kun seurataan vikojen etenemistä niiden elinkaaren läpi.

Luokittelussa käytetyt arvot ovat usein organisaatiolle räätälöityjä. Yllä esitetyt ovat vain esimerkkejä joistakin alalla tyypillisesti käytetyistä arvoista. Jotta luokitteluarvot olisivat hyödyllisiä, on tärkeää, että niitä käytetään yhdenmukaisesti. Liian monta luokittelukenttää tekee vian avaamisesta ja käsittelystä aikaavievää, joten on tärkeää miettiä kerättävien tietojen merkitystä suhteessa jokaisen käsitellyn vian aiheuttamiin kustannuksiin. Työkalulla kerättävien luokittelutietojen räätälöintimahdollisuus on tärkeä tekijä työkalua valittaessa.

6.5 Perussyyanalyysi

Perussyyanalyysin tarkoitus on määrittää, mikä aiheutti vian syntymisen, ja tuottaa aineistoa, joka auttaa muuttamaan prosesseja niin, että suhteellisesti merkittäviä vikamääriä aiheuttavat alkuperäissyyt saadaan poistettua. Perussyyanalyysin tekee yleensä henkilö, joka tutkii ja joko korjaa ongelman tai päättää, että ongelmaa ei pidä tai ei voi korjata. Tämä henkilö on yleensä toteuttaja. Alustavan perussyyanalyysin suorittaa tavallisesti Testausasiantuntija, joka tekee ”valistuneen arvauksen” siitä, mikä on ongelman aiheuttaja. Varmistaessaan korjauksen toimivuutta Testausasiantuntija todentaa myös toteuttajan ilmoittaman alkuperäissyyn taustan. Alkuperäissyyn määrittelyn yhteydessä on myös tavallista päätellä tai varmistaa vaihe, jolloin kyseinen vika syntyi.

Tyypillisiin alkuperäissyihin kuuluvat seuraavat:

- epäselvät vaatimukset
- puuttuvat vaatimukset
- väärät vaatimukset
- vääränlainen suunnittelun toteutus
- vääränlainen liittymien toteutus
- logiikkavirheet koodissa
- laskentavirheet
- laitteistovirheet
- rajapintavirheet
- epäkelpo aineisto.

Tiedot alkuperäissyistä kootaan yhteen sen määrittämiseksi, onko olemassa yleisiä tekijöitä, jotka johtavat vikojen syntyyn. Jos esimerkiksi monien vikojen syynä ovat epäselvät vaatimukset, on järkevää panostaa enemmän tehokkaisiin vaatimusten katselmointeihin. Samoin, jos rajapintojen toteutus aiheuttaa ongelmia toteuttajaryhmien välillä, voidaan tarvita yhteisiä suunnittelupalavereja.

Perussyyanalyysistä saatujen tietojen käyttäminen prosessien kehittämiseen auttaa organisaatiota seuraamaan tehokkaiden prosessimuutosten hyötyjä ja laskemaan sellaisten vikojen kustannukset, joiden voidaan päätellä olevan peräisin jostain tietystä alkuperäissyystä. Tämä voi auttaa rahoituksen hankkimista sellaisia prosessimuutoksia varten, jotka voivat vaatia lisätyökalujen ja laitteiden

© International Software Testing Qualifications Board

hankkimista sekä aikataulumuutoksia. ISTQB Asiantuntijatason ”Improving the Test Process” –sertifikaattisisällössä [ISTQB_EL_ITP] käsitellään perussyyanalyysiä tarkemmin.

© International Software Testing Qualifications Board

7. Testaustyökalut - 45 min.

Avainsanat

avainsanaohjattu testaus, testiaineiston valmisteluväline, testin suoritustyökalu, testisuunnittelutyökalu

Oppimistavoitteet: Testaustyökalut

7.2 Testaustyökalut ja automaatio

TA-7.2.1 (K2) Selittää testiaineiston valmistelutyökalujen, testisuunnittelutyökalujen ja testin suoritustyökalujen käyttämisen hyödyt.
TA-7.2.2 (K2) Selittää Testausasiantuntijan rooli avainsana-ohjatussa automaatiossa.
TA-7.2.3 (K2) Selittää automatisoidun testisuorituksen häiriön selvittämistoimenpiteet.

© International Software Testing Qualifications Board

7.1 Esittely

Testaustyökalut voivat parantaa merkittävästi testauksen tehokkuutta ja tarkkuutta, mutta vain silloin, jos oikeita työkaluja käytetään oikealla tavalla. Testaustyökaluja täytyy hallinnoida osana hyvin johdettua testausorganisaatiota. Testaustyökalujen monimutkaisuus ja soveltuvuus käyttöön vaihtelevat laajasti, ja työkalumarkkinat muuttuvat jatkuvasti. Työkaluja on yleensä saatavilla sekä kaupallisilta työkalutoimittajilta samoin kuin useilta shareware- tai freeware-sivustoilta.

7.2 Testaustyökalut ja automaatio

Suuri osa Testausasiantuntijan työstä vaatii työkalujen tehokasta käyttöä. Tieto siitä, mitä työkalua käytetään milloinkin, voi parantaa Testausasiantuntijan tehokkuutta ja voi auttaa paremman testikattavuuden saavuttamiseen käytettävissä olevan ajan puitteissa.

7.2.1 Testisuunnittelutyökalut

Testisuunnittelutyökaluja käytetään testauksessa käytettävien testitapausten ja testiaineiston luomisessa. Nämä työkalut voivat käyttää pohjana määrämuotoisia vaatimusdokumentteja, malleja (esim. UML) tai Testausasiantuntijan tuottamia syötteitä. Testisuunnittelutyökalut on usein suunniteltu ja rakennettu toimimaan tietyssä muodossa olevan materiaalin ja tiettyjen tuotteiden, kuten määrätyn vaatimustenhallintatyökalun kanssa.

Testisuunnittelutyökalu voi tuottaa Testausasiantuntijalle tietoa, jota hän voi käyttää määritellessään, minkä tyyppisiä testejä tarvitaan tavoitellun testikattavuuden tai järjestelmältä odotetun luottamustason saavuttamiseksi tai tuoteriskien pienentämistoimenpiteitä varten. Esimerkiksi luokittelupuutyökalu luo (ja näyttää) joukon yhdistelmiä, jotka tarvitaan valittujen kattavuuskriteerien perusteella halutun kattavuuden saavuttamiseksi. Testausasiantuntija voi käyttää tätä tietoa määrittäessään, mitkä testitapaukset täytyy suorittaa.

7.2.2 Testiaineiston valmisteluvälineet

Testiaineiston valmistelutyökalut tarjoavat useita etuja. Jotkut näistä työkaluista pystyvät analysoimaan dokumentteja, kuten esimerkiksi vaatimuskuvausta tai jopa lähdekoodia, ja päättelemään, minkälaista aineistoa testauksen aikana tarvitaan kattavuustason saavuttamiseksi. Toiset testiaineiston valmistelutyökalut voivat poimia tuotantojärjestelmästä aineistojoukon ja ”puhdistaa” tai ”anonymisoida” sen kaiken henkilökohtaisen tiedon poistamiseksi rikkomatta silti aineistojoukon sisällön yhtenäisyyttä. Näin käsiteltyä aineistoa voidaan sen jälkeen käyttää testauksessa ilman tietoturvavuotojen tai henkilökohtaisten tietojen väärinkäytön riskiä. Tämä on erityisen tärkeää, kun tarvitaan suuria määriä todenmukaista aineistoa. Toisia aineistonluontityökaluja voidaan käyttää aineiston luomiseksi määrättyjen syötearvojoukkojen perusteella (eli satunnaistestausta varten). Jotkut työkalut analysoivat tietokantarakennetta sen määrittämiseksi, minkälaisia syötteitä Testausasiantuntijalta tarvitaan.

7.2.3 Testien suorituksen automatisointityökalut

Testien suoritustyökaluja käyttää tyypillisesti Testausasiantuntija kaikilla testaustasoilla testien suoritukseen ja testien tulosten tarkastamiseen. Suoritustyökaluun liittyy tyypillisesti yksi tai useampia seuraavista tavoitteista:

- kustannusten vähentäminen (työmäärän ja/tai ajan)
- suuremman testimäärän suorittaminen
- samojen testien suorittaminen monissa ympäristöissä
- testien suorittamisen toistettavuuden lisääminen

© International Software Testing Qualifications Board

- sellaisten testien suorittaminen, joiden suoritus manuaalisesti olisi mahdotonta (eli suuren aineistomäärän validointitestit)

Nämä tavoitteet ovat usein päällekkäisiä päätavoitteena olevan testikattavuuden kasvattamisen ja samanaikaisen kustannusten pienentämisen kanssa.

7.2.3.1 Sovellettavuus

Testien suoritustyökaluihin sijoitetun investoinnin tuotto on tavallisesti korkein silloin, kun automatisoidaan regressiotestejä, koska niiden ylläpidon odotetaan olevan vähäistä ja niitä toistetaan usein. Aloitustestien automatisointi voi olla myös osa tehokasta automatisoinnin käyttöä, koska testejä käytetään usein, niiden tulokset tarvitaan nopeasti, ja koska mahdollisista korkeammista ylläpitokustannuksista huolimatta tarjolla on automatisoitu keino uuden koonnin arvioimiseksi jatkuvan integroinnin ympäristössä.

Testien suoritustyökaluja käytetään tyypillisesti järjestelmä- ja integrointitestaustasoilla. Joitain työkaluja, erityisesti API-testaustyökaluja, voidaan käyttää myös yksikkötestaustasolla. Työkalujen käyttöön panostaminen alueilla, joilla ne ovat kaikkein soveltuvimpia, auttaa parantamaan sijoitukselle saatavaa tuottoa.

7.2.3.2 Testiautomaatiotyökalujen perusteet

Testien suoritustyökalut toimivat suorittamalla joukon toimintaohjeita, jotka on kirjoitettu ohjelmointikielellä, jota kutsutaan myös usein skriptikieleksi. Työkalulle annettavat ohjeet ovat hyvin yksityiskohtaisia ja ne kuvaavat syötteet, syötteiden järjestyksen, syötteissä käytettävät tietyt arvot ja odotetut lopputulokset. Tämä voi tehdä yksityiskohtaisista skripteistä hyvin herkän testattavaan ohjelmistoon tehtyjen muutosten suhteen, erityisesti silloin, kun työkalu on tekemisissä graafisen käyttöliittymän kanssa.

Useimmat suoritustyökalut sisältävät vertailijan, joka mahdollistaa todellisten tulosten vertaamisen tallennettuihin odotettuihin tuloksiin.

7.2.3.3 Testiautomaation käyttöönotto

Testien suorituksen automatisoinnissa (samoin kuin ohjelmoinnissa) on taipumus siirtyä yksityiskohtaisista matalan tason ohjeista korkeamman tason kieliin ja kirjastojen, makrojen ja aliohjelmien hyödyntämiseen. Avainsanaohjatun ja toimisanaohjatun suunnittelutekniikan kaltaiset tekniikat tallentavat joukon ohjeita ja viittaavat niihin erityisillä ”avainsanoilla” tai ”toimisanoilla”. Tällöin Testausasiantuntija voi kirjoittaa testitapaukset luonnollisella kielellä ja jättää taustalla olevan ohjelmointikielen ja alemman tason toiminnot huomiotta. Tällaisen modulaarisen kirjoitustekniikan käyttäminen helpottaa testattavan ohjelmiston ylläpitoa, kun tehdään muutoksia sen toiminnallisuuteen ja liittymiin. [Bath08] Avainsanojen käyttöä automatisoiduissa skripteissä kuvataan tarkemmin edempänä.

Avainsanojen tai toimisanojen luomista ohjaamaan voidaan käyttää malleja. Tutkimalla liiketoimintamalleja, jotka on usein sisällytetty vaatimusdokumentaatioon, Testausasiantuntija voi päätellä keskeiset liiketoimintaprosessit, jotka täytyy testata. Tämän jälkeen voidaan määritellä näiden prosessien askeleet, mukaan luettuna prosessin aikana esiintyvät päätöksentekokohdat. Päätöksentekokohdista voi tulla toimisanoja, joita testiautomaatio voi hakea avainsana- tai toimisanataulukoista ja sitten käyttää niitä. Liiketoimintaprosessien mallinnus on menetelmä liiketoimintaprosessin dokumentoimiseksi ja sitä voidaan käyttää näiden avainprosessien päätöksentekokohtien tunnistamiseen. Mallinnus voidaan tehdä manuaalisesti tai käyttämällä työkaluja, jotka toimivat liiketoimintasääntöihin ja prosessikuvauksiin perustuvien syötteiden pohjalta.

7.2.3.4 Automatisoinnin menestyksen parantaminen

Jokainen ehdolla oleva testitapaus tai testijoukko on arvioitava sen suhteen, onko se automatisoinnin arvoinen, kun ollaan päättämässä, mitä testejä automatisoidaan. Monet epäonnistuneet automaatioprojektit ovat perustuneet siihen, että on automatisoitu olemassa olleita manuaalisia testitapauksia ilman, että on tarkistettu niiden automatisoinnin todellinen hyöty. Tietyssä tilanteessa

© International Software Testing Qualifications Board

optimaalinen testijoukko (testisetti) saattaa sisältää manuaalisia, puoliautomaattisia ja täysin automatisoituja testejä.

Seuraavia seikkoja pitäisi miettiä, kun ollaan toteuttamassa testien suorituksen automatisointiprojektia:

Mahdolliset hyödyt:

- Automatisoitujen testien suoritukseen kuluva aika on helpommin ennakoitavissa.
- Regressiotestaus ja automatisoitujen testien avulla tehtävä vikojen kelpuuttaminen on nopeampaa ja luotettavampaa projektin myöhemmissä vaiheissa.
- Automatisointityökalujen käyttö voi lisätä testaajan tai testaustiimin arvostusta ja teknistä kasvua.
- Automatisoinnista voi olla erityisesti hyötyä iteratiivisissa ja inkrementaalisissa kehitysmalleissa jokaisen koonnin tai iteraation regressiotestauksen parantamisessa.
- Tiettyjen testityyppien kattaminen voi olla mahdollista vain automatisointityökalujen avulla (esim. suurten aineistojen kelpuutustehtävät).
- Testien suorituksen automatisointi voi olla kustannustehokkaampaa kuin manuaalinen testaus, kun on kyse suurien aineistojen syöttämisestä, konversiosta ja vertailusta, sillä automatisointi tarjoaa nopean ja yhdenmukaisen tietojen syötön ja todentamisen.

Mahdolliset riskit:

- Puutteellinen, tehoton tai vääränlainen manuaalinen testaus saatetaan automatisoida “sellaisena kuin se on”.
- Testimateriaalin ylläpito voi olla vaikeaa, sillä siihen voi tulla useita muutoksia testattavan ohjelmiston muuttuessa.
- Testaajien välitön mukanaolo testien suorituksessa voi vähentyä, mikä saattaa johtaa siihen, että löydetään vähemmän virheitä.
- Testaustiimillä ei ehkä ole riittävästi taitoja automatisointityökalujen tehokkaaseen käyttöön.
- Automatisoiduksi saattaa tulla myös epäoleellisia testejä vain siksi, että ne ovat olemassa ja ne ovat vakaita, vaikka ne eivät lisää testauksen yleistä kattavuutta.
- Testeistä saattaa tulla hyödyttömiä, kun ohjelma vakautuu (hyönteismyrkkyparadoksi).

Testauksen suoritusautomaatiovälineen käyttöönoton aikana ei ole aina järkevää automatisoida manuaalisia testejä sellaisenaan, vaan on parempi määritellä testit uudelleen, jotta ne soveltuvat paremmin automatisoitaviksi. Tähän kuuluu testitapausten uudelleenmuotoilu, uudelleenkäyttötapojen miettiminen, syötteiden käytön laajentaminen käyttämällä muuttujia kovakoodattujen arvojen sijaan ja testityökalun kaikkien ominaisuuksien täysi hyödyntäminen. Testien suoritustyökalut pystyvät yleensä käymään läpi useita testejä, ryhmittelemään niitä, toistamaan testejä ja muuttamaan niiden suoritusjärjestystä samalla kun ne tarjoavat myös analysointiin ja raportointiin tarvittavat ominaisuudet.

Monien testien suoritustyökalujen kohdalla tarvitaan ohjelmointitaitoja tehokkaiden ja toiminnallisesti pätevien testien (skriptien) ja testijoukkojen luomiseksi. On tavallista, että suurien automatisoitujen testijoukkojen ylläpitäminen ja hallinnointi on hyvin vaikeaa, jos niitä ei ole suunniteltu huolellisesti. Testityökaluun, ohjelmointiin ja suunnittelutekniikoihin liittyvä oikeanlainen koulutus on tärkeää, kun halutaan varmistaa, että työkalusta saatava täysi hyöty jalkautetaan tehokkaasti.

Testisuunnittelun aikana on tärkeää varata aikaa siihen, että automatisoidut testitapaukset suoritetaan aika ajoin manuaalisesti, jotta palautetaan mieleen, kuinka testi toimii, ja voidaan varmistaa testin oikeanlainen toiminta ja katselmoida syöteaineiston kelvollisuus ja kattavuus.

7.2.3.5 Avainsana-ohjattu automaatio

Avainsanoja (kutsutaan joskus myös toimisanoiksi) käytetään enimmäkseen, mutta ei pelkästään, kuvaamaan järjestelmän avulla suoritettavia ylemmän tason liiketoimintatehtäviä (esim. ”tilauksen peruutus”). Jokaista avainsanaa käytetään tyypillisesti edustamaan useita toimijan ja testattavan

© International Software Testing Qualifications Board

järjestelmän välisiä toimenpiteitä. Avainsanojen sarjoja (joihin kuuluu myös tarvittava testiaineisto) käytetään testitapausten määrittelyssä. [Buwalda01].

Testiautomaatiossa avainsana toteutetaan yhtenä tai useampana suoritettavana testiskriptinä. Työkalut lukevat testitapauksia, jotka muodostuvat peräkkäisistä avainsanoista, jotka kutsuvat avainsanojen toiminnallisuuden toteuttavia testiskriptejä. Skriptit toteutetaan hyvin modulaarisesti, jotta niiden liittäminen määrättyihin avainsanoihin on helppoa. Tällaisten modulaaristen skriptien toteuttamiseen tarvitaan ohjelmointitaitoja.

Avainsana-ohjatun testiautomaation pääasiallisia etuja ovat seuraavat:

- Liiketoiminta-alueen asiantuntijat voivat määritellä avainsanoja, jotka liittyvät johonkin tiettyyn sovellukseen tai liiketoiminnan alueeseen. Tämä voi tehdä testitapausten määrittelystä tehokkaampaa.
- Henkilö, jolla on pääasiassa toimialueosaamista, voi hyötyä automatisoiduista testitapausten suorituksesta (kunhan avainsanat on toteutettu skripteiksi) ilman, että hänen täytyy ymmärtää taustalla olevaa automaatiokoodia.
- Avainsanoja käyttämällä kirjoitettuja testitapauksia on helpompi ylläpitää, koska niiden muuttamisen tarve on epätodennäköisempää, vaikka testattavan ohjelmiston yksityiskohdat muuttuvat.
- Testisuunnitelmat ovat riippumattomia niiden toteutuksesta. Avainsanojen toteutukseen on käytettävissä useita erilaista skriptikieliä ja työkaluja.

Tekninen testausasiantuntija laatii yleensä automaatioskriptit (varsinaisen automaatiokoodin), joka käyttää avainsanojen/toimisanojen tietoja, kun taas Testausasiantuntija yleensä laatii ja ylläpitää avainsana-/toimisanamateriaalin. Avainsanaohjattu automaatio suoritetaan yleensä järjestelmätestausvaiheessa, mutta koodin toteuttamien voi alkaa jopa niin varhain kuin integrointivaiheessa. Iteratiivisessa ympäristössä testiautomaation kehittäminen on jatkuva prosessi.

Kun syöteavainsanat ja aineisto on luotu, Testausasiantuntija ottaa yleensä vastuulleen avainsanaohjattujen testitapausten suorittamisen ja mahdollisesti esiin tulleiden häiriöiden analysoinnin. Kun eteen tulee poikkeustilanne, Testausasiantuntijan on tutkittava häiriön syy sen määrittämiseksi, onko ongelma avainsanoissa, syöteaineistossa, itse automaatioskriptissä vai testattavassa sovelluksessa. Yleensä ongelman selvittämisen ensimmäinen askel on saman testin suorittaminen manuaalisesti samalla aineistolla sen tarkistamiseksi, onko häiriö itse sovelluksessa. Jos häiriö ei tule tällöin esiin, Testausasiantuntijan pitää katselmoida häiriötilanteeseen johtaneiden testien sarja sen selvittämiseksi, syntyikö ongelma aikaisemmassa vaiheessa (kenties laadittiin vääränlainen aineisto), mutta ei tullut esiin ennen kuin myöhemmin prosessissa. Mikäli Testausasiantuntija ei pysty määrittelemään häiriön syytä, ongelmanratkaisutiedot pitäisi luovuttaa Tekniselle testausasiantuntijalle tai toteuttajalle jatkoanalyysiä varten.

7.2.3.6 Syitä automaation epäonnistumiseen

Testien suoritusautomaatioprojektit epäonnistuvat usein niille asetettujen tavoitteiden saavuttamisessa. Nämä epäonnistumiset voivat johtua riittämättömästä joustavuudesta testausvälineen käytössä, testaustiimin riittämättömistä ohjelmointitaidoista tai epärealistisista odotuksista sen suhteen, minkälaisia ongelmia testien suoritusautomaatiolla voidaan ratkaista. On tärkeää huomata, että kaikki testien suoritusautomaatio vaatii hallintaa, panostusta, osaamista ja seurantaa, aivan kuten mikä tahansa ohjelmistokehitysprojekti. Pitkäaikaisen arkkitehtuurin luomiseen ja sitä seuraavien kunnollisten suunnittelukäytäntöjen, kokoonpanonhallinnan ja hyvien ohjelmointikäytäntöjen laatimiseen täytyy varata aikaa. Automatisoidut testiskriptit pitää testata, koska niissä on todennäköisesti virheitä. Skriptejä täytyy ehkä mukauttaa suorituskyvyn varmistamiseksi. Työkalun käytettävyys täytyy ottaa huomioon, ei pelkästään toteuttajan näkökulmasta vaan myös niiden ihmisten, jotka tulevat käyttämään työkalua skriptien suorittamiseksi. Voi olla tarpeen suunnitella työkalun ja käyttäjän väliin liittymä, joka mahdollista pääsyn käsiksi testitapauksiin tavalla, joka on testaajan kannalta looginen, mutta joka silti tarjoaa työkalulle sen tarvitseman pääsyn käsiksi tietoihin.

© International Software Testing Qualifications Board

8. Viitteet

8.1 Standardit

- [ISO25000] ISO/IEC 25000:2005, Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE)
  Luvut 1 ja 4
- [ISO9126] ISO/IEC 9126-1:2001, Software Engineering - Software Product Quality,
  Luvut 1 ja 4
- [RTCA DO-178B/ED-12B]: Software Considerations in Airborne Systems and Equipment Certification, RTCA/EUROCAE ED12B.1992.
  Luku 1

8.2 ISTQB dokumentit

- [ISTQB_AL_OVIEW] ISTQB Advanced Level Overview, Version 1.0
- [ISTQB_ALTM_SYL] ISTQB Advanced Level Test Manager Syllabus, Version 1.0
- [ISTQB_ALTTA_SYL] ISTQB Advanced Level Technical Test Analyst Syllabus, Version 1.0
- [ISTQB_FL_SYL] ISTQB Foundation Level Syllabus, Version 2011
- [ISTQB_GLOSSARY] Standard glossary of terms used in Software Testing, Version 2.2, 2012

8.3 Kirjat

[Bath08] Graham Bath, Judy McKay, “The Software Test Engineer’s Handbook”, Rocky Nook, 2008, ISBN 978-1-933952-24-6

[Beizer95] Boris Beizer, “Black-box Testing”, John Wiley & Sons, 1995, ISBN 0-471-12094-4

[Black02]: Rex Black, “Managing the Testing Process (2nd edition)”, John Wiley & Sons: New York, 2002, ISBN 0-471-22398-0

[Black07]: Rex Black, “Pragmatic Software Testing”, John Wiley and Sons, 2007, ISBN 978-0-470-12790-2

[Buwalda01]: Hans Buwalda, “Integrated Test Design and Automation”, Addison-Wesley Longman, 2001, ISBN 0-201-73725-6

[Cohn04]: Mike Cohn, “User Stories Applied: For Agile Software Development”, Addison-Wesley Professional, 2004, ISBN 0-321-20568-5

[Copeland03]: Lee Copeland, “A Practitioner's Guide to Software Test Design”, Artech House, 2003, ISBN 1-58053-791-X

[Craig02]: Rick David Craig, Stefan P. Jaskiel, “Systematic Software Testing”, Artech House, 2002, ISBN 1-580-53508-9

[Gerrard02]: Paul Gerrard, Neil Thompson, “Risk-based e-business Testing”, Artech House, 2002, ISBN 1-580-53314-0

[Gilb93]: Tom Gilb, Graham Dorothy, “Software Inspection”, Addison-Wesley, 1993, ISBN 0-201-63181-4

[Graham07]: Dorothy Graham, Erik van Veenendaal, Isabel Evans, Rex Black “Foundations of Software Testing”, Thomson Learning, 2007, ISBN 978-1-84480-355-2

© International Software Testing Qualifications Board

[Grochmann94]: M. Grochmann (1994), Test case design using Classification Trees, in: conference proceedings STAR 1994

[Koomen06]: Tim Koomen, Leo van der Aalst, Bart Broekman, Michiel Vroon "TMap NEXT, for result driven testing", UTN Publishers, 2006, ISBN 90-72194-80-2

[Myers79]: Glenford J. Myers, “The Art of Software Testing”, John Wiley & Sons, 1979, ISBN 0-471-46912-2

[Splaine01]: Steven Splaine, Stefan P. Jaskiel, “The Web-Testing Handbook”, STQE Publishing, 2001, ISBN 0-970-43630-0

[vanVeenendaal12]: Erik van Veenendaal, “Practical risk-based testing – The PRISMA approach”, UTN Publishers, The Netherlands, ISBN 9789490986070

[Wiegers03]: Karl Wiegers, “Software Requirements 2”, Microsoft Press, 2003, ISBN 0-735-61879-8

[Whittaker03]: James Whittaker, “How to Break Software”, Addison-Wesley, 2003, ISBN 0-201-79619-8

[Whittaker09]: James Whittaker, “Exploratory Software Testing”, Addison-Wesley, 2009, ISBN 0-321-63641-4

8.4 Muut lähteet

Seuraavat viittaukset kohdistuvat Internetistä ja muualta saatavilla olevaan tietoon. Vaikka nämä lähdeviitteet tarkistettiin tämän Jatkotason sertifikaattisisällön julkaisuhetkellä, ISTQB ei kuitenkaan vastaa siitä, jos lähdemateriaali ei ole enää myöhemmin saatavilla.

- Luku 3
  - Czerwonka, Jacek: www.pairwise.org
  - Bug Taxonomy: www.testingeducation.org/a/bsct2.pdf
  - Sample Bug Taxonomy based on Boris Beizer’s work: inet.uni2.dk/~vinter/bugtaxst.doc
  - Good overview of various taxonomies: testingeducation.org/a/bugtax.pdf
  - Heuristic Risk-Based Testing By James Bach
  - From “Exploratory & Risk-Based Testing (2004) www.testingeducation.org"
  - Exploring Exploratory Testing , Cem Kaner and Andy Tikam , 2003
  - Pettichord, Bret, “An Exploratory Testing Workshop Report”, www.testingcraft.com/exploratorypettichord
- Luku 4
  - www.testingstandards.co.uk

© International Software Testing Qualifications Board