Sertifioitu Testaaja
Certified Tester

Perustason sertifikaattisisältö
Foundation Level Syllabus

Versio 2010
Käännösversio 2010

Perustuu englanninkieliseen versioon 30.3.2010
Based on English version 30th of March, 2010

International Software Testing Qualifications Board

Versiohistoria

Versio	Päivä	Huomiot
V2010 Finnish	30.9.2010	Certified Tester Foundation Level Syllabus version 2010 Finnish
ISTQB 2010	Effective 30-Mar-2010	Certified Tester Foundation Level Syllabus Maintenance Release – see **Virhe. Viitteen lähdettä ei löytynyt.**
V2009 Finnish	1.5.2009	Certified Tester Foundation Level Syllabus version 2008 Finnish
ISTQB 2007	01-May-2007 (12Apr07)	Certified Tester Foundation Level Syllabus Maintenance Release – see Appendix E – Release Notes Syllabus 2007
ISTQB 2005	01-July-2005	Certified Tester Foundation Level Syllabus
ASQF V2.2	July-2003	ASQF Syllabus Foundation Level Version 2.2 “Lehrplan „Grundlagen des Softwaretestens“
ISEB V2.0	25-Feb-1999	ISEB Software Testing Foundation Syllabus V2.0 25 February 1999

Sisällysluettelo

Kiitokset

International Software Testing Qualifications Board Foundation-tason työryhmä (Versio 2010): Thomas Müller (puheenjohtaja), Rahul Verma, Martin Klonk ja Armin Beer. Ydintiimi kiittää katselmointitiimiä (Rex Black, Mette Bruhn-Pederson, Debra Friedenberg, Klaus Olsen, Tuula Pääkkönen, Meile Posthuma, Hans Schaefer, Stephanie Ulrich, Pete Williams, Erik van Veendendaal) ja kaikkia kansallisia hallituksia muutosehdotuksista nykyiseen sertifikaattisisällön versioon.

International Software Testing Qualifications Board Foundation-tason työryhmä (Versio 2007): Thomas Müller (puheenjohtaja), Dorothy Graham, Debra Friedenberg ja Erik van Veendendaal ja katselmointitiimi (Hans Schaefer, Stephanie Ulrich, Meile Posthuma, Anders Pettersson ja Wonil Kwon) ja kaikki kansalliset hallitukset ehdotuksineen.

International Software Testing Qualifications Board Foundation-tason työryhmä (Versio 2005): Thomas Müller (puheenjohtaja), Rex Black, Sigrid Eldh, Dorothy Graham, Klaus Olsen, Maaret Pyhäjärvi, Geoff Thompson ja Erik van Veendendaal ja katselmointitiimi ja kaikki kansalliset hallitukset ehdotuksineen.

Johdatus tähän sertifikaattisisältöön

Tämän dokumentin tarkoitus

Tämä sertifikaattisisältö muodostaa pohjan International Software Testing Qualification -vaatimusten Perustasolle. International Software Testing Qualifications Board (ISTQB) antaa sen käyttöön kansallisille tutkintolautakunnille koulutuksen tarjoajien akkreditointia varten sekä pohjaksi muokattaessa koekysymyksiä kansalliselle kielelle. Koulutuksen tarjoajat valitsevat tarkoituksenmukaiset opetusmenetelmät ja tuottavat kurssimateriaalin akkreditoitavaksi. Sertifikaattisisältö auttaa kokelaita tutkintoon valmistautumisessa.

Tietoa sertifikaattisisällön historiasta ja taustasta löytyy Liitteestä A.

Sertifioitu testaaja -perustaso ohjelmistotestauksessa

Perustason pätevyys on tarkoitettu kaikille, jotka osallistuvat ohjelmistotestaukseen. Tämä tarkoittaa esimerkiksi testaajina, testausanalyytikkoina, testaussuunnittelijoina, testauskonsultteina, testauspäälliköinä, hyväksymistestaajina ja sovelluskehittäjinä toimivia henkilöitä. Tämä perustason pätevyys soveltuu myös jokaiselle, joka tarvitsee perusymmärtämystä ohjelmistotestauksesta, kuten esimerkiksi projektipäälliköille, laatupäälliköille, ohjelmistokehityspäälliköille, liiketoiminta-analyytikoille, atk-päälliköille ja johdon konsulteille. Perustason sertifikaatin suorittaneet voivat jatkaa ohjelmistotestauksen korkeamman tason pätevyyteen.

Oppimistavoitteet/osaamistaso

Sertifikaattisisällön jokaisen kappaleen yhteydessä on mainittu sen oppimistavoitteet ja ne on luokiteltu seuraavasti:

1. K1: muistaa, tunnistaa, palauttaa mieleen
2. K2: ymmärtää, selittää, perustella, verrata, luokitella, ryhmitellä, antaa esimerkki, tehdä yhteenveto
3. K3: soveltaa, käyttää
4. K4: analysoida.

Tarkempia tietoja ja esimerkkejä oppimistavoitteista löytyy Liitteestä B.

Kaikki Termit-otsikon alla välittömästi lukujen otsikoiden alapuolella luetellut termit on tarkoitettu muistettavaksi (K1), vaikka tätä ei nimenomaisesti ole mainittu oppimistavoitteissa.

Tutkinto

Perustason sertifiointitutkinto pohjautuu tähän sertifikaattisisältöön. Tutkintokysymyksiin vastaaminen voi edellyttää, että käytetään aineistoa, joka perustuu tämän sertifikaattisisällön useampaan kappaleeseen. Sertifikaattisisällön mistä tahansa kappaleesta saatetaan esittää kysymyksiä.

Muodoltaan tutkinto on monivalintakoe.

Tutkinnon voi suorittaa akkreditoidun valmennuskurssin osana tai erillisessä kuulustelussa (esim. jossain kokeita tarjoavassa keskuksessa tai kaikille avoimessa koetilaisuudessa). Valmennuskurssille osallistuminen ei ole esivaatimus tutkintoon osallistumiselle.

Akkreditointi

ISTQB:n tunnustama kansallinen hallitus voi akkreditoida koulutuksen tarjoajat, joiden kurssimateriaali on tämän sertifikaattisisällön mukainen. Akkreditoinnin ohjeet on hankittava akkreditoinnin suorittavalta lautakunnalta tai elimeltä. Akkreditoidun kurssin katsotaan olevan tämän sertifikaattisisällön mukainen, ja ISTQB-tutkinto saadaan järjestää osana kurssia.

Lisää tietoa koulutuksen tarjoajille löytyy Liitteestä D.

Tarkkuustaso

Tämän sertifikaattisisällön yksityiskohtaisuus mahdollistaa kansainvälisellä tasolla yhdenmukaisen opetuksen ja tutkinnon. Tämän päämäärän saavuttamiseksi sertifikaattisisältö koostuu seuraavista osista:

- yleiset ohjaavat tavoitteet, jotka kuvaavat, mihin perustasolla tähdätään
- opetettavaksi tarkoitettujen tietojen luettelo kuvauksineen sekä tarvittaessa viitteet lisälähteisiin
- tavoiteltavaa kognitiivista oppimistulosta ja ajattelutapaa kuvaavat oppimistavoitteet kullekin tietämysalueelle
- luettelo termeistä, jotka opiskelijan on kyettävä tunnistamaan ja jotka tämän tulee ymmärtää
- opetettavaksi tarkoitettujen avainkäsitteiden kuvaukset lähteineen, kuten hyväksytty kirjallisuus tai standardit.

Sertifikaattisisältö ei ole kuvaus ohjelmistotestauksen koko tietämysalueesta; se kertoo perustason koulutuskursseilta vaadittavan tarkkuustason.

Sertifikaattisisällön rakenne

Päälukuja on kuusi. Ylimmän tason otsikko näyttää luvussa käsiteltävien oppimistavoitteiden ylimmän tason ja kertoo luvun käsittelyyn tarkoitetun ajan. Esimerkiksi:

2. Testaus ohjelmiston elinkaaren aikana (K2)	**115 minuuttia**

Tämä otsikko kertoo, että luvun 2 oppimistavoitteet ovat tasolla K1 (koska ilmoitettu taso on korkeampi) ja K2 (mutta ei K3), ja että luvun aineiston opettamisen on tarkoitettu kestävän 115 minuuttia. Jokainen luku koostuu useasta kappaleesta. Kullekin kappaleelle on myös kerrottu oppimistavoitteet sekä sen käsittelemiseen tarvittava aika. Ne kappaleen alakohdat, joiden käsittelyaikaa ei ole kerrottu, käsitellään kappaleelle varatun ajan puitteissa.

1. Testauksen perusteet (K2) — 155 minuuttia

Oppimistavoitteet Testauksen perusteille

Oppimistavoitteet kuvaavat, mitä osaat tehdä kunkin jakson läpikäymisen jälkeen.

1.1 Miksi testaus on välttämätöntä? (K2)

LO-1.1.1 Pystyt kuvaamaan, myös esimerkkien avulla, millaista vahinkoa vika ohjelmistossa voi aiheuttaa henkilölle, ympäristölle tai yritykselle. (K2)
LO-1.1.2 Erotat vian alkuperäissyyn vian vaikutuksista. (K2)
LO-1.1.3 Osaat perustella esimerkkien avulla, miksi testaus on tarpeellista. (K2)
LO-1.1.4 Osaat kuvata, miksi testaus on osa laadunvarmistusta, ja kertoa esimerkkejä siitä, kuinka testaus myötävaikuttaa laadun parantamiseen. (K2)
LO-1.1.5 Tunnistat termit virhe, vika, häiriö, ja niiden synonyymit erehdys ja "bugi" ja osaat selittää esimerkkien avulla niiden merkitysten erot. (K2)

1.2 Mitä testaus on? (K2)

LO-1.2.1 Tunnistat testauksen yleiset tavoitteet. (K1)
LO-1.2.2 Osaat kertoa esimerkkejä testauksen tavoitteista ohjelmiston elinkaaren eli vaiheissa. (K2)
LO-1.2.3 Pystyt selittämään testauksen ja debuggauksen (virheiden jäljityksen) eron. (K2)

1.3 Seitsemän testauksen pääperiaatetta (K2)

LO-1.3.1 Osaat selittää testauksen seitsemän pääperiaatetta. (K2)

1.4 Testauksen perusprosessi (K1)

LO-1.4.1 Tunnistat viisi testauksen perustehtävää suunnittelusta testauksen päättämiseen sekä jokaisen perustehtävän yksittäiset tehtävät. (K1)

1.5 Testauksen psykologia (K2)

LO-1.5.1 Tiedostat testauksen onnistumiseen vaikuttavat psykologiset tekijät. (K1)
LO-1.5.2 Pystyt vertaamaan testaajan ja kehittäjän ajattelutapoja. (K2)

1.6 Eettiset säännöt (K2)

1.1 Miksi testaus on välttämätöntä? (K2)	**20 minuuttia**

Termit

Bugi, erehdys, häiriö, laatu, riski, vika, virhe

1.1.1 Tietojärjestelmien yhteys testaukseen (K1)

Tietojärjestelmät ovat kasvava osa jokapäiväistä elämää liiketoimintalähtöisistä sovelluksista (esim. pankkitoiminta) kulutustavaroihin (esim. autot). Useimmilla ihmisillä on kokemusta ohjelmistoista, jotka eivät ole toimineet odotetusti. Ohjelmisto, joka ei toimi oikein, voi johtaa monenlaisiin ongelmiin, kuten rahan, ajan tai maineen menetykseen, ja jopa loukkaantumisiin tai kuolemaan.

1.1.2 Ohjelmistovikojen syyt (K2)

Ihminen voi tehdä virheen (tai erehdyksen), joka aiheuttaa vian ohjelmiston tai järjestelmän ohjelmakoodiin tai dokumenttiin. Jos viallinen koodi suoritetaan, järjestelmä tai ohjelmisto epäonnistuu siinä, mitä sen pitäisi tehdä (tai tekee jotain, mitä sen ei pitäisi), ja aiheuttaa häiriön. Viat ohjelmistoissa, järjestelmissä tai dokumenteissa voivat johtaa häiriöihin, mutta kaikki viat eivät välttämättä tee niin.

Vikoja syntyy, koska ihmiset ovat erehtyväisiä. Muita syitä ovat aikataulupaineet, koodin ja infrastruktuurin monimutkaisuus, teknologiamuutokset ja/tai useiden järjestelmien vuorovaikutus toisiinsa.

Häiriöt voivat johtua myös ympäristön aiheuttamista syistä. Esimerkiksi säteily, magnetismi, sähkökentät ja saasteet voivat aiheuttaa vikoja laitteistoissa tai vaikuttaa ohjelman suoritukseen muuttamalla laitteiston toimintaolosuhteita.

1.1.3 Testauksen rooli ohjelmistojen kehityksessä, ylläpidossa ja käytössä (K2)

Perusteellinen järjestelmien ja dokumentaation testaus voi auttaa vähentämään riskiä, että käytön aikana esiintyy ongelmia, sekä parantamaan ohjelmistojärjestelmien laatua, jos löydetyt viat korjataan ennen kuin järjestelmä julkaistaan tuotantokäyttöön.

Ohjelmiston testausta voidaan vaatia myös sopimusteknisistä tai oikeudellisista syistä tai teollisuusstandardien vuoksi.

1.1.4 Testaus ja laatu (K2)

Testauksen avulla on mahdollista mitata ohjelmiston laatua löydettyjen virheiden perusteella sekä toiminnallisten että ei-toiminnallisten (esim. luotettavuus, käytettävyys, toiminnallinen tehokkuus, ylläpidettävyys ja siirrettävyys) vaatimusten ja ominaisuuksien suhteen. Lisää tietoa ei-toiminnallisesta testauksesta löytyy luvusta 2. Lisää tietoa ohjelmistojen ominaisuuksista: kts. ‘Software Engineering – Software Product Quality’ (ISO 9126).

Testaus voi kasvattaa luottamusta ohjelmiston laatuun, jos se löytää vain vähän tai ei yhtään vikoja. Oikein suunniteltu testi, joka menee virheettä läpi, pienentää järjestelmän riskien kokonaistasoa. Kun testaus löytää vikoja, ohjelmiston laatu paranee, kun viat korjataan.

Aiemmista projekteista tulisi ottaa opiksi. Ymmärtämällä muissa projekteissa löydettyjen vikojen alkuperäissyyt, prosesseja voidaan kehittää, minkä puolestaan pitäisi ennaltaehkäistä aiemmin löydettyjen vikojen uudelleensyntymistä ja näin ollen parantaa uusien järjestelmien laatua. Tämä on laadunvarmistuksen näkökulma.

Testaus pitäisi sisällyttää yhdeksi laadunvarmistuksen tehtävistä (eli kehitysstandardien, koulutuksen ja vika-analyysin oheen).

1.1.5 Milloin on testattu tarpeeksi? (K2)

Kun päätetään testauksen riittävyydestä, tulisi ottaa huomioon sekä riskitaso, mukaan luettuna tekniset ja liiketoimintariskit sekä tuoteriskit, että projektin rajoitteet, kuten aika ja budjetti. Riskiä käsitellään tarkemmin luvussa 5.

Testauksen tulisi tuottaa riittävästi tietoa eri osapuolille, jotta he voivat tehdä tietoon pohjautuvan päätöksen testattavan ohjelmiston tai järjestelmän siirtämisestä seuraavaan kehitysvaiheeseen tai luovuttamisesta asiakkaalle.

1.2 Mitä testaus on? (K2)	**30 minuuttia**

Termit

Katselmointi, testauksen tavoite, testaus, testitapaus, vaatimus, virheiden jäljitys.

Tausta

Yleinen käsitys testauksesta on, että se käsittää vain testien suorittamisen esimerkiksi ohjelmaa käyttämällä. Tämä on osa testausta, mutta ei kaikkea testaukseen liittyvistä tehtävistä.

Testaukseen liittyviä tehtäviä esiintyy ennen ja jälkeen testien suorituksen. Näihin kuuluvat testauksen suunnittelu ja hallinta, testattavien tilanteiden valinta, testitapausten suunnittelu ja suoritus, tulosten tarkastaminen, lopetusehtojen arvioiminen, raportointi testausprosessista ja testattavasta järjestelmästä, sekä testauksen viimeistely ja päättäminen, kun testausvaihe on suoritettu loppuun. Testaukseen kuuluvat myös dokumenttien (myös lähdekoodin) katselmoinnit ja staattinen analyysi.

Sekä dynaamista että staattista testausta voidaan käyttää samanlaisten tavoitteiden saavuttamiseen, ja ne tuottavat tietoa sekä testattavan järjestelmän että kehitys- ja testausprosessien parantamiseksi.

Testauksella voi olla seuraavat tavoitteet:

- löydetään vikoja
- saavutetaan luottamus laatuun
- tuotetaan tietoa päätöksentekoa varten
- ehkäistään vikoja.

Testien aikaiseen suunnitteluun liittyvät ajatteluprosessi ja tehtävät (testauksen pohjamateriaalin todentaminen testien suunnittelun avulla) voi auttaa ehkäisemään vikojen syntymistä koodiin. Dokumenttien (esim. vaatimusten) katselmoinnit sekä epäkohtien tunnistaminen ja ratkaisu auttavat myös estämään vikojen ilmaantumista koodiin.

Testauksen eri näkökulmat ottavat huomioon erilaiset tavoitteet. Esimerkiksi kehityksenaikaisessa testauksessa (eli komponentti- integraatio- ja järjestelmätestaus) päätavoite voi olla mahdollisimman monien häiriöiden aiheuttaminen, jotta viat ohjelmistossa tunnistettaan ja voidaan korjata. Hyväksymistestauksessa päätavoite voi olla sen vahvistaminen, että ohjelmisto toimii odotetusti, jotta saavutetaan luottamus siihen, että ohjelmisto täyttää sille asetetut vaatimukset. Joissain tapauksissa testauksen päätavoite voi olla ohjelmiston laadun arviointi (ilman aikomusta virheiden korjaamiseen), jotta eri osapuolille saadaan tietoa riskeistä, jotka liittyvät ohjelman julkaisuun sovittuna ajankohtana. Ylläpitotestaukseen kuuluu usein sen testaus, että uusia vikoja ei ole syntynyt järjestelmään muutosten aikana. Käyttötestauksen aikana päätavoite voi olla järjestelmän laatuominaisuuksien, kuten luotettavuuden tai käytettävyyden, arviointi.

Virheiden jäljitys eli debuggaus ja testaus ovat eri asioita. Dynaaminen testaus voi tuoda esiin häiriöt, jotka ovat vikojen aiheuttamia. Virheiden jäljitys on kehitystoimenpide, jonka avulla löydetään, analysoidaan ja poistetaan häiriön aiheuttaja. Tätä seuraavalla testaajan suorittamalla uusintatestauksella taataan, että korjaus todellakin poistaa häiriön. Vastuu tehtävistä kuuluu eri rooleille: testaajat testaavat ja kehittäjät debuggaavat.

Testausprosessi ja siihen liittyvät tehtävät on selitetty luvussa 1.4.

1.3 Seitsemän testauksen periaatetta (K2)	35 minuuttia

Termit

Täydellinen testaus

Periaatteet

Viimeisen 40 vuoden aikana on ehdotettu muutamia testausperiaatteita, jotka esittelevät kaikelle testaukselle yhteiset yleiset suuntaviivat.

Periaate 1 - Testaus osoittaa vikojen olemassaolon
Testauksella voidaan osoittaa, että vikoja on olemassa, mutta sillä ei voida osoittaa, että vikoja ei ole. Testaus pienentää ohjelmistosta löytämättömien vikojen jäljellä olon todennäköisyyttä, mutta vaikka yhtään vikaa ei löydetä, se ei ole todiste virheettömyydestä.

Periaate 2 - Täydellinen testaus on mahdotonta
Kaiken testaaminen (syötteiden ja esiehtojen kaikki yhdistelmät) ei ole mahdollista lukuun ottamatta triviaaleja tapauksia. Täydellisen testauksen sijaan pitäisi käyttää riskianalyysia ja priorisointia testauksen kohdistamisessa.

Periaate 3 - Aikainen testaus
Testaustehtävät tulisi aloittaa mahdollisimman aikaisessa vaiheessa ohjelmiston tai järjestelmän kehityksen elinkaaressa, ja ne pitäisi kohdistaa määriteltyihin tavoitteisiin.

Periaate 4 - Vikojen kasaantuminen
Testauspanostus tulee kohdistaa suhteutettuna moduulien odotettuun ja myöhemmin havaittuun virhetiheyteen. Yleensä pieni osa komponenteista sisältää suurimman osan ennen julkistusta tehdyssä testauksessa löydetyistä vioista, tai on vastuussa suurimmasta osasta käytönaikaisia häiriöitä.

Periaate 5 - Hyönteismyrkkyparadoksi
Jos samoja testejä toistetaan uudelleen ja uudelleen, lopulta sama testitapausten joukko ei enää löydä uusia vikoja. Tämän "hyönteismyrkkyparadoksin" ylipääsemiseksi testitapauksia on säännöllisesti katselmoitava ja muokattava, ja uusia, erilaisia testejä on laadittava, jotta ohjelmistosta tai järjestelmästä käydään läpi eri alueita ja löydetään mahdollisesti enemmän vikoja.

Periaate 6 - Testaus on tilanneriippuvaista
Testausta tehdään eri tavalla eri tilanteissa. Esimerkiksi turvallisuuskriittinen sovellus testataan eri tavalla kuin verkkokauppasivusto.

Periaate 7 - Virheettömyyden harhaluulo
Vikojen löytäminen ja korjaaminen ei auta, jos rakennettu järjestelmä on käyttökelvoton eikä täytä käyttäjien tarpeita ja odotuksia.

1.4 Testauksen perusprosessi (K1)	**35 minuuttia**

Termit

Havainto, lopetusehdot, regressiotestaus, testattava tilanne, testauksen kohteen kuvaus, testauksen materiaalit, testauksen yhteenvetoraportti, testauspolitiikka, testaussuunnitelma, testiaineisto, testijoukko, testikattavuus, testiloki (testipäiväkirja), testin suorittaminen, testiproseduuri, uudelleentestaus, varmistustestaus.

Tausta

Näkyvin osa testausta on testien suoritus. Ollakseen kuitenkin sisäisesti ja ulkoisesti tehokkaita, testaussuunnitelmissa pitäisi ottaa huomioon myös testauksen suunnitteluun, testitapausten suunnitteluun, suorituksen valmisteluun ja tilan arviointiin kuluva aika.

Perustestausprosessi koostuu seuraavista toimenpiteistä.

- Suunnittelu ja valvonta
- Analysointi ja suunnittelu
- Valmistelu ja suoritus
- Lopetusehtojen arviointi ja raportointi
- Testauksen päättämistoimenpiteet

Vaikka tehtävät ovat loogisesti peräkkäisiä, prosessissa ne voivat mennä päällekkäin tai tapahtua samanaikaisesti. Yleensä näitä päätehtäviä täytyy muokata järjestelmän ja projektin kannalta sopiviksi.

1.4.1 Testauksen suunnittelu ja valvonta (K1)

Testauksen suunnittelutehtäviin kuuluvat testauksen tavoitteiden määrittäminen sekä näiden tavoitteiden ja testauksen mission saavuttamiseksi tarvittavien testaustehtävien määrittely.

Testauksen valvonta tarkoittaa jatkuvaa testauksen todellisen ja suunnitellun etenemisen vertailua sekä raportointia testauksen tilasta ja suunnitellusta etenemisestä poikkeamista. Siihen kuuluu myös toimenpiteisiin ryhtyminen tarvittaessa projektin mission ja tavoitteiden saavuttamiseksi. Jotta testausta voidaan hallita, testaustehtäviä pitää seurata läpi projektin. Testauksen suunnittelussa otetaan huomioon seuranta- ja hallintatehtäviltä tuleva palaute.

Testauksen suunnittelun ja valvonnan tehtävät on selitetty luvussa 5.

1.4.2 Testien analysointi ja suunnittelu (K1)

Testien analysoinnin ja suunnittelun yhteydessä yleiset testaustavoitteet muutetaan konkreettisiksi testattaviksi tilanteiksi ja testitapauksiksi.

Testien analysointiin ja suunnittelun kuuluvat seuraavat päätehtävät:

- testauksen lähdedokumenttien (kuten vaatimukset, ohjelmiston eheyden taso (riskitaso), riskianalyysiraportit, arkkitehtuuri, suunnittelu, rajapinnat) katselmointi
- testauksen kohteen kuvauksen ja testauksen kohteiden testattavuuden arviointi
- testattavien nimikkeiden, määrittelyiden, käyttäytymisen ja rakenteen analyysiin perustuva testattavien tilanteiden tunnistaminen ja priorisointi
- ylemmän tason testitapausten suunnittelu ja priorisointi

- testattavien tilanteiden ja testitapausten tueksi tarvittavan testiaineiston tunnistaminen
- testiympäristön kokoonpanon suunnittelu ja tarvittavan infrastruktuurin ja välineiden tunnistaminen
- kaksisuuntaisen jäljitettävyyden luominen testauksen lähdedokumenttien ja testitapausten välille.

1.4.3 Testien toteutus ja suorittaminen (K1)

Testien toteutus ja suorittaminen ovat tehtäviä, joiden aikana testiproseduurit tai skriptit määritellään yhdistelemällä testitapaukset tiettyyn järjestykseen ja mukaan otetaan kaikki muu tarvittava tieto, jota tarvitaan testien suorittamiseen. Testiympäristö pystytetään ja testit suoritetaan.

Testin toteutukseen ja suoritukseen kuuluvat seuraavat päätehtävät:

- testitapausten viimeistely, toteutus ja priorisointi (mukaan luettuna testiaineiston määrittäminen)
- testiproseduurien suunnittelu ja priorisointi, testiaineiston luominen ja, valinnaisesti, testikehysten valmistelu ja automatisoitujen testiskriptien kirjoittaminen
- testijoukkojen luominen testiproseduureista testien suorittamisen tehostamiseksi
- sen varmistaminen, että testiympäristö on pystytetty oikein
- testauksen lähdedokumenttien ja testitapausten välisen kaksisuuntaisen jäljitettävyyden todentaminen ja päivittäminen
- testiproseduurien suorittaminen suunnitellussa järjestyksessä joko käsin tai testien suoritustyökaluja käyttämällä.
- testin suorituksen tulosten kirjaaminen ja testattavan ohjelmiston, testityökalujen ja testausmateriaalin tunnisteiden ja versioiden tallennus
- todellisten tulosten vertaaminen odotettuihin tuloksiin
- poikkeavuuksien raportointi havaintoina ja niiden analysointi niiden syyn selvittämiseksi (esim. vika koodissa, määritetyssä testiaineistossa, testidokumentissa, tai virhe testin suoritustavassa)
- testaustehtävien toistaminen kunkin poikkeavuuden aiheuttamien toimenpiteiden jälkeen. Esimerkiksi aikaisemmin epäonnistuneiden testien ajaminen uudelleen vian korjaantumisen varmistamiseksi (varmistustestaus), korjatun testin suorittaminen ja/tai testien ajaminen, jotta varmistetaan, että vikoja ei ole syntynyt ohjelmiston muuttamattomille alueille tai että vian korjaaminen ei paljastanut muita vikoja (regressiotestaus).

1.4.4 Testauksen hyväksymiskriteerit ja raportointi (K1)

Lopetusehtojen arviointi on toimenpide, jolloin testien suoritusta arvioidaan määriteltyjä tavoitteita vasten. Tämä pitäisi tehdä jokaisella testitasolla (katso kohta 2.2).

Lopetuskriteerin arviointiin kuuluvat seuraavat päätehtävät:

- testilokien tarkistaminen testaussuunnittelussa määriteltyjä lopetusehtoja vasten
- sen arvioiminen, tarvitaanko lisää testejä vai pitäisikö määriteltyjä lopetuskriteerejä muuttaa
- testauksen yhteenvetoraportin kirjoittaminen eri osapuolia varten.

1.4.5 Testauksen päättämistehtävät (K1)

Testauksen päättämistehtävissä kerätään tietoja loppuunviedyistä toimenpiteistä kokemusten, testimateriaalin, tietojen ja numeroiden yhteen kokoamiseksi. Päättämistehtäviä tehdään projektin tarkistuspisteiden kohdalla, esimerkiksi kun tietojärjestelmä julkaistaan, testausprojekti päätetään (tai peruutetaan), etappi tai tarkistuspiste on saavutettu tai ylläpitojulkaisu on saatu valmiiksi.

Testauksen päättämistehtäviin kuuluvat seuraavat päätehtävät:

- sen tarkistaminen, mitkä suunnitelluista tuotoksista on toimitettu
- havaintoraporttien sulkeminen tai muutostietojen kirjaaminen avoimien havaintojen osalta
- järjestelmän hyväksymisen dokumentoiminen

- testauksen materiaalien, testiympäristön ja testauksen infrastruktuurin viimeistely ja arkistointi myöhempää uudelleenkäyttöä varten
- testauksen materiaalien luovuttaminen ylläpito-organisaatiolle
- sen analysoiminen, mitä tehdystä voi ottaa opiksi tulevia julkaisuja ja projekteja silmällä pitäen
- kerätyn tiedon käyttäminen testauskypsyyden parantamiseksi.

1.5 Testauksen psykologia(K2)	25 minuuttia

Termit

Riippumattomuus, virheenarvaus.

Tausta

Testauksen ja katselmoinnin aikana käytettävä ajattelutapa on erilainen kuin ohjelmistokehityksen aikana käytettävä. Oikealla asennoitumisella kehittäjät kykenevät testaamaan omaa koodiaan, mutta tämän vastuun eriyttäminen testaajalle tehdään tyypillisesti työpanoksen keskittämiseksi ja lisäetujen, kuten koulutettujen ammattitestaajien riippumattoman näkemyksen, saamiseksi. Riippumatonta testausta voidaan tehdä jokaisella testauksen tasolla.

Tietty riippumattomuuden aste (tekijän ennakkonäkemysten välttäminen) tehostaa usein vikojen ja häiriöiden löytämistä. Riippumattomuus ei kuitenkaan ole tuttuuden korvike, ja kehittäjät voivat tehokkaasti löytää monia vikoja omasta koodistaan. Testauksen riippumattomuudelle voidaan määritellä useita tasoja, kuten seuraavassa on esitelty matalasta korkeaan:

- Testit suunnittelee testattavan ohjelmiston tekijä(t) (matala riippumattomuus)
- Testit suunnittelee eri henkilö kuin tekijä (esim. samasta kehitystiimistä)
- Testit suunnittelee henkilö saman organisaation eri yksiköstä (esim. riippumattomasta testaustiimistä) tai testauksen asiantuntija (esim. käytettävyyden tai suorituskykytestauksen asiantuntija)
- Testit suunnittelee henkilö eri organisaatiosta tai yrityksestä (ulkoistettu testaus tai ulkopuolisen tahon suorittama sertifiointitestaus)

Tavoitteet ohjaavat ihmisiä ja projekteja. Ihmisillä on tapana järjestää suunnitelmansa johdon tai muiden osallisten asettamien tavoitteiden (esim. vikojen löytäminen tai ohjelmiston toimivuuden varmistaminen) mukaan. Siksi on tärkeää ilmaista testauksen tavoitteet selvästi.

Häiriöiden tunnistaminen testauksen aikana voidaan käsittää tuotteen tai tekijän kritisoinniksi. Sen vuoksi testaus nähdään usein tuhoavana tehtävänä, vaikka se on hyvin rakentavaa tuoteriskien hallinnassa. Häiriöiden etsiminen järjestelmästä vaatii uteliaisuutta, ammattimaista pessimismiä, kriittistä silmää, yksityiskohtien huomaamista, hyvää viestintää kehittäjien kanssa, ja kokemusta, johon virheenarvaus voidaan pohjata.

Jos virheistä, vioista tai häiriöistä viestitään rakentavalla tavalla, epäsopu testaajien, suunnittelijoiden ja sovelluskehittäjien kanssa voidaan välttää. Tämä pätee yhtä hyvin katselmointien kuin testauksen aikana löydettyihin vikoihin.

Testaaja ja testauspäällikkö tarvitsevat hyviä ihmissuhdetaitoja pystyäkseen välittämään asiatietoa vioista, etenemisestä ja riskeistä rakentavalla tavalla. Virhetiedot voivat auttaa ohjelmiston tai dokumentin tekijää parantamaan taitojaan. Testauksen aikana löydetyt ja korjatut viat säästävät myöhemmin aikaa ja rahaa sekä pienentävät riskejä.

Viestintäongelmia voi esiintyä erityisesti, jos testaajat nähdään vain ei-toivottujen virheilmoitusten tuojina. On kuitenkin olemassa monia tapoja parantaa testaajien ja muiden välistä kommunikaatiota ja suhteita:

- Aloita yhteistyöllä ennemmin kuin taistelulla - muistuta kaikkia parempilaatuisesta järjestelmästä yhteisenä tavoitteena.
- Tiedota löydöksistä neutraalilla, tosiasioihin keskittyvällä tavalla ilman henkilökohtaista kritiikkiä tuotteen tekijää kohtaan; kirjaa esimerkiksi havaintoraportit ja katselmointihavainnot puolueettomasti ja totuudenmukaisesti.
- Yritä ymmärtää, miltä toisesta henkilöstä tuntuu ja miksi hän reagoi jollain tietyllä tavalla.
- Varmista, että henkilö on ymmärtänyt, mitä olet sanonut, ja päinvastoin.

1.6 Eettiset säännöt (K2)	10 minuuttia

Ohjelmistotestauksen kanssa tekemisissä olevat henkilöt voivat saada tietoonsa luottamuksellisia asioita ja joutua käsittelemään salassa pidettäviä tietoja. Eettiset säännöt ovat välttämättömiä muun muassa sen varmistamiseksi, että tällaisia tietoja ei käytetä epäasianmukaisella tavalla. ACM:n ja IEEE:n insinööreille laatimat eettiset säännöt huomioon ottaen ISTQB on määrittänyt seuraavat eettiset säännöt:

YLEINEN HYÖTY – Sertifioidut ohjelmistotestaajat toimivat yleisen hyödyn mukaisesti

ASIAKAS JA TYÖNANTAJA – Sertifioidut ohjelmistotestaajat toimivat tavalla, joka on heidän asiakkaansa ja työnantajansa parhaaksi, mutta samalla yleisen hyödyn mukaisesti.

TUOTE – Sertifioidut ohjelmistotestaajat varmistavat, että heidän laatimansa testattavien tuotteiden ja järjestelmien tuotokset täyttävät korkeimmat mahdolliset ammatilliset standardit.

ARVOSTELUKYKY – Sertifioidut ohjelmistotestaajat toimivat rehellisesti ja puolueettomasti ammatillisissa arvioissaan.

JOHTAMINEN – Sertifioidut ohjelmistotestauspäälliköt ja -johtajat tukevat ja edistävät eettistä lähestymistapaa ohjelmistotestauksen johtamisessa.

AMMATTIKUNTA – Sertifioidut ohjelmistotestaajat pyrkivät edistämään luottamusta testaajiin ja testaajien hyvää mainetta yleisen hyödyn mukaisesti.

TYÖTOVERIT – Sertifioidut ohjelmistotestaajat toimivat reilusti ja kannustavasti työtovereitaan kohtaan sekä edistävät yhteistyötä ohjelmistokehittäjien kanssa.

OMA ITSE – Sertifioidut ohjelmistotestaajat pyrkivät jatkuvasti kehittämään ammattitaitoaan ja edistämään eettistä lähestymistapaa testaajana toimimiseen.

Viitteet

1.1.5 Black, 2001, Kaner, 2002

1.2 Beizer, 1990, Black, 2001, Myers, 1979

1.3 Beizer, 1990, Hetzel, 1988, Myers, 1979

1.4 Hetzel, 1988

1.4.5 Black, 2001, Craig, 2002

1.5 Black, 2001, Hetzel, 1988

2. Testaus ohjelmiston elinkaaren aikana (K2)

115 minuuttia

Oppimistavoitteet testaukselle ohjelmiston elinkaaren aikana

Oppimistavoitteet kuvaavat, mitä osaat tehdä kunkin jakson läpikäymisen jälkeen.

2.1 Ohjelmistokehitysmallit (K2)

LO-2.1.1 Ymmärrät kehityksen, testaustehtävien ja tuotosten välisen suhteen ohjelmistokehityksen elinkaaressa, ja osaat kertoa esimerkkejä projektin ja tuotteen piirteiden sekä tilanteen perusteella. (K2)
LO-2.10.2 Ymmärrät, että ohjelmistokehitysmalleja täytyy muokata projektin ja tuotteen ominaisuuksien perusteella tilanteeseen sopiviksi. (K1)
LO-2.1.3 Tunnistat hyvän testauksen piirteet, jotka pätevät kaikkiin elinkaarimalleihin. (K1)

2.2 Testaustasot (K2)

LO-2.2.1 Pystyt vertailemaan eri testaustasoja: niiden päätavoitteet, testauksen tyypilliset kohteet, tyypilliset testaustyypit (esim. toiminnalliset tai rakenteelliset) ja tasoon liittyvät vaihetuotteet, ketkä testauksen suorittavat, ja tyypillisesti tunnistettavat viat ja häiriöt. (K2)

2.3 Testaustyypit(K2)

LO-2.3.1 Pystyt vertaamaan esimerkkien avulla neljää ohjelmistojen testauksen tyyppiä (toiminnallinen, ei-toiminnallinen, rakenteellinen ja muutokseen pohjautuva). (K2)
LO-2.3.2 Ymmärrät, että toiminnallista ja rakenteellista testausta tehdään kaikilla testaustasoilla. (K1)
LO-2.3.3 Tunnistat ja osaat kuvata ei-toiminnallisiin vaatimuksiin pohjautuvia ei-toiminnallisia testityyppejä. (K2)
LO-2.3.4 Tunnistat ja osaat kuvata tietojärjestelmän rakenteen tai arkkitehtuurin analyysiin pohjautuvia testityyppejä. (K2)
LO-2.3.5 Osaat kuvata uudelleentestauksen ja regressiotestauksen tarkoituksen. (K2)

2.4 Ylläpitotestaus (K2)

LO-2.4.1 Pystyt vertaamaan ylläpitotestausta (olemassa olevan järjestelmän testaus) uuden sovelluksen testaukseen käytettävien testityyppien, testauksen käynnistykseen vaikuttavien tekijöiden ja tarvittavan testauksen määrän perusteella. (K2)
LO-2.4.2 Tunnistat syyt ylläpitotestaukselle (muutos, migraatio tai eläköityminen). (K1)
LO-2.4.3. Osaat kuvata regressiotestauksen ja vaikutusanalyysin merkityksen ylläpidossa. (K2)

2.1 Ohjelmistokehitysmallit (K2)	**20 minuuttia**

Termit

Iteratiivis-inkrementaalinen kehitysmalli, kelpuuttaminen, todentaminen, valmisohjelmisto (COTS), V-malli.

Tausta

Testausta ei ole olemassa yksinään; testaustehtävät liittyvät ohjelmiston kehitystehtäviin. Erilaiset kehityselinkaarimallit tarvitsevat erilaisia lähestymistapoja testaukseen.

2.1.1 V-malli (vaiheittainen kehitysmalli) (K2)

Vaikka V-mallista on eri versioita, tyypillinen V-malli käyttää neljää testitasoa, jotka vastaavat neljää kehitystasoa.

Tässä sertifikaattisisällössä käsiteltävät neljä tasoa ovat

- komponentti- (yksikkö-)testaus
- integrointitestaus
- järjestelmätestaus
- hyväksymistestaus.

Käytännössä V-mallissa voi olla enemmän, vähemmän tai eri kehityksen ja testauksen tasoja, riippuen projektista ja ohjelmistotuotteesta. Esimerkiksi komponenttitestauksen jälkeen voi tulla komponentti-integrointitestaus ja järjestelmätestauksen jälkeen järjestelmäintegraatiotestaus.

Ohjelmiston vaihetuotteet (kuten liiketoimintaskenaariot tai käyttötapaukset, vaatimusmäärittelyt, suunnitteludokumentit ja koodi), jotka tuotetaan kehityksen aikana, muodostavat usein perustan yhden tai useamman testitason testaukselle. Yleisten vaihetuotteiden viitteisiin kuuluvat Capability Maturity Model Integration (CMMI) tai 'Ohjelmiston elinkaariprosessit' (IEEE/IEC 12207). Todentaminen ja kelpuuttaminen (sekä aikainen testien suunnittelu) voidaan suorittaa ohjelmiston vaihetuotteiden kehityksen aikana.

2.1.2 Iteratiivis-inkrementaaliset kehitysmallit (K2)

Iteratiivis-inkrementaalinen kehitys on järjestelmän vaatimusmäärittelyn, suunnittelun, järjestelmän toteutuksen ja testauksen prosessi, joka suoritetaan lyhyempien kehityskierrosten sarjana. Esimerkkejä ovat prototyyppien käyttö, "nopea sovelluskehitys" (Rapid Application Development, RAD), Rational Unified Process (RUP) ja ketterät kehitysmallit. Iteraatiossa syntynyt järjestelmä voidaan testata usealla tasolla jokaisen iteraatiokierroksen aikana. Inkrementti, joka on liitetty aikaisemmin kehitettyihin inkrementteihin, muodostaa kasvavan osittaisen järjestelmän, jota pitää myös testata. Ensimmäisen iteraation jälkeen regressiotestauksen merkitys kasvaa jatkuvasti. Todentaminen ja kelpuuttaminen voidaan tehdä jokaiselle inkrementille.

2.1.3 Testaus elinkaaren aikana (K2)

Jokaiseen elinkaarimalliin kuuluu useita hyvän testauksen ominaisuuksia:

- Jokaiselle kehityksen tehtävälle on olemassa vastaava testaustehtävä.
- Jokaiselle testitasolle on erityisesti sille määritellyt nimenomaiset tavoitteet
- Tietyn testitason testien analysoinnin ja suunnittelun pitäisi alkaa vastaavan kehitysvaiheen aikana.

- Testaajien pitäisi olla mukana katselmoimassa dokumentteja niin pian kuin niiden luonnoksia on saatavissa kehityksen elinkaaren aikana.

Testitasoja voidaan yhdistää ja järjestää uudelleen projektin tai järjestelmäarkkitehtuurin luonteen mukaan. Esimerkiksi kun ollaan integroimassa kaupallista valmisohjelmistoa järjestelmään, ostaja voi suorittaa integrointitestausta järjestelmätasolla (esim. integraatio infrastruktuuriin ja muihin järjestelmiin tai järjestelmän käyttöönotto) sekä hyväksymistestausta (toiminnallinen ja/tai ei-toiminnallinen sekä käyttäjä- ja/tai käyttötestaus).

2.2 Testitasot (K2)	**40 minuuttia**

Termit

Ajuri, alfatestaus, betatestaus, ei-toiminnallinen vaatimus, integraatio, integrointitestaus, järjestelmätestaus, kenttätestaus, komponenttitestaus (tunnetaan myös yksikkö-, moduuli- tai ohjelmatestauksena), (käyttäjän) hyväksymistestaus, testiohjattu kehitys, testitaso, (testi)tynkä, testiympäristö, toiminnallinen vaatimus, vakaustestaus.

Tausta

Jokaiselle testitasolle voidaan tunnistaa seuraavat asiat: tason yleiset tavoitteet, vaihetuotteet, joihin viitataan testitapauksien tuottamiseksi (eli testauksen lähdedokumentaatio), testauksen kohde (eli mitä testataan), kyseisellä tasolla tyypillisesti löydettävät viat ja häiriöt, testikehysvaatimukset ja työkalutuki, sekä erityiset lähestymistavat ja vastuut.

Järjestelmän kokoonpanoon liittyvän aineiston testausta pitää harkita testisuunnittelun aikana, mikäli tällainen aineisto on osa järjestelmää.

2.2.1 Yksikkötestaus (K2)

Testauksen pohjamateriaali:

- Komponentin vaatimukset
- Yksityiskohtaiset suunnitelmat
- Koodi

Tyypilliset testauksen kohteet:

- Komponentit
- Ohjelmat
- Aineiston konversio- ja migraatio-ohjelmat

Tietokantamoduulit

Yksikkötestaus (tunnetaan myös komponentti-, moduuli- tai ohjelmatestauksena) etsii vikoja yksinään testattavista ohjelmiston osista (esim. moduuleista, ohjelmista, olioista, luokista, jne.) ja todentaa niiden toimintaa. Se voidaan tehdä erillään muusta järjestelmästä, riippuen kehityselinkaaren tilanteesta ja järjestelmästä. Tynkiä, ajureita tai simulaattoreita voidaan käyttää apuna.

Yksikkötestaukseen voi sisältyä toiminnallisuuden ja erityisen ei-toiminnallisten piirteiden testausta, kuten resurssien käyttö (esim. muistivuodot) tai vakaustestaus, samoin kuin rakenteellista testausta (esim. haarakattavuus). Testitapaukset tuotetaan vaihetuotteista, kuten komponentin määrityksistä, ohjelmiston rakenteesta tai tietomallista.

Tyypillisesti yksikkötestauksessa tarkastellaan suoraan testattavaa koodia kehitysympäristön tuen, kuten yksikkötestauskehyksen tai debuggaustyökalun, avulla. Käytännössä suorittajana on tavallisesti koodin kirjoittanut ohjelmoija. Viat korjataan tavallisesti heti kun ne löydetään, ilman, että havaintoja hallitaan muodollisesti.

Yksi lähestymistapa yksikkötestaukseen on valmistella ja automatisoida testitapauksia ennen koodausta. Tätä kutsutaan testauslähtöiseksi tai testausohjatuksi kehitykseksi. Tämä lähestymistapa on erittäin iteratiivinen ja perustuu kierroksiin, joissa testitapaukset suunnitellaan, ja sen jälkeen pieniä osia koodia toteutetaan ja integroidaan, ja yksikkötestejä suoritetaan ja esiin tulleita vikoja korjataan, kunnes testit menevät läpi.

2.2.2 Integrointitestaus (K2)

Testauksen pohjamateriaali:

- Ohjelmiston ja järjestelmän suunnitelmat
- Arkkitehtuuri
- Työnkulut
- Käyttötapaukset

Tyypilliset testauksen kohteet:

- Alijärjestelmien tietokantatoteutukset
- Infrastruktuuri
- Rajapinnat

Järjestelmän kokoonpano

- Kokoonpanotiedot

Integrointitestaus testaa komponenttien välisiä rajapintoja, järjestelmän eri osien, kuten käyttöjärjestelmän, tiedostojärjestelmän ja/tai laitteiston osien välistä vuorovaikutusta, tai järjestelmien välisiä rajapintoja.

Integrointitestaustasoja voi olla useampi kuin yksi ja sitä voidaan suorittaa erikokoisille kohteille seuraavasti:

1. Komponentti-integrointitestauksessa testataan ohjelmistokomponenttien välistä vuorovaikutusta ja sitä tehdään yksikkötestauksen jälkeen.
2. Järjestelmäintegrointitestauksessa testataan eri järjestelmien välistä vuorovaikutusta, ja sitä voidaan tehdä järjestelmätestauksen jälkeen. Tässä tapauksessa kehitysorganisaatio voi kontrolloida vain rajapinnan yhtä puolta. Tämä voi muodostaa riskin. Työketjuina toteutetut liiketoimintaprosessit voivat koskea useita järjestelmiä. Alustojen vuorovaikutukseen liittyvät seikat voivat olla merkittäviä.

Mitä suurempi integraatiokokonaisuus on, sitä vaikeammaksi tulee häiriöiden eristäminen tiettyyn komponenttiin tai järjestelmään, mikä voi johtaa lisääntyneeseen riskiin ja ongelmanselvityksen ajantarpeeseen.

Systemaattiset integrointistrategiat voivat perustua järjestelmäarkkitehtuuriin (kuten ylhäältä-alas tai alhaalta-ylös), toiminnallisiin tehtäviin, tapahtumien prosessointijärjestykseen tai johonkin toiseen järjestelmän tai komponentin piirteeseen. Vikojen eristämisen helpottamiseksi ja aikaiseksi löytämiseksi integroinnin pitäisi normaalisti olla vaiheittaista ennemmin kuin "kertarysäys".

Tiettyjen ei-toiminnallisten piirteiden (esim. suorituskyvyn) testaus samoin kuin toiminnallinen testaus voidaan ottaa mukaan integrointitestaukseen.

Jokaisessa integraation vaiheessa testaajat keskittyvät vain itse integraatioon. Jos he esimerkiksi integroivat moduulia A moduuliin B, he ovat kiinnostuneita testaamaan näiden moduulien välistä kommunikaatiota, eivät jommankumman moduulin toiminnallisuutta, sillä se on testattu jo yksikkötestauksessa. Sekä toiminnallisia että rakenteellisia lähestymistapoja voidaan käyttää.

Ihannetilanteessa testaajien pitäisi ymmärtää ohjelmiston arkkitehtuuri ja vaikuttaa integroinnin suunnitteluun. Jos integrointitestit suunnitellaan ennen kuin komponentteja tai järjestelmiä rakennetaan, komponentit voidaan rakentaa järjestyksessä, joka tarvitaan testauksen suorittamiseksi kaikkein tehokkaimmin.

2.2.3 Järjestelmätestaus (K2)

Testauksen pohjamateriaali:

- Järjestelmän ja ohjelmiston vaatimusmäärittelyt
- Käyttötapaukset
- Toiminnalliset määritykset
- Riskianalyysiraportit

Tyypilliset testauksen kohteet:

- Järjestelmän tekninen dokumentaatio, käyttöohjeet, loppukäyttäjän ohjeet
- Järjestelmän kokoonpano

Kokoonpanotiedot

Järjestelmätestaus koskee koko järjestelmän tai tuotteen toimintaa. Testauksen laajuus pitää kuvata selkeästi kokonaistestaussuunnitelmassa tai kyseisen tason testaussuunnitelmassa.

Järjestelmätestauksessa testiympäristön pitäisi vastata lopullista kohdetta tai tuotantoympäristöä niin paljon kuin mahdollista, jotta voidaan minimoida ympäristöstä aiheutuvien häiriöiden löytämättä jäämisen riskit.

Järjestelmätestaus voi sisältää testejä, jotka perustuvat riskeihin ja/tai vaatimusmäärittelyihin, liiketoimintaprosesseihin, käyttötapauksiin tai korkean tason kuvauksiin tai malleihin järjestelmän toiminnasta, vuorovaikutuksesta käyttöjärjestelmän kanssa ja järjestelmän resursseista.

Järjestelmätestauksen pitäisi tutkia järjestelmän toiminnallisia ja ei-toiminnallisia vaatimuksia sekä järjestelmän käyttämän aineiston laatua. Testaajat joutuvat myös tekemisiin puutteellisten tai dokumentoimattomien vaatimusten kanssa. Toiminnallisten vaatimusten järjestelmätestaus alkaa käyttämällä testattavan järjestelmänosan perusteella soveltuvimpia määrittelypohjaisia (mustalaatikko) tekniikoita. Esimerkiksi liiketoimintasääntöjen seurausten yhdistelmien kuvaamiseksi voidaan laatia päätöstaulu. Sen jälkeen voidaan käyttää rakenteeseen pohjautuvia tekniikoita (lasilaatikko), kun arvioidaan rakenne-elementtien, kuten esimerkiksi valikkorakenteen tai web-sivun navigaation, testauksen perusteellisuutta. (Katso Luku 4).

Järjestelmätestauksen suorittaa usein itsenäinen testaustiimi.

2.2.4 Hyväksymistestaus (K2)

Testauksen pohjamateriaali:

- Käyttäjävaatimukset
- Järjestelmävaatimukset
- Käyttötapaukset
- Liiketoimintaprosessit
- Riskianalyysiraportit

Tyypilliset testauksen kohteet:

Liiketoimintaprosessit kokonaan integroidussa järjestelmässä

- Toiminnalliset ja ylläpitoprosessit
- Käyttäjäproseduurit
- Lomakkeet
- Raportit

Kokoonpanotiedot

Hyväksymistestaus on usein asiakkaiden tai järjestelmän käyttäjien vastuulla; muitakin sidosryhmiä voi olla mukana.

Hyväksymistestauksen tavoite on saada luottamusta järjestelmään, sen osiin tai tiettyihin ei-toiminnallisiin järjestelmän piirteisiin. Vikojen löytäminen ei ole päätavoite hyväksymistestauksessa. Hyväksymistestauksessa voidaan arvioida järjestelmän valmiutta käyttöönottoon ja käyttöön, vaikka se ei välttämättä ole viimeinen testauksen vaihe. Esimerkiksi laaja järjestelmäintegrointitesti voidaan suorittaa järjestelmän hyväksymistestauksen jälkeen.

Hyväksymistestausta voidaan suorittaa useita kertoja elinkaaren aikana. Esimerkiksi:

- Kaupallinen valmisohjelmisto (COTS) voidaan hyväksymistestata, kun se asennetaan tai integroidaan.
- Komponentin käytettävyyden hyväksymistestaus voidaan tehdä yksikkötestauksen aikana.
- Uuden toiminnallisuuden lisäyksen hyväksymistestaus voi tapahtua ennen järjestelmätestausta.

Tyypillisiä hyväksymistestauksen muotoja ovat seuraavat:

Käyttäjän hyväksymistestaus
Tyypillisesti todentaa järjestelmän sopivuuden käyttöön liiketoiminnan edustajien näkökulmasta.

Käyttöön soveltuvuuden hyväksymistestaus
Järjestelmän hyväksyminen systeemikäyttäjän näkökulmasta. Tähän kuuluvat

- varmuuskopioinnin/palautuksen testaus
- virhetilanteesta toipuminen
- käyttäjien hallinta
- ylläpitotehtävät
- aineiston luominen ja siirto järjestelmien välillä
- tietoturvahaavoittuvuuksien ajoittainen tarkastaminen.

Sopimusten ja säännösten perusteella tehtävä hyväksymistestaus
Sopimus- tai säännöspohjainen hyväksymistestaus tehdään sopimuksen hyväksymiskriteereitä vastaan, kun tehdään tilaustyönä kehitettyä ohjelmistoa. Hyväksymiskriteerit pitäisi määritellä, kun sopimus tehdään. Säännösten perusteella tehtävää hyväksymistestausta tehdään kaikkia sellaisia säännöksiä vastaan, joita pitää noudattaa, kuten valtiovallan säätämät tai lakeihin tai turvallisuussäännöksiin pohjautuvat määräykset.

Alfa- ja beta- (tai kenttä-) testaus
Myyntiin tarkoitettujen kaupallisten valmisohjelmistojen valmistajat haluavat usein saada palautetta markkinoidensa potentiaalisilta tai olemassa olevilta käyttäjiltä ennen kuin ohjelmistotuote lasketaan kaupallisesti myyntiin. Alfatestaus suoritetaan kehitysorganisaation tiloissa, mutta sitä eivät tee järjestelmän kehittäjät. Betatestauksen, tai kenttätestauksen, potentiaaliset tai nykyiset asiakkaat suorittavat omissa tiloissaan.

Organisaatiot voivat käyttää myös muita nimityksiä, kuten toimittajan hyväksymistestausta ja toimipaikkakohtaista hyväksymistestausta järjestelmille, joita testataan ennen asiakkaan tiloihin siirtämistä sekä sen jälkeen.

2.3 Testaustyypit (K2)	40 minuuttia

Termit

Koodikattavuus, kuormitustestaus, käytettävyystestaus, lasilaatikkotestaus, luotettavuustestaus, mustalaatikkotestaus, määrittelypohjainen testaus, rakenteellinen testaus, rasitustestaus, siirrettävyystestaus, suorituskykytestaus, tietoturvatestaus, toiminnallinen testaus, yhteentoimivuustestaus, ylläpidettävyystestaus.

Tausta

Joukko testaustehtäviä voidaan kohdistaa tietojärjestelmän (tai järjestelmän osan) todentamiseen tietyn syyn tai testauksen tavoitteen perusteella.

Testityyppi keskittyy tiettyyn testaustavoitteeseen, joka voi olla mikä tahansa seuraavista:

- toiminto, joka ohjelmiston tulee suorittaa
- ei-toiminnallinen laatuominaisuus, kuten luotettavuus tai käytettävyys
- ohjelmiston tai järjestelmän rakenne tai arkkitehtuuri
- Muutoksiin liittyvä, eli sen varmistaminen, että viat on korjattu (uudelleentestaus) ja tahattomien muutosten etsiminen (regressiotestaus).

Ohjelmistosta voidaan laatia malli, jota voidaan käyttää rakenteellisessa testauksessa (esim. tietovirtamalli tai valikkorakennemalli), ei-toiminnallisessa testauksessa (esim. suoritusmalli, käytettävyysmalli, tietoturvauhkamalli) ja toiminnallisessa testauksessa (esim. prosessivuomalli, tilasiirtymämalli tai selväkieliset määritykset).

2.3.1 Toimintojen testaus (Toiminnallinen testaus) (K2)

Toiminnot, joita järjestelmä, alijärjestelmä tai komponentin pitää suorittaa, voidaan kuvata vaihetuotteissa, kuten vaatimusmäärittelyissä, käyttötapauksissa, tai toiminnallisessa määrittelyssä, tai ne voivat olla dokumentoimattomia. Toiminnot ovat se, "mitä" järjestelmä tekee.

Toiminnalliset testit perustuvat toimintoihin ja ominaisuuksiin (dokumenteissa kuvattuihin tai testaajien ymmärtämiin) ja niiden yhteentoimivuuteen tiettyjen järjestelmien kanssa, ja niitä voidaan suorittaa kaikilla testitasoilla (esimerkiksi komponenteille tehdyt testit voivat perustua komponenttimäärittelyyn).

Määrittelypohjaisia tekniikoita voidaan käyttää testattavien tilanteiden ja testitapauksien johtamiseen ohjelmiston tai järjestelmän toiminnallisuuden perusteella. (Katso luku 4) Toiminnallinen testaus tutkii ohjelmiston ulkoista käyttäytymistä (mustalaatikkotestaus).

Yksi toiminnallisen testauksen tyyppi, tietoturvatestaus, tutkii toimintoja (esim. palomuuri), jotka liittyvät vihamielisten ulkopuolisten aiheuttamien uhkien, kuten virusten, löytämiseen. Toinen toiminnallisen testauksen tyyppi, yhteentoimivuustestaus, arvioi ohjelmistotuotteen kykyä toimia yhdessä yhden tai useamman määritetyn komponentin tai järjestelmän kanssa.

2.3.2 Ei-toiminnallisten ominaisuuksien testaus (ei-toiminnallinen testaus) (K2)

Ei-toiminnalliseen testaukseen kuuluvat suorituskykytestaus, kuormitustestaus, rasitustestaus, käytettävyystestaus, ylläpitotestaus, luotettavuustestaus ja siirrettävyystestaus, mutta ei se rajoitu pelkästään näihin. Siinä testataan, "kuinka" järjestelmä toimii.

Ei-toiminnallista testausta voidaan tehdä kaikilla testitasoilla. Termi ei-toiminnallinen testaus kuvaa testit, joita tarvitaan mittaamaan järjestelmien ja ohjelmistojen piirteitä, joita voidaan mitata muuttuvalla asteikolla, kuten vasteaikoja suorituskykytestauksessa. Näitä testejä voidaan pohjata laatumalliin, kuten esimerkiksi

'Software Engineering – Software Product Quality' (ISO 9126)':ssa esiteltyyn. Ei-toiminnallinen testaus tutkii ohjelmiston ulkoista käyttäytymistä ja käyttää useimmissa tapauksissa mustalaatikkotekniikoita siihen tarvittavien testien suunnittelussa.

2.3.3 Ohjelmiston rakenteen/arkkitehtuurin testaus (rakenteellinen testaus) (K2)

Rakenteellista (lasilaatikko)testausta voidaan suorittaa kaikilla testitasoilla. Rakenteelliset tekniikat ovat hyödyllisimpiä määrittelypohjaisten tekniikoiden jälkeen, jolloin niiden avulla voidaan mitata testauksen perusteellisuutta rakenteeseen pohjautuvan kattavuuden arvioinnin kautta.

Kattavuus kertoo laajuuden, jolla testijoukko on käynyt läpi ohjelmiston osan, ja se ilmaistaan prosentteina katettavista asioista. Jos kattavuus ei ole 100 %, voidaan suunnitella lisää testejä testaamaan kohdat, joita ei vielä ollut käyty läpi, ja siten lisätä kattavuutta. Kattavuustekniikoita käsitellään luvussa 4.

Kaikilla testitasoilla, mutta erityisesti komponentti- ja komponentti-integrointitestauksessa, voidaan käyttää työkaluja mittaamaan elementtien, kuten lauseiden tai päätöksien, koodikattavuutta. Rakenteellinen testaus voi pohjautua järjestelmän arkkitehtuuriin, kuten kutsuhierarkiaan.

Rakenteellisia testauksen lähestymistapoja voidaan soveltaa järjestelmä-, järjestelmäintegraatio- ja hyväksymistestaustasoilla (esim. liiketoimintamalleihin tai valikkorakenteisiin).

2.3.4 Muutosten testaus (varmistustestaus (uudelleentestaus) ja regressiotestaus) (K2)

Vian havaitsemisen ja korjauksen jälkeen ohjelmisto pitää testata uudestaan sen varmentamiseksi, että alkuperäinen vika on onnistuneesti poistettu. Tätä kutsutaan varmistamiseksi. Virheidenpoisto (vian korjaus) on kehitysaikainen tehtävä, ei testaustehtävä.

Regressiotestauksessa testataan jo testattu ohjelma uudelleen muutoksen jälkeen, jotta löydetään muutosten seurauksena syntyneet tai paljastuneet viat. Näitä vikoja voi olla joko testattavassa ohjelmistossa, tai toisessa siihen liittyvässä tai erillisessä ohjelmistokomponentissa. Tätä testausta suoritetaan, kun ohjelmisto tai sen ympäristö muuttuu. Regressiotestauksen laajuus perustuu siihen, kuinka suuri on riski, että aikaisemmin toimineessa ohjelmistossa on vikoja, joita ei löydetä.

Testien pitää olla toistettavissa, jos niitä on tarkoituksena käyttää varmistustestauksessa ja regressiotestauksen apuna.

Regressiotestausta voidaan suorittaa kaikilla tasoilla, ja siihen kuuluu toiminnallista, ei-toiminnallista ja rakenteellista testausta. Regressiotestijoukkoja ajetaan monta kertaa ja ne yleisesti kehittyvät hitaasti, joten regressiotestaus on vahva ehdokas automatisoitavaksi.

2.4 Ylläpitotestaus (K2)	15 minuuttia

Termit

Vaikutusanalyysi, ylläpitotestaus.

Tausta

Käyttöönoton jälkeen ohjelmisto on usein käytössä vuosia tai vuosikymmeniä. Tänä aikana järjestelmää ja sen ympäristöä usein korjataan, muutetaan tai laajennetaan. Julkaisujen suunnitteleminen etukäteen on elintärkeää ylläpitotestauksen onnistumiseksi. On tärkeää erottaa suunnitellut julkaisut ja hätäkorjaukset toisistaan. Ylläpitotestausta tehdään käytössä olevalle järjestelmälle, ja sen aiheuttavat muutokset, muunnokset tai ohjelmiston tai järjestelmän käytöstä poistaminen.

Muutoksiin kuuluvat suunnitellut päivitysmuutokset (esim. julkaisupohjaiset), korjaavat ja hätämuutokset sekä ympäristön muutokset, kuten suunnitellut käyttöjärjestelmä- tai tietokantapäivitykset, valmisohjelmistojen suunnitellut versionnostot tai korjaustiedostot uusien paljastuneiden tai löydettyjen käyttöjärjestelmän haavoittuvuuksien korjaamiseksi.

Muunnoksen (esim. yhdeltä alustalta toiselle) ylläpitotestauksen pitäisi sisältää sekä uuden ympäristön että muuttuneen ohjelmiston käyttötestejä. Muunnostestausta (konversiotestausta) tarvitaan, kun toisen sovelluksen tietoja siirretään ylläpidettävään järjestelmään.

Käytöstä poistuvan järjestelmän ylläpitotestaus voi sisältää tietojen migraation tai arkistoinnin testauksen, jos tarvitaan pitkäaikaista tietojen säilyttämistä.

Muuttuneiden alueiden testaamisen lisäksi ylläpitotestaukseen kuuluu muuttamattomien alueiden laaja regressiotestaus. Ylläpitotestauksen laajuus määräytyy muutoksen aiheuttamaan riskin, olemassa olevan järjestelmän koon ja muutoksen laajuuden perusteella. Muutoksista riippuen ylläpitotestausta voidaan tehdä millä tahansa tai kaikilla testitasoilla ja se voi kohdistua mihin tahansa tai kaikkiin testityyppeihin. Muutosten vaikutusten selvittämistä olemassa olevan järjestelmän kannalta kutsutaan vaikutusanalyysiksi, ja sitä käytetään apuna, kun päätetään, kuinka paljon regressiotestausta tulisi tehdä. Vaikutusanalyysiä voidaan käyttää regressiotestijoukon määrittelyssä.

Ylläpitotestaus voi olla vaikeaa, jos määrittelyasiakirjat ovat vanhentuneet tai puuttuvat, tai jos testaajat eivät tunne kyseistä sovellusaluetta.

Viitteet

2.1.3 CMMI, Craig, 2002, Hetzel, 1988, IEEE 12207

2.2 Hetzel, 1988

2.2.4 Copeland, 2004, Myers, 1979

2.3.1 Beizer, 1990, Black, 2001, Copeland, 2004

2.3.2 Black, 2001, ISO 9126

2.3.3 Beizer, 1990, Copeland, 2004, Hetzel, 1988

2.3.4 Hetzel, 1988, IEEE STD 829-1998

2.4 Black, 2001, Craig, 2002, Hetzel, 1988, IEEE STD 829-1998

3. Staattiset tekniikat (K2) — 60 minuuttia

Oppimistavoitteet staattisille tekniikoille

Oppimistavoitteet kuvaavat, mitä osaat tehdä kunkin jakson läpikäymisen jälkeen.

3.1 Staattiset tekniikat ja testausprosessi (K2)

LO-3.1.1 Tunnistat ohjelmiston vaihetuotteet, joita voidaan tutkia erilaisilla staattisilla tekniikoilla. (K1)
LO-3.1.2 Osaat kuvata staattisten tekniikoiden tärkeyden ja arvon ohjelmiston vaihetuotteiden arvioinnissa. (K2)
LO-3.1.3 Osaat selittää staattisten ja dynaamisten tekniikoiden erot ottaen huomioon tavoitteet, tunnistettavien vikojen tyypit ja näiden tekniikoiden roolin ohjelmiston elinkaaressa. (K2)

3.2 Katselmointiprosessi (K2)

LO-3.2.1 Tunnistat tyypillisen muodollisen katselmoinnin vaiheet, roolit ja vastuut. (K1)
LO-3.2.2 Pystyt selittämään erot erityyppisten katselmointien välillä: epämuodollinen katselmointi, tekninen katselmointi, läpikäynti ja tarkastus. (K2)
LO-3.2.3 Osaat kuvata katselmointien onnistumiseen vaikuttavat seikat. (K2)

3.3 Staattinen analyysi työkaluilla (K2)

LO-3.3.1 Tunnistat tyypilliset viat ja virheet, joita staattinen analyysi tunnistaa, ja osaat verrata niitä katselmointeihin ja dynaamiseen testaukseen. (K1)
LO-3.3.2 Osaat kuvata esimerkkien avulla tyypillisiä staattisen analyysin etuja. (K2)
LO-3.3.3 Osaat luetella tyypillisiä koodin ja suunnittelun vikoja, joita analysointityökaluilla voidaan tunnistaa. (K1)

3.1 Staattiset tekniikat ja testausprosessi (K2)	15 minuuttia

Termit

Dynaaminen testaus, staattinen testaus.

Tausta

Toisin kuin dynaaminen testaus, joka vaatii ohjelmiston suorittamista, staattiset testauksen tekniikat pohjautuvat koodin tai muun projektidokumentaation manuaalisen tutkimiseen (katselmointeihin) ja automaattiseen analyysiin (staattinen analyysi).

Katselmoinnit ovat tapa testata ohjelmiston vaihetuotteita (koodi mukaan luettuna) ja niitä voidaan suorittaa hyvissä ajoin ennen dynaamista testausta. Aikaisessa elinkaaren vaiheessa tehdyissä katselmoinneissa löydetyt viat (esim. vaatimuksista löydetyt viat) ovat usein paljon halvempia poistaa kuin ne, jotka löydetään testien suorituksessa.

Katselmointi voidaan tehdä täysin manuaalisena tehtävänä, mutta siihen on myös työkalutukea. Manuaalisen katselmoinnin päätehtävä on tutkia jotakin vaihetuotetta ja kommentoida sitä. Katselmoida voidaan mikä tahansa ohjelmiston vaihetuote, kuten vaatimuskuvaukset, suunnittelukuvaukset, koodi, testaussuunnitelmat, testisuunnitelmat, testitapaukset, testiskriptit, käyttöohjeet tai www-sivut.

Katselmointien hyötyihin kuuluvat aikainen vikojen löytäminen ja korjaus, kehityksen tuottavuuden parannukset, lyhentynyt kehitysaikataulu, vähentyneet testauksen kustannukset ja aikatarve, vähentyneet kustannukset ohjelmiston elinkaaren aikana, vikojen väheneminen ja parantunut kommunikaatio. Katselmoinnit voivat löytää esimerkiksi vaatimuksista puutteita, joita ei todennäköisesti löydetä dynaamisessa testauksessa.

Katselmoinneilla, staattisella analyysillä ja dynaamisella testauksella on sama tavoite - vikojen tunnistaminen. Ne ovat toisiaan täydentäviä: eri tekniikat voivat löytää erilaisia vikoja sisäisesti ja ulkoisesti tehokkaasti. Dynaamiseen testaukseen verrattuna staattiset tekniikat löytävät ennemminkin häiriöiden syitä (vikoja) kuin itse häiriöitä.

Tyypillisiä vikoja, joita löydetään katselmoinneissa helpommin kuin dynaamisessa testauksessa, ovat poikkeamat standardeista, viat vaatimuksissa ja suunnittelussa, riittämätön ylläpidettävyys ja väärät rajapintamääritykset.

3.2 Katselmointiprosessi (K2)	25 minuuttia

Termit

Aloitusehdot, epämuodollinen katselmointi, katselmoija, kirjuri, läpikäynti, metriikka, muodollinen katselmointi, tarkastus, tekninen katselmointi, vertaiskatselmus, vetäjä/tarkastuksen puheenjohtaja.

Tausta

Eri katselmointityypit vaihtelevat hyvin epämuodollisista (esim. ei kirjallisia ohjeita katselmoijille) hyvin muodollisiin, joille ominaista ovat tiimin osallistuminen, katselmoinnin tulosten dokumentointi ja dokumentoidut toimenpiteet katselmoinnin läpiviemiseksi. Katselmointiprosessin muodollisuuteen vaikuttavat eri tekijät, kuten kehitysprosessin kypsyys, lakiin tai säädöksiin pohjautuvat vaatimukset tai jäljitysketjun tarve.

Katselmoinnin suoritustapa riippuu sovitusta katselmoinnin tavoitteesta (esim. vikojen löytäminen, ymmärryksen saaminen, testaajien ja uusien tiimin jäsenten opettaminen tai keskustelu ja yhteispäätöksen tekeminen.)

3.2.1 Muodollisen katselmoinnin vaiheet (K1)

Tyypillisellä muodollisella katselmoinnilla on seuraavat päävaiheet:

1. Suunnittelu
    - Katselmointikriteerien määrittäminen
    - Henkilöiden valinta
    - Roolien jakaminen
2. Aloitus- ja lopetusehtojen määrittäminen muodollisemmille katselmointityypeille (esim. tarkastus)
    - Tarkasteltavien dokumentin osien valinta
3. Aloitus
    - Dokumenttien jakaminen
    - Tavoitteiden, prosessin ja dokumenttien selittäminen osallistujille
4. Aloitusehtojen tarkistaminen (muodollisissa katselmointityypeissä)
5. Yksilöllinen valmistautuminen
    - Valmistautuminen katselmointikokousta varten tarkastamalla dokumentit
6. Potentiaalisten vikojen, kysymysten ja kommenttien muistiinkirjaaminen
7. Tulosten tarkastelu/arviointi/kirjaaminen (katselmointikokous)
    - Keskustelu tai havaintojen listaaminen, mukaan luettuna päätökset tai pöytäkirja (muodollisemmissa katselmointityypeissä)
    - Vikojen listaaminen, suositukset vikojen käsittelyn suhteen, päätökset vikoihin liittyen
8. Tarkastelemalla, arvioimalla ja nauhoittamalla kokoukset tai seuraamalla sähköistä viestintää
9. Uusintatyö
10. Löydettyjen vikojen korjaaminen, jonka tyypillisesti tekee asiakirjan tekijä
    - Vikojen päivitetyn tilan kirjaaminen (muodollisissa katselmoinneissa)
11. Seuranta
    - Tarkistetaan, että viat on käsitelty
    - Kerätään mittaritiedot
12. Lopetusehtojen tarkistaminen (muodollisissa katselmointityypeissä)

3.2.2 Roolit ja vastuut (K1)

Tyypilliseen muodolliseen katselmointiin kuuluvat alla mainitut roolit:

- Johtaja: päättää katselmointien suorituksesta, varaa niille aikaa projektiaikatauluista ja päättää onko katselmoinnin tavoitteet saavutettu.
- (Katselmoinnin) vetäjä: henkilö, joka johtaa dokumentin tai dokumenttijoukon katselmoinnin. Tehtäviin kuuluvat katselmoinnin suunnittelu, kokouksen johtaminen sekä kokouksen jälkeinen seuranta. Tarvittaessa vetäjä voi toimia välittäjänä eri näkökulmien välillä, ja hän on usein henkilö, josta katselmoinnin onnistuminen riippuu.
- Tekijä: katselmoitavan dokumentin/dokumenttien kirjoittaja tai henkilö, jolla on päävastuu siitä/niistä.
- Katselmoijat: henkilöt, joilla on tietty tekninen tai liiketoiminnallinen tausta (kutsutaan myös tarkistajiksi). Tarvittavien alkuvalmisteluiden jälkeen he tunnistavat ja kuvaavat katselmoitavasta tuotteesta esiin tulleet löydökset (eli viat). Katselmoijat pitäisi valita edustamaan eri näkökulmia ja rooleja katselmointiprosessissa ja heidän pitäisi ottaa osaa katselmointikokouksiin.
- Kirjuri (tai tallentaja): dokumentoi kaikki löydökset, ongelmat ja avoimet asiat, jotka tunnistetaan kokouksen aikana.

Dokumenttien tarkastelu eri näkökulmista ja tarkistuslistojen käyttäminen voi tehdä katselmoinneista sisäisesti ja ulkoisesti tehokkaampia. Esimerkiksi roolipohjaiset (käyttäjä, ylläpitäjä, testaaja, käyttöoperaatiot) tarkistuslistat tai tarkistuslista tyypillisistä vaatimuksiin liittyvistä ongelmista voivat auttaa paljastamaan aiemmin löytämättömiä vikoja.

3.2.3 Katselmointityypit (K2)

Yksi dokumentti voi olla käydä läpi useamman kuin yhden katselmoinnin. Jos useampia kuin yhtä katselmointityyppiä käytetään, niiden järjestys voi vaihdella. Esimerkiksi epämuodollinen katselmointi voidaan järjestää ennen teknistä katselmointia, tai vaatimusmäärittelyt voidaan tarkastaa ennen niiden läpikäyntiä asiakkaan kanssa. Tavallisten katselmointityyppien pääpiirteet, vaihtoehdot ja tarkoitus ovat:

Epämuodollinen katselmointi

- Ei muodollista prosessia
- Voi sisältää pariohjelmointia tai teknisen johdon katselmoimassa suunnittelua ja koodia
- Tulokset voidaan dokumentoida
- Hyödyllisyys vaihtelee katselmoijan mukaan
- Päätavoite: edullinen tapa saada jonkinlaista hyötyä.

Läpikäynti

- Tekijän vetämä kokous
- Voi sisältää skenaarioiden käyttöä, kuivaharjoittelua, vertaisryhmän osallistumista
- Avoimia tilaisuuksia
    - Valinnaista: katselmoijien valmistautuminen etukäteen
    - Valinnaista: laaditaan katselmointiraportti, joka sisältää listan havainnoista
- Valinnaista: kirjuri (joka ei ole tekijä)
- Voi vaihdella käytännössä melko epämuodollisesta hyvin muodolliseen
- Päätarkoitus: oppiminen, ymmärryksen saaminen, vikojen löytäminen.

Tekninen katselmointi

- Dokumentoitu, määritelty vianetsimisprosessi, jossa ovat mukana kollegat ja tekniset asiantuntijat sekä valinnaisesti johdon edustus
- Voidaan suorittaa vertaiskatselmuksena ilman johdon osallistumista
- Ihannetilanteessa koulutetun vetäjän (ei tekijän) johtama

- Katselmoijien valmistautuminen ennen kokousta
- Valinnaista: tarkistuslistojen käyttö
- Laaditaan katselmointiraportti, jossa on lista havainnoista, arvio siitä, täyttääkö ohjelmisto sille asetetut vaatimukset, ja tarvittaessa suosituksia havaintoihin liittyen
- Voi vaihdella käytännössä melko epämuodollisesta hyvin muodolliseen
- Päätarkoitus: keskustella, tehdä päätöksiä, arvioida vaihtoehtoja, löytää vikoja, ratkaista teknisiä ongelmia ja tarkistaa yhdenmukaisuus määrittelyjen, suunnitelmien, säädösten ja standardien kanssa.

Tarkastus

- Koulutetun vetäjän (ei tekijän) johtama
- Tavallisesti vertaiskatselmus
- Määritetyt roolit
- Sisältää metriikoiden keruun
- Muodollinen prosessi, joka perustuu sääntöihin ja tarkistuslistoihin
- Määritellyt aloitus- ja lopetusehdot ohjelmistotuotteen hyväksymiselle
- Valmistautuminen ennen kokousta
- Tarkastusraportti; havaintolista
- Muodollinen seurantaprosessi
    - Valinnaista: prosessin parannusnäkökulma
- Valinnaista: lukija
- Päätavoite: löytää vikoja.

Läpikäynnit, tekniset katselmoinnit ja tarkastukset voidaan suorittaa vertaisryhmässä - samalta organisaatiotasolta olevien kollegoiden kesken. Tällaista katselmointia kutsutaan vertaiskatselmukseksi.

3.2.4 Katselmointien menestystekijät (K2)

Katselmoinnin onnistumiseen vaikuttavat seuraavat seikat:

- Jokaisella katselmoinnilla on selkeä edeltä määritelty tavoite.
- Katselmoinnin tavoitteen kannalta oikeat ihmiset ovat mukana.
- Testaajia arvostetaan katselmoijina, jotka tuovat lisäarvoa katselmointiin samalla, kun he oppivat tuotteesta, mikä auttaa heitä laatimaan testejä aikaisemmin.
- Vikalöydökset koetaan tervetulleina, ja ne esitetään objektiivisesti.
- Ihmis- ja psykologiset näkökulmat huomioidaan (esim. tehdään katselmoinnista tekijälle positiivinen kokemus)
- Katselmoinnissa vallitsee luottamuksen ilmapiiri; tulosta ei käytetä osallistujien arviointiin.
- Käytetään katselmointitekniikoita, jotka mahdollistavat asetettujen tavoitteiden saavuttamisen ja ovat sopivia ohjelmiston vaihetuotteiden tyypin ja tason sekä katselmoijien suhteen.
- Tarkistuslistoja tai rooleja käytetään soveltuvin osin tehostamaan vikojen tunnistamista.
- Järjestetään koulutusta katselmointitekniikoista, erityisesti muodollisemmista, kuten tarkastuksesta.
- Yrityksen johto tukee hyvää katselmointiprosessia (esim. järjestämällä riittävästi aikaa katselmointitehtäville projektien aikatauluissa.)
- Painopiste on oppimisessa ja prosessin parantamisessa.

3.3 Staattinen analyysi työkaluilla (K2)	20 minuuttia

Termit

Kompleksisuus, kontrollivuo, kääntäjä, staattinen analyysi, tietovirta.

Tausta

Staattisen analyysin tavoitteena on löytää vikoja ohjelmiston lähdekoodista ja ohjelmistomalleista. Staattinen analyysi suoritetaan ilman, että tosiasiassa ajetaan työkalulla tutkittavaa ohjelmistoa; dynaaminen testaus taas suorittaa ohjelmiston koodin. Staattinen analyysi voi löytää vikoja, joita on vaikea löytää testauksessa. Kuten katselmoinnit, staattinen analyysi löytää ennemmin vikoja kuin häiriöitä. Analysointityökalut analysoivat ohjelmakoodia (esim. kontrolli- ja tietovirtaa) samoin kuin ohjelmallisesti tuotettua koodia, kuten HTML- ja XML-kieltä.

Staattisen analyysin arvo on seuraavissa seikoissa:

- Löydetään vikoja aikaisin jo ennen testien suoritusta
- Saadaan mittaritietojen avulla aikaisia varoituksia koodin tai suunnittelun epäilyttävistä piirteistä, kuten esimerkiksi korkea kompleksisuusluku.
- Tunnistetaan vikoja, joita ei helposti löydy dynaamisessa testauksessa
- Löydetään riippuvuuksia ja epäjohdonmukaisuuksia ohjelmistomalleista, kuten linkit.
- Parannetaan koodin ja suunnittelun ylläpidettävyyttä.
- Ennaltaehkäistään vikoja, jos aiemmista kokemuksista otetaan kehityksessä opiksi.

Tyypillisiä vikoja, joita analysointityökaluilla löydetään, ovat:

- viittaaminen muuttujaan, jolle ei ole määritetty arvoa
- epäjohdonmukaisuudet moduulien ja komponenttien välisissä rajapinnoissa
- käyttämättömät tai väärin tai puutteellisesti määritellyt muuttujat
- saavuttamaton (kuollut) koodi
- puuttuva tai virheellinen logiikka (mahdolliset päättymättömät silmukat)
- huomattavan monimutkaiset rakenteet
- ohjelmointistandardien rikkominen
- tietoturvaheikkoudet
- koodin ja ohjelmistomallin syntaksivirheet.

Analysointityökaluja käyttävät tyypillisesti kehittäjät (tarkistaessaan työtään ennalta määriteltyjä sääntöjä tai ohjelmointistandardeja vastaan) sekä komponentti- ja integrointitestausta ennen että sen aikana ja suunnittelijat ohjelmiston mallinnuksen aikana. Analysointityökalut voivat tuottaa suuren määrän varoitusviestejä, jotka täytyy hallita hyvin, jotta työkalua voidaan käyttää mahdollisimman tehokkaasti.

Kääntäjät voivat tukea jonkin verran staattista analyysiä, esimerkiksi metriikoiden laskentaa.

Viitteet

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

4. Testisuunnittelutekniikat (K4) — 285 minuuttia

Oppimistavoitteet testisuunnittelutekniikoille

Oppimistavoitteet kuvaavat, mitä osaat tehdä kunkin jakson läpikäymisen jälkeen.

4.1 Testien kehitysprosessi (K3)

LO-4.1.1 Erotat testisuunnitelman, testitapaussuunnitelman ja testiproseduurin kuvaukset toisistaan. (K2)
LO-4.1.2 Tunnistat testattavan tilanteen, testitapauksen ja testiproseduurin erot. (K2)
LO-4.1.3 Pystyt arvioimaan testitapausten laatua tutkimalla, miten niihin on kuvattu selkeä jäljitettävyys vaatimuksiin sekä odotetut lopputulokset. (K2)
LO-4.1.4 Osaat muokata testitapauksista selkeärakenteisia testiproseduurikuvauksia, jotka ovat testaajan kannalta riittävän yksityiskohtaisella tasolla. (K3)

4.2 Testisuunnittelutekniikoiden luokittelu (K2)

LO-4.2.1 Tunnistat sekä määrittelypohjaisten (mustalaatikko) että rakenteeseen perustuvien (lasilaatikko) lähestymistapojen hyödyt testitapausten suunnittelulle ja pystyt luettelemaan kummankin tavanomaiset tekniikat. (K1)
LO-4.2.2 Osaat selittää määrittelypohjaisen, rakenteeseen perustuvan ja kokemusperusteisen testauksen piirteet ja niiden väliset erot. (K2)

4.3 Määrittelypohjaiset eli mustalaatikkotekniikat (K3)

LO-4.3.1 Osaat kirjoittaa testitapauksia annetuista ohjelmistomalleista käyttämällä ekvivalenssiositusta, raja-arvoanalyysiä, päätöstaulutestausta ja tilasiirtymätestausta. (K3)
LO-4.3.2 Ymmärrät jokaisen neljän tekniikan päätarkoituksen, millä tasoilla ja tyyppisessä testauksessa tekniikkaa voidaan käyttää, ja kuinka kattavuutta voidaan mitata. (K2)
LO-4.3.3 Ymmärrät, mitä käyttötapaustestaus on ja mitkä ovat sen hyödyt. (K2)

4.4 Rakenteeseen perustuvat eli lasilaatikkotekniikat (K4)

LO-4.4.1 Osaat kuvata, mitä koodikattavuus tarkoittaa ja miksi se on tärkeää. (K2)
LO-4.4.2 Osaat selittää käsitteet lause- ja päätöskattavuus ja pystyt perustelemaan, miksi niitä voidaan käyttää myös muilla tasoilla kuin komponenttitestauksessa (esim. liiketoimintaprosesseissa järjestelmätasolla). (K2)
LO-4.4.3 Pystyt kirjoittamaan testitapauksia annetuista kontrollivirroista käyttämällä lausetestausta ja päätöstestausta. (K3)
LO-4.4.4 Osaat arvioida lause- ja päätöskattavuutta valmiuden kannalta. (K4)

4.5 Kokemusperusteiset tekniikat (K2)

LO-4.5.1 Tunnistat syyt testitapausten kirjoittamiselle intuition, kokemuksen ja tavanomaisista vioista käytettävissä olevan tiedon perusteella. (K1)
LO-4.5.2 Osaat verrata kokemuspohjaisia tekniikoita määrittelypohjaisiin tekniikoihin. (K2)

4.6 Testaustekniikoiden valinta (K2)

LO-4.6.1 Pystyt luokittelemaan testisuunnittelutekniikat niiden käyttösoveltuvuuden mukaan testaustilanteen, testauksen pohjamateriaalin, mallien ja järjestelmän ominaisuuksien perusteella. (K2)

4.1 Testien kehitysprosessi (K3)	15 minuuttia

Termit

Jäljitettävyys, testin suorittamisen aikataulu, testiproseduurin kuvaus, testiskripti, testisuunnitelma, testisuunnittelu.

Tausta

Tässä luvussa kuvattu testien kehitysprosessi voi vaihdella hyvin epämuodollisesta, jossa on vähän tai ei lainkaan dokumentaatiota, hyvin muodolliseen (kuten on kuvattu jäljempänä). Muodollisuuden aste riippuu testauksen kokonaistilanteesta, johon kuuluvat testaus- ja kehitysprosessien kypsyys, aikarajoitteet, turvallisuuteen tai viranomaismääräyksiin liittyvät vaatimukset sekä mukana olevat henkilöt.

Testianalyysin aikana testauksen lähdedokumentaatio analysoidaan sen päättämiseksi, mitä testataan, eli tunnistetaan testattavat tilanteet. Testattavalla tilanteella tarkoitetaan asiaa tai tapahtumaa, joka voidaan todentaa yhdellä tai useammalla testitapauksella (esim. toiminto, tapahtuma, laatuominaisuus tai rakenteellinen elementti).

Jäljitettävyyden muodostaminen testattavista tilanteista takaisin määrityksiin ja vaatimuksiin mahdollistaa sekä vaikutusanalyysin, kun vaatimukset muuttuvat, että vaatimuskattavuuden määrittämisen testijoukolle. Testianalyysin aikana käytetään mm. tunnistettuihin riskeihin perustuvaa yksityiskohtaista testausstrategiaa testisuunnittelutekniikoiden valitsemiseksi (ks. luvusta 5 lisätietoa riskianalyysistä).

Testisuunnittelun aikana luodaan ja määritellään testitapaukset ja testiaineisto. Testitapaus muodostuu joukosta syötearvoja, suorituksen esiehdoista, odotetuista tuloksista ja suorituksen jälkiehdoista, jotka on suunniteltu kattamaan yksi tai useampia määrättyjä testauksen tavoitteita tai testattavia tilanteita. Standardi ‘Standard for Software Test Documentation’ (IEEE 829-1998) kuvaa testattavien tilanteiden kuvauksen (test design specification) ja testitapausten kuvausten (test case specification) sisällöt.

Odotetut tulokset pitäisi tuottaa osana testitapauksen määrittelyä, ja niiden pitäisi sisältää tulosmuuttujat, muutokset tietoihin ja tiloihin sekä kaikki muut testauksen seuraukset. Jos odotettuja tuloksia ei ole määritelty, uskottavalta näyttävä mutta virheellinen tulos voidaan tulkita oikeaksi. Odotetut tulokset pitäisi ihannetilanteessa määritellä ennen testin suorittamista.

Testien toteutuksen aikana testitapaukset suunnitellaan, toteutetaan, priorisoidaan ja järjestetään testiproseduurin kuvaukseen (IEEE 829-1998). Testiproseduuri määrittelee testin suorittamisen toimenpiteiden sarjan. Jos testit ajetaan testin suoritustyökalulla, toimenpiteiden sarja määritellään testiskriptissä (joka on automatisoitu testiproseduuri).

Eri testiproseduureista ja automatisoiduista testiskripteistä muodostetaan lopulta testien suoritusaikataulu, joka määrittelee järjestyksen, jossa eri testiproseduurit ja mahdollisesti automatisoidut testiskriptit ajetaan, milloin ne suoritetaan ja kuka ne ajaa. Testien suoritusaikataulussa otetaan huomioon mm. regressiotestit, priorisointi, ja tekniset ja loogiset riippuvuudet.

4.2 Testisuunnittelutekniikoiden luokittelu (K2)	15 minuuttia

Termit

Kokemusperusteinen testisuunnittelutekniikka, lasilaatikkotestisuunnittelutekniikka, mustalaatikkotestisuunnittelutekniikka, testisuunnittelutekniikka.

Tausta

Testisuunnittelutekniikan tarkoitus on tunnistaa testattavat tilanteet, testitapaukset ja testiaineisto.

Testaustekniikat jaetaan perinteisesti mustalaatikko- ja lasilaatikkotekniikoihin. Mustalaatikkotekniikat (joita kutsutaan myös määrittelypohjaisiksi tekniikoiksi) ovat tapa johtaa ja valita testattavat tilanteet, testitapaukset tai testiaineisto testauksen lähdedokumenttien analyysin perusteella. Niihin kuuluu sekä toiminnallista että ei-toiminnallista testausta. Mustalaatikkotestaus ei oletusarvoisesti käytä tietoa testattavan komponentin tai järjestelmän sisäisestä rakenteesta. Lasilaatikkotekniikat (joita kutsutaan myös rakenteellisiksi tai rakenteeseen perustuviksi tekniikoiksi) perustuvat komponentin tai järjestelmän rakenteen analyysiin. Mustalaatikko- ja lasilaatikkotestauksessa voidaan myös hyödyntää kehittäjien, testaajien ja käyttäjien kokemuksia, kun päätetään, mitä pitäisi testata.

Jotkin tekniikat kuuluvat selkeästi yhteen luokkaan; toisissa on elementtejä useammasta kuin yhdestä luokasta.

Tässä sertifikaattisisällössä kutsutaan määrittelypohjaisia tekniikoita mustalaatikkotekniikoiksi ja rakenteeseen pohjautuvia tekniikoita lasilaatikkotekniikoiksi. Näiden lisäksi käsitellään kokemusperusteisia tekniikoita.

Määrittelypohjaisille tekniikoille tyypillisiä piirteitä ovat seuraavat:

- Malleja, joko muodollisia tai epämuodollisia, käytetään ratkaistavan ongelman, ohjelmiston tai sen komponenttien määrittelyyn
- Näistä malleista testitapauksia voidaan määritellä systemaattisesti.

Rakenteeseen pohjautuville tekniikoille tyypillisiä piirteitä ovat seuraavat:

- Testitapausten määrittämisessä käytetään tietoa siitä, kuinka ohjelmisto on rakennettu (esimerkiksi koodi ja yksityiskohtaiset suunnittelutiedot).
- Olemassa olevien testitapausten tuoma ohjelmiston kattavuus voidaan mitata ja kattavuuden lisäämiseksi voidaan luoda systemaattisesti lisää testitapauksia.

Kokemuspohjaisille tekniikoille tyypillisiä piirteitä ovat seuraavat:

- Ihmisten tietämystä ja kokemusta käytetään testitapausten määrittämisessä.
- Testaajien, kehittäjien, käyttäjien ja muiden sidosryhmien tietämys ohjelmistosta, sen käytöstä ja ympäristöstä on yksi tiedon lähde.
- Tieto todennäköisistä vioista ja niiden jakautumisesta on toinen tiedon lähde.

4.3 Määrittelypohjaiset eli mustalaatikkotekniikat (K3)	**150 minuuttia**

Termit

Ekvivalenssiositus, käyttötapaustestaus, päätöstaulutestaus, raja-arvoanalyysi, tilasiirtymätestaus.

4.3.1 Ekvivalenssiositusmenetelmä (K3)

Ekvivalenssiosituksessa ohjelmiston tai järjestelmän syötteet jaetaan ryhmiin, joiden odotetaan ilmaisevan samanlaista käyttäytymistä, joten on todennäköistä, että ne käsitellään samalla tavalla. Ekvivalenssiosiot (tai -luokat) voidaan määritellä sekä kelvolliselle aineistolle että epäkelvolle aineistolle eli syötteille, jotka pitäisi hylätä. Myös tulosarvot, sisäiset arvot, aikariippuvaiset arvot (esim. ennen tai jälkeen tapahtuman) ja rajapintaparametrit (esim. integrointitestauksen aikana testattaviin integroitaviin komponentteihin liittyvät) voidaan jakaa ekvivalenssiluokkiin. Testit voidaan suunnitella kattamaan kaikki kelvolliset ja epäkelvot luokat. Ekvivalenssiositus soveltuu kaikille testauksen tasoille.

Ekvivalenssiositusta voidaan käyttää syöte- ja tuloskattavuuden saavuttamiseksi Sitä voidaan käyttää ihmisen antamiin syötteisiin, syötteisiin järjestelmän rajapintojen kautta tai rajapintaparametreihin integrointitestauksessa.

4.3.2 Raja-arvoanalyysi (K3)

Ekvivalenssiluokkien rajakohdat ovat paikkoja, joissa ohjelman käyttäytyminen muita alueita todennäköisemmin on väärää, joten rajat ovat alue, josta testaus todennäköisesti löytää vikoja. Luokan maksimi- ja minimiarvot ovat sen raja-arvoja. Raja-arvo kelvolliselle osiolle on kelvollinen raja-arvo; epäkelvon osion raja-arvo on epäkelpo raja-arvo. Testit voidaan suunnitella kattamaan sekä kelvolliset että epäkelvot raja-arvot. Kun testitapauksia suunnitellaan, valitaan testi jokaiselle raja-arvolle.

Raja-arvoanalyysiä voidaan käyttää kaikilla testitasoilla. Sitä on suhteellisen helppo soveltaa ja sen vikojenlöytämiskyky on korkea. Yksityiskohtaiset määrittelyt ovat avuksi, kun määritellään kiinnostavia rajoja.

Tätä tekniikkaa pidetään usein ekvivalenssiluokituksen tai muiden mustalaatikkotekniikoiden laajennuksena. Sitä voidaan käyttää ekvivalenssiluokkiin, jotka on laadittu näytöllä annettaville käyttäjän syötteille yhtä hyvin kuin esimerkiksi aika-alueille (esim. aikakatkaisu, tapahtumien nopeusvaatimukset) tai taulukkoalueille (esim. taulukon koko on 256*256).

4.3.3 Päätöstaulutestaus (K3)

Päätöstaulut ovat hyvä tapa kuvata järjestelmävaatimuksia, joissa on loogisia ehtoja, ja dokumentoida sisäistä järjestelmäsuunnittelua. Niitä voidaan käyttää tallentamaan monimutkaisia liiketoimintasääntöjä, joita järjestelmän tulee noudattaa. Kun päätöstauluja luodaan, määritykset analysoidaan ja järjestelmän ehdot ja toimenpiteet tunnistetaan. Syötteiden ehdot ja toimenpiteet on useimmiten ilmaistu niin, että ne voivat olla joko Tosia tai Epätosia (Boolean). Päätöstaulu sisältää eri tapahtumien ehdot, usein kaikkien syötteiden tosi-epätosi-yhdistelmät ja niistä seuraavat lopputulokset jokaiselle ehtoyhdistelmälle. Jokainen taulukon sarake vastaa liiketoimintasääntöä, joka määrittelee yksilöllisen ehtoyhdistelmän, ja siten johtaa sääntöön liittyvien toimenpiteiden suorittamiseen. Päätöstaulutestauksen yhteydessä tyypillisesti käytetty kattavuusstandardi on, että saraketta kohti on vähintään yksi testitapaus, mikä tyypillisesti tarkoittaa kaikkien ehtoyhdistelmien kattamista

Päätöstaulutestauksen vahvuus on siinä, että se luo ehtojen yhdistelmiä, joita ei ehkä muuten käytäisi läpi testauksen aikana. Sitä voidaan käyttää kaikissa tilanteissa, joissa ohjelmiston toiminta riippuu useista loogisista päätöksistä.

4.3.4 Tilasiirtymätestaus (K3)

Järjestelmä voi tuottaa erilaisia vastauksia riippuen senhetkisistä olosuhteista tai aikaisemmasta historiasta (järjestelmän tilasta). Tässä tapauksessa tämä järjestelmän piirre voidaan kuvata tilasiirtymäkaaviona. Se sallii testaajan tutkia ohjelmistoa sen tilojen, tilojen välisten siirtymien, tilasiirtymän käynnistävien syötteiden ja tapahtumien, sekä siirtymiä seuraavien toimenpiteiden suhteen. Testattavan järjestelmän tai objektin tilat ovat erillisiä, yksilöitävissä ja lukumäärältään rajallisia.

Tilataulukko kertoo tilojen ja syötteiden välisen suhteen, ja se voi paljastaa mahdollisia epäkelpoja siirtymiä.

Testejä voidaan suunnitella kattamaan tyypillinen tilasiirtymäjakso, kattamaan jokainen tila, käymään läpi jokainen siirtymä tai tietyt siirtymien sarjat, tai testaamaan epäkelpoja tilasiirtymiä.

Tilasiirtymätestausta käytetään paljon sulautettujen ohjelmistojen tuotannossa ja yleensä teknisessä automaatiossa. Tekniikat ovat kuitenkin myös sopivia, kun mallinnetaan liiketoiminnan osia, joilla on tietyt tilat, tai kun testataan näyttödialogien etenemistä (esim. Internet-sovelluksia tai liiketoimintaskenaarioita).

4.3.5 Käyttötapaustestaus (K2)

Testejä voidaan määritellä käyttötapauksista. Käyttötapaus kuvaa toimijoiden, kuten käyttäjien ja järjestelmän, välistä vuorovaikutusta, joka tuottaa hyötyä järjestelmän käyttäjälle. Käyttötapauksia voidaan kuvata abstraktilla tasolla (liiketoiminnan käyttötapaukset, teknologiasta riippumattomat, liiketoimintaprosessin tasolla) tai järjestelmätasolla (järjestelmäkäyttötapaukset järjestelmän toiminnallisella tasolla). Jokaisella käyttötapauksella on esiehdot, joiden on täytyttävä, jotta käyttötapaus voidaan suorittaa onnistuneesti. Jokainen käyttötapaus päättyy jälkiehtoihin, joita ovat havainnoitavat tulokset ja järjestelmän lopputila käyttötapauksen suorituksen jälkeen. Käyttötapauksella on yleensä valtavirtaskenaario (eli todennäköisin etenemispolku) sekä joskus vaihtoehtoisia haarautumia.

Käyttötapaukset kuvaavat "prosessivirtoja" järjestelmän läpi perustuen sen todennäköiseen käyttöön, joten käyttötapauksista luodut testitapaukset ovat hyödyllisimpiä löytämään vikoja prosessin etenemisestä järjestelmän oikean käytön aikana. Käyttötapaukset ovat erittäin hyödyllisiä, kun suunnitellaan hyväksymistestejä, joihin asiakas tai käyttäjä osallistuu. Ne auttavat myös paljastamaan integraatiovikoja, jotka johtuvat eri komponenttien vuorovaikutuksesta ja toisilleen aiheuttamasta häirinnästä, jota yksittäisen komponentin testaus ei paljastaisi. Testitapausten suunnitteluun käyttötapausten pohjalta voidaan yhdistää muita määrittelypohjaisia testaustekniikoita.

4.4 Rakenteeseen perustuvat eli lasilaatikkotekniikat (K4)	60 minuuttia

Termit

Koodikattavuus, lausekattavuus, päätöskattavuus, rakennepohjainen testaus.

Tausta

Rakennepohjainen testaus/lasilaatikkotestaus pohjautuu ohjelmiston tai järjestelmän tunnistettuun rakenteeseen, kuten voidaan seuraavista esimerkeistä nähdä:

- Komponenttitaso: ohjelmistokomponentin rakenne eli lauseet, päätökset, haarautumat tai jopa määrätyt polut
- Integrointitaso: rakenne voi olla kutsupuu (kaavio, jossa moduulit kutsuvat toisia moduuleita).
- Järjestelmätaso: rakenne voi olla valikkorakenne, liiketoimintaprosessi tai www-sivun rakenne.

Tässä osassa käsitellään kolmea koodiin liittyvää rakenteellista koodikattavuustekniikkaa, jotka perustuvat lauseisiin, haarautumiin ja päätöksiin. Päätöstestauksessa kontrollivuokaaviota voidaan käyttää havainnollistamaan jokaisen päätöksen vaihtoehtoja.

4.4.1 Lausetestaus ja -kattavuus (K4)

Yksikkötestauksessa lausekattavuudella arvioidaan, kuinka monta prosenttia suoritettavista lauseista testijoukko on käynyt läpi. Lausetestauksessa testitapaukset laaditaan tiettyjen lauseiden suorittamiseksi yleensä lausekattavuuden kasvattamiseksi.

Lausekattavuus määräytyy jakamalla testitapausten läpikäymien suoritettavien lauseiden määrä kaikkien testattavan koodin sisältämien suoritettavien lauseiden määrällä.

4.4.2 Päätöstestaus ja -kattavuus (K4)

Päätöskattavuudella, joka liittyy haaratestaukseen, arvioidaan, kuinka monta prosenttia päätöstuloksista (esim. IF-lauseen Tosi- ja Epätosi-vaihtoehdot) testijoukko on käynyt läpi. Päätöstestauksessa testitapaukset laaditaan sellaisiksi, että ne suorittavat tietyt päätöstulokset. Haaraumat muodostavat päätöskohtia ohjelmakoodissa.

Päätöskattavuus määräytyy jakamalla (suunniteltujen tai suoritettujen) testitapausten kattamien päätöstulosten määrä testattavan koodin sisältämien kaikkien mahdollisten päätöstulosten määrällä.

Päätöstestaus on kontrollivuotestauksen muoto, koska siinä luodaan määrätty kontrollivuo päätöskohtien läpi. Päätöskattavuus on vahvempi kuin lausekattavuus: 100 % päätöskattavuus takaa 100 % lausekattavuuden, mutta ei toisin päin.

4.4.3 Muut rakenteeseen perustuvat tekniikat (K1)

On olemassa vahvempia rakenteellisen kattavuuden tasoja kuin päätöskattavuus, esimerkiksi ehtokattavuus ja moniehtokattavuus.

Kattavuuden arviointia voidaan tehdä myös muilla testitasoilla. Esimerkiksi integrointitasolla testijoukon läpikäymien moduulien, komponenttien tai luokkien prosenttiosuus voidaan ilmaista moduuli-, komponentti- tai luokkakattavuutena.

Työkalutuki on hyödyllistä koodin rakenteellisessa testauksessa.

4.5 Kokemusperusteiset tekniikat (K2)	30 minuuttia

Termit

Tutkiva testaus, vikahyökkäys.

Tausta

Kokemuspohjaisessa testauksessa testit luodaan testaajien osaamisen ja intuition sekä samankaltaisista sovelluksista ja teknologioista kertyneen osaamisen perusteella. Kun näitä tekniikoita käytetään suunnitelmallisten tekniikoiden lisänä, ne voivat olla hyödyllisiä sellaisten testien löytämiseksi, joita ei helposti tunnisteta muodollisilla tekniikoilla, erityisesti kun niitä sovelletaan muodollisempien lähestymistapojen jälkeen. Tämä tekniikka voi kuitenkin testaajien kokemuksesta riippuen olla tehokkuudeltaan hyvin vaihteleva.

Yleisesti käytetty kokemuspohjainen tekniikka on virheenarvaus. Yleisesti testaajat ennakoivat vikoja kokemuksen perusteella. Järjestelmällinen lähestymistapa virheenarvaukseen on listata mahdollisia virheitä ja suunnitella testejä hyökkäämään näitä virheitä vastaan. Tätä suunnitelmallista lähestymistapaa kutsutaan vikahyökkäykseksi. Näitä vika- ja häiriölistoja voidaan laatia kokemuksen, saatavilla olevan vika- ja häiriötiedon pohjalta, sekä perustuen yleiseen tietämykseen siitä, miksi ohjelmisto epäonnistuu.

Tutkiva testaus on testausohjeessa lueteltuihin tavoitteisiin perustuvaa samanaikaista testien suunnittelua, suoritusta, testaustapahtumien tallennusta ja oppimista, joka suoritetaan sovituissa aikajaksoissa. Tämä lähestymistapa on hyödyllisin silloin, kun saatavilla on vain muutamia tai vaillinaisia määrittelyjä ja aikataulu on kireä, tai jos halutaan lisätä tai täydentää muuta, muodollisempaa testausta. Se voi myös toimia testausprosessin tarkistuskeinona auttamassa varmistamaan, että vakavimmat viat löydetään.

4.6 Testaustekniikoiden valinta (K2)	**15 minuuttia**

Termit

Ei erityistermejä

Tausta

Käytettävien testaustekniikoiden valinta riippuu monesta tekijästä, joihin kuuluvat järjestelmän tyyppi, sääntelystandardit, asiakas- tai sopimukselliset vaatimukset, riskin taso, riskin tyyppi, testaustavoite, saatavilla oleva dokumentaatio, testaajien osaaminen, aika ja budjetti, kehityksen elinkaari, käyttötapausmallit ja löydetyistä virhetyypeistä saatu aiempi kokemus.

Jotkut tekniikat soveltuvat paremmin tiettyihin tilanteisiin ja tietyille testitasoille, toiset soveltuvat kaikille testitasoille.

Testitapauksia laatiessaan testaajat käyttävät usein eri testitapausten yhdistelmiä, kuten prosessiin, sääntöihin ja aineistoon pohjautuvia tekniikoita varmistamaan, että testattava kohde testataan riittävän kattavasti.

Viitteet

4.1 Craig, 2002, Hetzel, 1988, IEEE STD 829-1998

4.2 Beizer, 1990, Copeland, 2004

4.3.1 Copeland, 2004, Myers, 1979

4.3.2 Copeland, 2004, Myers, 1979

4.3.3 Beizer, 1990, Copeland, 2004

4.3.4 Beizer, 1990, Copeland, 2004

4.3.5 Copeland, 2004

4.4.3 Beizer, 1990, Copeland, 2004

4.5 Kaner, 2002

4.6 Beizer, 1990, Copeland, 2004

5. Testauksenhallinta (K3) — 170 minuuttia

Oppimistavoitteet testauksenhallinnalle

Oppimistavoitteet kuvaavat, mitä osaat tehdä kunkin jakson läpikäymisen jälkeen.

5.1 Testausorganisaatio (K2)

LO-5.1.1 Tunnistat testauksen riippumattomuuden tärkeyden. (K1)
LO-5.1.2 Pystyt luettelemaan riippumattoman testauksen hyödyt ja haitat organisaation sisällä. (K2)
LO-5.1.3 Tunnistat testaustiimiä perustettaessa harkittavat eri tiimin jäsenten roolit. (K1)
LO-5.1.4 Tunnistat tyypilliset testauspäällikön ja testaajan tehtävät. (K1)

5.2 Testaussuunnittelu ja työmääräarviointi (K3)

LO-5.2.1 Tunnistat testaussuunnittelun eri tasot ja tavoitteet. (K1)
LO-5.2.2 Osaat kuvata lyhyesti testaussuunnitelman, testisuunnitelman ja testiproseduurin tarkoituksen ja sisällön standardin 'Standard for Software Test Documentation' (IEEE 829-1998) pohjalta. (K2)
LO-5.2.3 Erotat käsitteellisesti erilaiset testauksen lähestymistavat, kuten analyyttinen, mallipohjainen, suunnitelmallinen, prosessien/standardien mukainen, dynaaminen/heuristinen, neuvoa-antava ja regressionvastainen. (K2)
LO-5.2.4 Erotat toisistaan järjestelmän testauksen suunnittelun ja testauksen suorituksen aikataulun suunnittelun. (K2)
LO-5.2.5 Osaat laatia testausaikataulun annetulle testijoukolle, ottaen huomioon priorisoinnin ja tekniset sekä loogiset riippuvuudet. (K3)
LO-5.2.6 Osaat luetella testauksen valmistelun ja suorittamisen vaiheet, jotka pitää ottaa huomioon testaussuunnittelussa. (K1)
LO-5.2.7 Tunnistat tyypillisiä tekijöitä, jotka vaikuttavat testauksen työmäärään. (K1)
LO-5.2.8 Erotat kaksi käsitteellisesti erilaista arviointitapaa: Metriikka- ja asiantuntemuspohjainen lähestymistapa. (K2)
LO-5.2.9 Tunnistat/pystyt perustelemaan riittävät aloitus- ja lopetusehdot tietylle testitasolle ja joukolle testitapauksia (esim. integrointitestaus, hyväksymistestaus tai käytettävyystestauksen testitapaukset). (K2)

5.3 Testauksen edistymisen seuranta ja kontrollointi (K2)

LO-5.3.1 Tunnistat yleisimmät käytössä olevat metriikat, joilla seurataan testien valmistumista ja suorittamista. (K1)
LO-5.3.2 Pystyt selittämään ja vertailemaan testauksen raportoinnissa ja hallinnassa käytettäviä testausmittareita (esim. löydetyt ja korjatut viat, läpäistyt ja epäonnistuneet testit). (K2)
LO-5.3.3 Osaat kuvata testauksen yhteenvetoraportin tarkoituksen ja sisällön ‘Standard for Software Test Documentation’ (IEEE 829-1998) -standardin mukaan. (K2)

5.4 Kokoonpanon hallinta (K2)

LO-5.4.1 Osaat kuvata, kuinka kokoonpanonhallinta tukee testausta. (K2)

5.5 Riskit ja testaus (K2)

LO-5.5.1 Osaat kuvata riskin mahdollisena ongelmana, joka voi uhata yhden tai useamman sidosryhmän jäsenen projektin tavoitteen toteutumista. (K2)

LO-5.5.2 Muistat, että riskit määritellään (tapahtumisensa) todennäköisyyden ja vaikutuksen (vahinko, joka seuraa riskin toteutumisesta) perusteella. (K1)
LO-5.5.3 Erotat projektiriskit ja tuoteriskit toisistaan. (K2)
LO-5.5.4 Tunnistat tyypillisiä tuote- ja projektiriskejä. (K1)
LO-5.5.5 Pystyt kuvaamaan esimerkkien avulla, kuinka riskianalyysiä ja riskienhallintaa voidaan käyttää testauksen suunnittelussa. (K2)

5.6 Havaintojen hallinta (K3)

LO-5.6.1 Tiedät 'Standard for Software Test Documentation' (IEEE 829-1998) -standardiin pohjautuvan havaintoraportin sisällön. (K1)
LO-5.6.2 Osaat kirjoittaa havaintoraportin testauksen aikana havaitusta häiriöistä. (K3)

5.1 Testausorganisaatio (K2)	**30 minuuttia**

Termit

Testaaja, testauspäällikkö.

5.1.1 Testausorganisaatio ja riippumattomuus (K2)

Vikojen löytämisen tehokkuutta testauksessa ja katselmoinneissa voidaan parantaa käyttämällä riippumattomia testaajia. Vaihtoehdot riippumattomuudelle ovat seuraavat:

- Ei riippumattomia testaajia, kehittäjät testaavat itse oman koodinsa.
- Riippumattomat testaajat kehitystiimissä
- Organisaation sisäinen riippumaton testaustiimi tai ryhmä, joka raportoi projektin johdolle tai liikkeenjohdolle
- Riippumattomat testaajat liiketoimintaorganisaatiosta tai käyttäjäyhteisöstä
- Tiettyjä testauskohteita testaavat riippumattomat testausasiantuntijat, kuten käytettävyys-, tietoturva- tai sertifiointitestaajat (jotka sertifioivat ohjelmistotuotteen standardeja ja säädöksiä vastaan)
- Ulkoistettu tai organisaation ulkopuolinen riippumaton testaus.

Suurten, monimutkaisten tai turvallisuuskriittisten projektien testauksessa on yleensä parasta käyttää useita testaustasoja, joista riippumattomat testaajat testaavat osan tai jopa kaikki. Kehittäjät voivat osallistua testaukseen, varsinkin alemmilla tasoilla, mutta heidän objektiivisuuden puutteensa rajoittaa usein heidän tehokkuuttaan. Riippumattomilla testaajilla voi olla valta vaatia ja määritellä testausprosesseja ja sääntöjä, mutta testaajien pitäisi omaksua tällaisia prosessiin liittyviä rooleja vain, jos heille on annettu siihen selkeät valtuudet johdon taholta.

Riippumattoman testauksen hyödyt:

- Riippumaton testaaja näkee muita, erityyppisiä vikoja ja on vapaa ennakko-oletuksista.
- Riippumaton testaaja voi todentaa olettamukset, joita on tehty järjestelmän määrittely- ja toteutusvaiheiden aikana.

Haittapuoliin kuuluvat mm. seuraavat asiat:

- Irrallisuus kehitystiimistä (jos suhtaudutaan täysin itsenäisinä)
- Kehittäjät voivat menettää vastuuntuntonsa laadunvarmistuksen suhteen
- Riippumattomat testaajat voidaan nähdä pullonkaulana tai heitä voidaan syyttää julkaisun viivästymisestä.

Testaustehtäviä voivat tehdä ihmiset, jotka toimivat tietyssä testausroolissa, tai ne voi tehdä jonkun toisen roolin edustaja, kuten projektipäällikkö, laatupäällikkö, kehittäjä, liiketoiminnan tai toimialueen asiantuntija, tai infrastruktuurin tai IT-toimintojen edustaja.

5.1.2 Testauspäällikön ja testaajan tehtävät (K1)

Tässä sertifikaattisisällössä käsitellään kahta tehtävää, testauspäällikön ja testaajan. Tehtävät ja toimenpiteet, joita näissä rooleissa toimivat ihmiset tekevät, riippuvat projektin ja tuotteen tilanteesta sekä rooleja edustavista ihmisistä ja organisaatiosta.

Joskus testauspäällikköä kutsutaan testausvastaavaksi tai testauskoordinaattoriksi. Testauspäällikön roolissa voi toimia projekti-, kehitys-, laadunvarmistuspäällikkö tai testausryhmän päällikkö. Suuremmissa

projekteissa voi olla kaksikin roolia: testauspäällikkö ja testausvastaava. Tyypillisesti testauspäällikkö suunnittelee, seuraa ja kontrolloi testauksen toimia ja tehtäviä kuten on määritelty osassa 1.4.

Tyypillisiin testauspäällikön tehtäviin voivat kuulua seuraavat:

- Testausstrategian ja -suunnitelman koordinointi projektipäälliköiden ja muiden kanssa
- Projektin testausstrategian ja organisaation testauspolitiikan kirjoittaminen tai katselmointi
- Testausnäkökulman tuominen muihin projektin tehtäviin, kuten integroinnin suunnitteluun
- Testien suunnittelu - ottaen huomioon tilanteen ja ymmärtäen testaustavoitteet ja riskit - mukaan luettuna testausnäkökulmien valinta, tarvittavan työmäärän, panostuksen ja kustannusten arviointi, resurssien hankinta, testitasojen ja -kierrosten määrittäminen sekä havaintojen hallinnan suunnittelu
- Määrittelyn, valmistelun, toteutuksen ja testien suorituksen käynnistäminen, testituloksien seuranta ja lopetusehtojen tarkistaminen
- Suunnittelun mukauttaminen testitulosten ja etenemisen (joskus dokumentoitu tilanneraporteissa) perusteella sekä toimenpiteisiin ryhtyminen ongelmien selvittämiseksi
- Riittävän kokoonpanonhallinnan käyttöönotto testausmateriaalille jäljitettävyyden aikaansaamiseksi
- Sopivien metriikoiden käyttöönotto testauksen edistymisen mittaamiseksi sekä testauksen ja tuotteen laadun arvioimiseksi
- Sen päättäminen, mitä tulisi automatisoida, kuinka laajasti ja miten
- Testausta tukevien työkalujen valinta ja työkalujen käyttökoulutuksen järjestäminen testaajille
- Testausympäristön toteutuksesta päättäminen
- Testauksen yhteenvetoraporttien kirjoittaminen testauksen aikana kerätyn informaation pohjalta.

Tyypillisiin testaajan tehtäviin voivat kuulua seuraavat:

- Testaussuunnitelmien katselmointi ja niihin myötävaikuttaminen
- Käyttäjävaatimusten, määrittelyjen ja mallien analysointi, katselmointi ja arviointi testattavuuden varmistamiseksi
- Testisuunnitelmien luominen
- Testiympäristön pystyttäminen (usein yhteistyössä järjestelmähallinnan ja verkonhallinnan kanssa)
- Testiaineiston valmistelu ja hankinta
- Testien toteuttaminen kaikilla testitasoilla, testien suorittaminen ja kirjaaminen, tulosten arviointi ja odotetuista tuloksista poikkeamien dokumentointi
- Testauksenhallintatyökalujen ja testauksen seurantatyökalujen käyttäminen sovitulla tavalla
- Testien automatisointi (tarvittaessa kehittäjän tai testiautomaatioasiantuntijan tukemana)
- Komponenttien ja järjestelmien suorituskyvyn mittaaminen (soveltuvissa tilanteissa)
- Toisten laatimien testien katselmointi.

Ihmiset, jotka työskentelevät testianalyysin, testisuunnittelun, tiettyjen testityyppien tai testausautomaation parissa, voivat olla näiden roolien asiantuntijoita. Riippuen testitasosta sekä tuotteeseen ja projektiin liittyvistä riskeistä, eri ihmiset voivat ottaa testaajan roolin, jolloin säilytetään tietty riippumattomuuden aste. Tyypillisesti testaajat komponentti- ja integrointitasolla ovat kehittäjiä, hyväksymistasolla liiketoiminta-asiantuntijoita ja käyttäjiä ja käyttöön soveltuvuuden hyväksymistestauksessa operaattoreita.

5.2 Testaussuunnittelu ja työmääräarviointi (K3)	**40 minuuttia**

Termit

Testauksen lähestymistapa, testausstrategia.

5.2.1 Testaussuunnittelu (K2)

Tässä osassa käsitellään testaussuunnittelun tarkoitusta kehitys- ja toteutusprojekteissa sekä ylläpitotehtävissä. Suunnittelu voidaan dokumentoida projekti- tai kokonaistestaussuunnitelmassa sekä testitasojen, kuten järjestelmä- ja hyväksymistestaus, erillisissä testaussuunnitelmissa. Testaussuunnittelu-dokumenttien sisältö on kuvattu standardissa ‘Standard for Software Test Documentation’ (IEEE 829-1998).

Suunnitteluun vaikuttavat organisaation testauspolitiikka, testauksen laajuus, tavoitteet, riskit, rajoitteet, kriittisyys, testattavuus sekä resurssien saatavuus. Mitä pidemmälle projekti ja testaussuunnittelu edistyvät, sitä enemmän tietoa on saatavilla ja sitä tarkempia yksityiskohtia voidaan sisällyttää suunnitelmaan.

Testaussuunnittelu on jatkuva tehtävä ja sitä tehdään kaikissa elinkaariprosesseissa ja -tehtävissä. Palautetta testaustehtävistä käytetään muuttuvien riskien tunnistamisessa, jotta suunnittelua voidaan muokata.

5.2.2 Testaussuunnittelun tehtävät (K3)

Koko järjestelmän tai sen osan testaussuunnittelun tehtäviin voivat kuulua seuraavat:

- Testauksen laajuuden ja riskien määrittäminen ja testauksen tavoitteiden tunnistaminen
- Testauksen yleisen lähestymistavan määrittäminen, mukaan luettuna testitasot sekä aloitus- ja lopetusehdot
- Testaustehtävien integrointi ja koordinointi ohjelmiston elinkaaren tehtäviin (hankinta, täydentäminen, kehitys, tuotanto ja ylläpito)
- Päättäminen siitä, mitä testataan, mitkä roolit suorittavat testaustehtävät, kuinka testaustehtävät suoritetaan ja kuinka testitulokset arvioidaan
- Testianalyysin ja -suunnitteluntehtävien aikataulutus
- Testien toteutuksen, suorituksen ja arvioinnin aikataulutus
- Resurssien määrääminen määritetyille eri tehtäville
- Testausdokumentaation määrän, yksityiskohtaisuuden, rakenteen ja mallipohjien määrittäminen
- Metriikoiden valinta testien valmistelun ja suorituksen, vikojen selvityksen sekä riskien mittaamiseksi ja seuraamiseksi
- Testiproseduurien yksityiskohtaisuuden määrittäminen, jotta saatavilla on tarpeeksi tietoa toistettavien testien valmistelun ja suorituksen tueksi.

5.2.3 Aloitusehdot (K2)

Aloitusehdoissa määritellään, milloin testaus aloitetaan, kuten esimerkiksi testaustason alussa tai kun joukko testejä on valmis suoritettavaksi.

Tyypillisiin aloitusehtoihin voi kuulua seuraavia:

- Testiympäristö on käytettävissä ja valmis
- Testityökalut ovat valmiina käyttöön testiympäristössä
- Testattava koodi on saatavilla
- Testiaineisto on saatavilla.

5.2.4 Lopetusehdot (K2)

Lopetusehdot määrittelevät, milloin testaus lopetetaan, kuten testitason lopuksi tai kun testijoukko on saavuttanut määrätyn tavoitteen.

Tyypillisiin lopetusehtoihin voi kuulua seuraavia:

- testauksen perusteellisuutta kuvaavat mittarit, kuten koodin, toiminnallisuuden tai riskin kattavuus
- vikatiheyden arviot tai luotettavuusmittarit
- kulut
- jäljellä olevat riskit, kuten korjaamattomat viat tai testauskattavuuden puuttuminen tietyiltä alueilta
- aikataulut, esimerkiksi markkinoille julkaisuaika.

5.2.5 Testauksen työmääräarviointi (K2)

Testauksen työmäärän arviointiin on kaksi lähestymistapaa:

- metriikkalähtöinen lähestymistapa: testaustyömäärän arviointi aiempien tai samanlaisten projektien metriikoiden pohjalta tai tyypillisten arvojen perusteella
- asiantuntijalähtöinen lähestymistapa: tehtävien omistajan tai asiantuntijoiden tekemä arvio.

Kun testaustyömäärä on arvioitu, voidaan tunnistaa resurssit ja laatia aikataulu.

Testaustyömäärä voi riippua monista tekijöistä, kuten:

- tuotteen ominaisuudet: määrittelyiden tai muun testimalleissa käytettävän tiedon (eli testauksen lähdedokumentaation) laatu, tuotteen koko, ongelma-alueen monimutkaisuus, luotettavuus- ja tietoturvavaatimukset ja vaatimukset dokumentaatiolle
- kehitysprosessin ominaisuudet: organisaation vakaus, käytetyt työkalut, testausprosessi, mukana olevien ihmisten taidot ja aikataulupaineet
- testauksen lopputulokset: vikojen määrä ja tarvittavan uudelleen tekemisen määrä.

5.2.6 Testauksen lähestymistavat (Testausstrategiat) (K2)

Testauksen lähestymistapa tarkoittaa testausstrategian sopeuttamista määrättyyn projektiin. Lähestymistapa määritellään ja sitä tarkennetaan testaussuunnitelmissa ja testisuunnitelmissa. Siinä kuvataan tyypillisesti mm. (testaus)projektin tavoitteen ja riskiarvioinnin perusteella tehdyt päätökset. Lähestymistapoja määriteltäessä alkaa myös testausprosessin suunnittelu, testisuunnittelutekniikoiden ja käytettävien testaustyyppien valinta sekä aloitus- ja lopetusehtojen määrittely.

Valitut lähestymistavat riippuvat projektin tilanteesta ja valinnassa otetaan huomioon riskit, uhkat ja turvallisuus, käytettävissä olevat resurssit ja osaaminen, teknologia, järjestelmän luonne (esim. räätälöity järjestelmä vs. valmisohjelmisto), testauksen tavoitteet sekä lait ja säädökset.

Tyypillisiin lähestymistapoihin kuuluvat

- analyyttiset lähestymistavat, kuten riskipohjainen testaus, jossa testaus kohdistetaan alueille, joilla riski on suurin
- mallipohjaiset lähestymistavat, kuten stokastinen testaus, jossa käytetään tilastollista tietoa häiriömääristä (kuten luotettavuuden kasvumallit) tai järjestelmän käytöstä (kuten käyttöprofiilit)
- menetelmälliset lähestymistavat, kuten häiriöihin (mukaan luettuna virheenarvaus ja vikahyökkäykset), kokemukseen, tarkistuslistoihin ja laatuominaisuuksiin pohjautuvat lähestymistavat
- prosesseihin tai standardeihin perustuvat lähestymistavat, kuten teollisuusstandardeissa tai monissa ketterissä metodologioissa määritellyt lähestymistavat

- dynaamiset ja heuristiset lähestymistavat, kuten tutkiva testaus, jossa testaus on enemmän tapahtumiin reagoivaa kuin ennalta suunniteltua, ja jossa suoritus ja arviointi ovat yhtäaikaisia tehtäviä
- konsultatiiviset lähestymistavat, kuten ne, joissa testikattavuuden määrittelyä ohjaavat ensisijaisesti testiryhmän ulkopuolisten teknologia- ja/tai liiketoiminta-alueen asiantuntijoiden neuvot
- taantumista eli regressiota vastustavat lähestymistavat, kuten olemassa olevan testimateriaalin uudelleenkäyttö, toiminnallisten regressiotestien laaja automaatio ja vakiotestijoukot.

Eri lähestymistapoja voidaan yhdistellä; esimerkkinä riskipohjainen dynaaminen lähestymistapa.

5.3 Testauksen edistymisen seuranta ja kontrollointi (K2)	20 minuuttia

Termit

Häiriötiheys, testauksen kontrollointi, testauksen seuranta, testiraportti, vikatiheys

5.3.1 Testauksen edistymisen seuranta (K1)

Testauksen seurannan tarkoituksena on antaa palautetta ja näkyvyyttä testaustehtäviin. Seurattavaa tietoa voidaan kerätä käsin tai automaattisesti, ja sitä voidaan käyttää mittaamaan lopetuskriteereitä, kuten kattavuutta. Metriikoita voidaan käyttää edistymisen arvioinnissa suunniteltua aikatauluja ja budjettia vasten. Tyypillisiin testausmetriikoihin kuuluvat

- testitapausten valmisteluun käytetyn työmäärän prosenttiosuus (tai suunnitelluista testitapauksista toteutettujen prosenttiosuus)
- testiympäristön valmisteluun käytetyn työmäärän prosenttiosuus
- testitapausten suorittaminen (esim. ajettujen/ajamattomien testitapausten määrä, ja läpäistyt ja epäonnistuneet testitapaukset)
- vikatiedot (esim. vikatiheys, löydetyt ja korjatut viat, häiriötiheys ja uudelleentestauksen tulokset)
- vaatimusten, riskien tai koodin testauskattavuus
- testaajien subjektiivinen luottamus tuotteeseen
- testauksen määräpäivät
- testauksen kulut, mukaan luettuna kustannukset verrattuna seuraavan vian löytämisestä tai seuraavan testin suorituksesta saataviin etuihin.

5.3.2 Testauksen raportointi (K2)

Testauksen raportointi käsittelee ja kokoaa yhteen testausta käsittelevää tietoa, mm.

- mitä tapahtui testausjakson aikana, kuten päivämäärät, jolloin lopetuskriteerit täyttyivät
- analysoitua tietoa ja lukuja, jotka tukevat suosituksia tuleviksi toimenpiteiksi sekä niistä tehtäviä päätöksiä. Tällaisia toimenpiteitä ovat esimerkiksi jäljellä olevien vikojen arviointi, testauksen jatkamisen taloudellinen hyöty, jäljellä olevat riskit, ja luottamuksen taso testattuun ohjelmistoon.

Testauksen yhteenvetoraportin sisältö on kuvattu standardissa ‘Standard for Software Test Documentation’ (IEEE 829-1998).

Mittaritietoa pitäisi kerätä sekä testitasojen aikana että niiden lopussa, jotta voidaan arvioida

- testaustavoitteiden riittävyys kyseiselle testitasolle
- valittujen testauksen lähestymistapojen riittävyys
- testauksen tehokkuus suhteessa sen tavoitteisiin.

5.3.3 Testauksen kontrollointi (valvonta) (K2)

Testauksen kontrollointi kuvaa kaikki ohjaavat ja korjaavat toimenpiteet, jotka suoritetaan informaation ja kerättyjen ja raportoitujen metriikoiden seurauksena. Toimenpiteet voivat kattaa mitkä tahansa testaustehtävät ja vaikuttaa mihin tahansa muihin ohjelmiston elinkaaren aktiviteettiin tai tehtävään.

Esimerkkejä testauksen kontrolloinnin toimenpiteistä ovat

- päätösten tekeminen testien monitoroinnista saadun tiedon perusteella
- testien uudelleenpriorisointi, kun tunnistettu riski toteutuu (esim. ohjelmisto toimitetaan myöhässä)
- testausaikataulun muuttaminen sen perusteella, milloin testiympäristö on käytettävissä
- sellaisten aloitusehtojen asettaminen, joissa vaaditaan, että kehittäjä on testannut korjaukset uudelleen (uudelleentestaus) ennen kuin ne hyväksytään ohjelman koontiin.

5.4 Kokoonpanon hallinta (K2)	10 minuuttia

Termit

Kokoonpanon hallinta, versionhallinta.

Tausta

Kokoonpanon hallinnan tarkoitus on saada aikaan ohjelmiston tai järjestelmän tuotteiden (komponentit, tieto ja dokumentaatio) välille yhtenäisyys ja ylläpitää sitä läpi projektin ja tuotteen elinkaaren.

Testauksen kannalta kokoonpanon hallinta voi sisältää sen varmistamista, että

- kaikki testausmateriaalin osat ovat yksilöityjä ja versiohallinnan alaisia, niitä seurataan muutosten varalta, ja ne liittyvät toisiinsa sekä kehityskohteisiin (testiobjekteihin), jotta jäljitettävyys voidaan säilyttää läpi testausprosessin
- kaikkiin yksilöityihin dokumentteihin ja ohjelmiston osiin viitataan yksiselitteisesti testausdokumentaatiossa.

Testaajan kannalta kokoonpanon hallinta auttaa yksiselitteisesti yksilöimään (ja toistamaan) testatun asian, testausdokumentit, testit ja testikehyksen.

Kokoonpanon hallinnan toimenpiteet ja infrastruktuuri (työkalut) pitäisi valita, dokumentoida ja käynnistää testaussuunnittelun aikana.

5.5 Riskit ja testaus (K2)	30 minuuttia

Termit

Projektiriski, riski, riskipohjainen testaus, tuoteriski.

Tausta

Riski voidaan määritellä tapahtuman, vaaran, uhan tai tilanteen toteutumisen mahdollisuutena ja sen ei-toivottuina seurauksina eli mahdollisena ongelmana. Riskitaso määräytyy haitallisen tapahtuman todennäköisyyden ja vaikutuksen (tapahtumaa seuraavan vahingon) perusteella.

5.5.1 Projektiriskit (K2)

Projektiriskit ovat riskejä, jotka liittyvät projektin kykyyn saavuttaa sille asetetut tavoitteet. Tällaisia riskejä muodostavat

- organisatoriset tekijät:
  - osaamisen, koulutuksen ja henkilöstön puute
  - henkilöstöön liittyvät asiat
  - toimintatapoihin liittyvät seikat, kuten
    - ongelmat testaajien tavassa tuoda esiin tarpeitaan ja testaustuloksia
    - testauksesta ja katselmoinneista saadun tiedon hyödyntämättä jättäminen (esim. ei paranneta kehityksen tai testauksen käytäntöjä)
  - väärät asenteet tai odotukset testausta kohtaan (ei esimerkiksi arvosteta sitä, että testauksen aikana löytyy vikoja)
- tekniset seikat:
  - ongelmat oikeiden vaatimusten määrittelemisessä
  - missä määrin olemassa olevat rajoitteet estävät vaatimusten täyttämisen
  - testiympäristö ei ole ajoissa valmiina
  - myöhäiset aineistokonversiot, migraation suunnittelu sekä konversio-/migraatiotyökalujen kehitys ja testaaminen
  - suunnittelun, koodin, kokoonpanotietojen, testiaineiston ja testien huono laatu
- toimittajaan liittyvät seikat:
  - kolmannen osapuolen epäonnistuminen
  - sopimusasiat.

Analysoidessaan, hallitessaan ja pienentäessään näitä riskejä testauspäällikkö seuraa yleisesti vakiintuneita projektinhallinnan periaatteita. Standardissa 'Standard for Software Test Documentation' (IEEE 829-1998) määritellyssä testisuunnitelman pohjassa vaaditaan, että riskit ja varasuunnitelmat tuodaan esiin.

5.5.2 Tuoteriskit (K2)

Mahdollisia häiriöalueita (haitallisia tulevaisuuden tapahtumia ja uhkia) ohjelmistossa tai järjestelmässä kutsutaan tuoteriskeiksi, koska ne muodostavat riskin tuotteen laadulle. Näihin kuuluvat

- häiriöalttiin ohjelmiston toimittaminen
- mahdollisuus, että ohjelmisto/laitteisto voi aiheuttaa vahinkoa yksilölle tai yritykselle
- huonot ohjelmiston ominaisuudet (esimerkiksi toiminnallisuus, luotettavuus, käytettävyys ja suorituskyky)
- aineiston epäyhtenäisyys ja huono laatu (esim. migraatio-ongelmat, konversio-ongelmat, tiedonsiirto-ongelmat, aineistostandardien rikkomukset)

- ohjelmisto, joka ei suorita siltä odotettuja toimintoja.

Tietoa riskeistä käytetään, kun päätetään, mistä testaus aloitetaan ja mistä testataan enemmän; testausta käytetään vähentämään epäsuotuisten tapahtumien toteutumisen riskiä tai vähentämään epäsuotuisten seurausten vaikutusta.

Tuoteriskit muodostavat projektin onnistumiseen liittyvän erityisen riskityypin. Testaus riskienhallinta-tehtävänä tuottaa palautetta jäljelle jäävästä riskistä mittaamalla sekä kriittisten vikojen poistamisen että varasuunnitelmien tehokkuutta.

Riskipohjainen lähestymistapa testaukseen tuo ennakoivia tilaisuuksia vähentää tuoteriskien tasoja projektin alkuvaiheista lähtien. Siihen kuuluvat tuoteriskien tunnistaminen ja niiden käyttäminen ohjaamassa testisuunnittelua ja testien hallintaa, määrittelyä, valmistelua ja testien suorittamista. Riskipohjaisessa lähestymistavassa tunnistettuja riskejä voidaan käyttää

- käytettävien testaustekniikoiden määrittämisessä
- testauksen laajuuden määrittämisessä
- testauksen priorisoinnissa, jotta kriittiset viat löydetään niin aikaisin kuin mahdollista
- määrittämään, voidaanko muita kuin testaustehtäviä käyttää riskien vähentämiseen (esim. koulutuksen tarjoaminen kokemattomille suunnittelijoille).

Riskipohjainen testaus hyödyntää projektin sidosryhmien kollektiivista tietämystä ja näkemystä riskien sekä niiden käsittelyssä tarvittavien testaustasojen määrittämisessä.

Tuotehäiriön mahdollisuuden minimoimiseksi riskienhallinnan toimenpiteet tarjoavat ohjatun lähestymistavan, jolla voi

- arvioida (ja uudelleenarvioida säännöllisesti), mikä voi mennä pieleen (riskit)
- määritellä, mitkä riskit on tärkeää käsitellä
- toteuttaa riskien käsittelyssä tarvittavat toimenpiteet.

Lisäksi testaus voi tukea uusien riskien tunnistamista, voi auttaa määrittelemään, mitä riskejä pitäisi pienentää, ja se voi vähentää epävarmuutta riskeistä.

5.6 Havaintojen hallinta (K3)	40 minuuttia

Termit

Havaintojen hallinta, havaintojen kirjaus, havaintoraportti

Tausta

Koska yksi testauksen tavoitteista on löytää vikoja, poikkeamat todellisten ja odotettujen lopputulosten välillä täytyy kirjata havaintoina. Havainto tulee tutkia, jolloin se voi osoittautua viaksi. Havaintojen ja vikojen käsittelemiseksi pitää määrittää sopivat toimenpiteet. Havaintoja pitäisi seurata niiden löytämisestä ja luokittelusta lähtien aina korjaukseen ja ratkaisun varmentamiseen asti. Jotta kaikkia havaintoja voidaan hallita loppuun asti, organisaation pitää luoda prosessi ja säännöt luokittelulle.

Havaintoja voidaan kirjata ohjelmistotuotteen kehityksen, katselmoinnin, testauksen tai käytön aikana. Niissä voidaan tuoda esiin joko ongelmia koodissa tai toimivassa järjestelmässä tai missä tahansa dokumentaatiossa, mukaan luettuna vaatimukset, kehitysdokumentit, testausdokumentit ja käyttäjälle tarkoitetut tiedot, kuten "Ohjeet" tai asennusohjeet.

Havaintoraporteilla on seuraavat tavoitteet:

- tarjota kehittäjille ja muille osapuolille palautetta ongelmasta, jotta mahdollistetaan sen tunnistaminen, eristäminen ja korjaus siten kuin on tarpeellista
- tarjota testauspäälliköille keino seurata testattavan järjestelmän laatua ja testauksen etenemistä
- tarjota ideoita testausprosessin kehittämiseksi.

Havaintoraportin yksityiskohtia voivat olla

- raportointipäivämäärä, ilmoittajaorganisaatio ja raportin tekijä
- odotetut ja todelliset tulokset
- testauksen kohteen tunniste (kokoonpanon osa) ja ympäristö
- ohjelmiston tai järjestelmän elinkaariprosessin vaihe, jossa havainto huomattiin
- havainnon kuvaus, joka mahdollistaa tilanteen toistamisen ja ratkaisemisen, mukaan luettuna lokit, tietokannan sisältölistaukset tai näytönkaappaukset
- seurauksen laajuus tai aste sidosryhmien kannalta
- seurauksen vakavuus järjestelmän kannalta
- korjauksen kiireellisyys tai prioriteetti
- havainnon tila (esim. avoin, lykätty, kaksoiskappale, odottaa korjausta, odottaa uudelleentestausta, suljettu)
- johtopäätökset, suositukset ja hyväksynnät
- yleiset huomiot, kuten muut alueet, joihin havaintoa seuraava muutos voi vaikuttaa
- muutoshistoria, kuten projektiryhmän jäsenten suorittamien toimenpiteiden sarja havainnon eristämiseksi, korjaamiseksi ja korjauksen varmistamiseksi
- viitteet, mukaan luettuna testisuunnitelma, joka paljasti ongelman.

Havaintoraportin rakenne on myös kuvattu standardissa ‘Standard for Software Test Documentation’ (IEEE 829-1998).

Viitteet

5.1.1 Black, 2001, Hetzel, 1988

5.1.2 Black, 2001, Hetzel, 1988

5.2.5 Black, 2001, Craig, 2002, IEEE Std 829-1998, Kaner 2002

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

5.4 Craig, 2002

5.5.2 Black, 2001 , IEEE Std 829-1998

5.6 Black, 2001, IEEE Std 829-1998

6. Testauksen työkalutuki (K2) — 80 minuuttia

Oppimistavoitteet testauksen työkalutuelle

Oppimistavoitteet kuvaavat, mitä osaat tehdä kunkin jakson läpikäymisen jälkeen.

6.1 Testityökalujen tyypit (K2)

LO-6.1.1 Osaat luokitella erityyppiset testityökalut niiden käyttötarkoituksen sekä testauksen perusprosessin ja ohjelmiston elinkaaren tehtävien mukaisesti. (K2)
LO-6.1.3 Osaat selittää käsitteen testaustyökalu sekä testauksen työkalutuen tarkoituksen. (K2)

6.2 Työkalujen tehokas käyttö: mahdolliset edut ja riskit (K2)

LO-6.2.1 Pystyt tekemään yhteenvedon testausautomaation ja testauksen työkalutuen hyödyistä ja riskeistä. (K2)
LO-6.2.2 Muistat testauksen suoritustyökaluihin, staattiseen analyysiin ja testauksenhallintavälineisiin liittyvät erityisesti huomioon otettavat seikat. (K1)

6.3 Työkalun käyttöönotto organisaatiossa (K1)

LO-6.3.1 Pystyt esittämään pääperiaatteet, jotka liittyvät työkalun käyttöönottoon organisaatiossa. (K1)
LO-6.3.2 Osaat kuvata proof-of-concept -menettelyn tavoitteet työkalun arvioinnissa ja pilottivaiheen tavoitteet työkalun käyttöönotossa. (K1)
LO-6.3.3 Tiedostat, että hyvään työkalujen avulla saatavaan tukeen liittyy muitakin asioita kuin pelkästään työkalun hankinta. (K1)

6.1 Testityökalujen tyypit (K2)	45 minuuttia

Termit

Debuggaustyökalu, dynaamisen analyysin työkalu, havaintojen hallintatyökalu, katselmointityökalu, kattavuustyökalu, kokoonpanon hallintatyökalu, kuormitustestaustyökalu, mallinnustyökalu, mittausjärjestelyjen vaikutus, monitorointityökalu, rasitustestaustyökalu, staattisen analyysin työkalu, suorituskykytestaustyökalu, testauksenhallintatyökalu, testiaineiston valmisteluväline, testikehys, testin suoritustyökalu, testisuunnittelutyökalu, tietoturvatyökalu, vaatimustenhallintatyökalu, vertailija, yksikkötestauskehystyökalu.

6.1.1 Testauksen työkalutuen merkityksen ja tarkoituksen ymmärtäminen (K2)

Testaustyökaluja voidaan käyttää yhteen tai useampaan tehtävään, jotka tukevat testausta. Näihin kuuluvat

1. Työkalut, joita käytetään suoraan testauksessa, kuten testien suoritustyökalut, testiaineiston valmisteluvälineet ja vertailutyökalut
2. Työkalut, jotka auttavat testausprosessin hallinnassa, kuten testien, testien tulosten, aineiston, vaatimusten, havaintojen, vikojen jne. hallinnassa sekä testien suorituksen raportoinnissa ja monitoroinnissa käytettävät työkalut
3. Työkalut, joita käytetään tutkimiseen tai tiedonhankintaan (esim. työkalut, jotka monitoroivat sovelluksen toimintaa)
4. Mitkä tahansa työkalut, jotka auttavat testausta (tässä merkityksessä myös taulukkolaskentaohjelmisto on myös testaustyökalu).

Testauksen työkalutukea voidaan käyttää yhteen tai useampaan tarkoitukseen tilanteesta riippuen:

- Parannetaan testaustehtävien tehokkuutta automatisoimalla toistuvia tehtäviä, tai tuetaan manuaalisia testaustehtäviä, kuten testauksen ja testien suunnittelua, raportointia ja monitorointia
- Automatisoidaan tehtäviä, jotka vaativat huomattavasti resursseja, mikäli ne tehdään käsin (esim. staattinen testaus)
- Automatisoidaan tehtäviä, joita ei voida tehdä manuaalisesti (esim. laajamittainen asiakas-palvelin-sovellusten suorituskykytestaus)
- Lisätään testauksen luotettavuutta (esim. automatisoimalla isojen aineistojen vertailu tai simuloimalla käyttäytymistä)

Termiä “testikehys” käytetään myös usein alalla ainakin kolmessa merkityksessä:

- Uudelleenkäytettävät ja laajennettavat testikirjastot, joita voidaan käyttää testityökalujen rakentamiseen (kutsutaan myös testipetiksi)
- Automaatiosuunnittelun tyyppi (esim. aineisto-ohjattu, avainsana-ohjattu)
- Testauksen suorituksen kokonaisprosessi.

Tässä sertifikaattisisällössä termiä “testikehys” käytetään kahdessa ensin mainitussa merkityksessä, kuten on kuvattu kohdassa 6.1.6.

6.1.2 Testityökalujen luokittelu (K2)

On olemassa lukuisia työkaluja, jotka tukevat testauksen eri osa-alueita. Työkalut voidaan luokitella useiden eri kriteerien perusteella, kuten käyttötarkoitus, kaupalliset/ilmaiset/open-source/shareware-ohjelmat, käytetty teknologia jne. Tässä sertifikaattisisällössä työkalut on luokiteltu niiden tukemien testaustehtävien mukaan.

Jotkut työkalut tukevat selkeästi yhtä tehtävää, toiset tukevat useampaa kuin yhtä, mutta ne on luokiteltu sen tehtävän alle, johon ne liittyvät läheisimmin. Yksittäisen välinetoimittajan välineet, esimerkiksi sellaiset, jotka on suunniteltu toimimaan yhdessä, voi olla niputettu yhteen pakettiin.

Jotkut testityökalut voivat olla tunkeutuvia, jolloin työkalu itse voi vaikuttaa testin lopputulokseen. Esimerkiksi todellinen ajoitus voi olla poikkeava välineen suorittamien ylimääräisten komentojen vuoksi, tai koodikattavuuden tulokset voivat olla erilaisia. Tunkeutuvien työkalujen aiheuttamia seurauksia kutsutaan mittausjärjestelyjen vaikutukseksi.

Jotkut työkalut tarjoavat enemmän ohjelmistokehittäjille sopivaa tukea (esim. komponentti- ja komponentti-integrointitestauksessa). Tällaiset työkalut on merkitty kirjaimella ("K") alla olevissa luokitteluissa.

6.1.3 Testauksen ja testien hallinnan työkalutuki (K1)

Hallintatyökalut soveltuvat kaikkiin testaustehtäviin koko ohjelmiston elinkaaren ajan.

Testauksenhallintatyökalut
Nämä työkalut tarjoavat rajapinnan testien suoritukseen, vikojen seurantaan sekä vaatimustenhallintaan ja tukevat testauksen kohteiden määrällistä analysointia ja raportointia. Ne tukevat myös testauskohteiden jäljitettävyyttä vaatimusmäärittelyihin, ja niissä voi olla myös oma versionhallintatoiminto tai liittymä ulkoiseen versionhallintavälineeseen.

Vaatimustenhallintatyökalut
Näiden työkalujen avulla säilytetään vaatimuslausekkeita, yksittäisten vaatimusten attribuutteja (kuten prioriteetti), luodaan yksilölliset tunnisteet ja ne tukevat vaatimusten jäljitettävyyttä yksittäisiin testeihin. Nämä välineet voivat auttaa myös epäyhtenäisten tai puuttuvien vaatimusten tunnistamisessa.

Havaintojenhallintatyökalut (Vianhallintatyökalut)
Havaintojenhallintatyökalujen avulla säilytetään ja hallitaan havaintoraportteja, eli vikoja, häiriöitä, muutospyyntöjä tai havaittuja ongelmia ja poikkeamia, ja ne auttavat havaintojen elinkaaren hallinnassa. Ne saattavat tukea myös tilastollista analyysiä.

Kokoonpanonhallintatyökalut
Kokoonpanon hallintatyökalut eivät ole tarkasti ottaen testaustyökaluja, mutta ne ovat tarpeellisia testimateriaalin ja siihen liittyvän ohjelmiston säilyttämisessä ja versionhallinnassa erityisesti silloin, kun kootaan useampaa kuin yhtä käyttöjärjestelmien, kääntäjien, selaimien jne. muodostamaa laitteisto/ohjelmistokokoonpanoa.

6.1.4 Staattisen testauksen työkalutuki (K1)

Staattisen testauksen työkalut tarjoavat kustannustehokkaan keinon löytää enemmän vikoja kehitysprosessin aikaisemmassa vaiheessa.

Katselmointityökalut
Nämä työkalut tukevat katselmointiprosessia, tarkistuslistojen ja katselmointien ohjeistuksen käyttöä, ja niitä käytetään katselmointikommenttien, vikojen ja työmäärätietojen säilyttämisessä ja raportoinnissa. Jos kyseessä on suuri tai maantieteellisesti hajallaan oleva tiimi, näistä välineistä voi olla lisäapua, mikäli ne tarjoavat keinon verkkokatselmointiin.

Staattisen analyysin työkalut (K)
Nämä työkalut auttavat ohjelmistokehittäjiä ja testaajia löytämään vikoja ennen dynaamista testausta tarjoamalla tukea ohjelmointistandardien noudattamisen valvontaan (mukaan luettuna tietoturvallinen koodaus) sekä rakenteiden ja riippuvuuksien analysointiin. Ne voivat myös auttaa suunnittelussa tai riskianalyysissä tuottamalla mittaritietoa koodista (esim. monimutkaisuus).

Mallinnustyökalut (K)
Näitä työkaluja käytetään ohjelmistomallien kelpuuttamisessa (esim. relaatiotietokannan fyysinen tietomalli) paljastamalla epäjohdonmukaisuuksia ja löytämällä vikoja. Nämä työkalut voivat usein auttaa testitapausten luomisessa mallin perusteella.

6.1.5 Testien määrittelyn työkalutuki (K1)

Testisuunnittelutyökalut
Näitä työkaluja käytetään syötteiden tai suoritettavien testien ja/tai testioraakkelien luomisessa vaatimuksista, graafisesta käyttöliittymästä, suunnittelumalleista (tila-, tieto- tai oliomalli) tai lähdekoodista.

Testiaineiston valmisteluvälineet
Testiaineiston valmisteluvälineillä käsitellään tietokantoja, tiedostoja tai tiedonsiirtoa, kun laaditaan testien suorittamisessa käytettävää testiaineistoa, jotta varmistetaan aineiston lähdetietojen tietosuoja muokkaamalla aineisto tunnistamattomaksi.

6.1.6 Testien suorituksen ja tapahtumien kirjaamisen työkalutuki (K1)

Testien suoritustyökalut
Nämä työkalut mahdollistavat testien suorittamisen automaattisesti tai puoliautomaattisesti skriptauskielen avulla käyttämällä tallennettuja syötteitä ja odotettuja tuloksia, ja ne yleensä tuottavat testilokin jokaisesta testiajosta. Niitä voidaan myös käyttää testien nauhoittamiseen, ja yleensä niissä on tuki skriptikielen avulla tai käyttöliittymän kautta tapahtuvalle aineiston parametrisoinnille sekä testien muunlaiselle muokkaukselle.

Testipeti-/yksikkötestauskehystyökalut (K)
Testipeti auttaa komponenttien tai järjestelmän osan testauksessa simuloimalla ympäristöä, jossa kyseistä testauksen kohdetta tullaan käyttämään ja tarjoamalla puuttuvien osien tilalle tynkiä tai ajureita.

Vertailijat
Vertailijat tunnistavat eroja tiedostojen, tietokantojen tai testitulosten välillä. Tyypillisesti testien suoritustyökalut sisältävät dynaamisia vertailutyökaluja, mutta suorituksen jälkeinen vertailu voidaan tehdä myös erillisellä vertailijatyökalulla. Testin vertailija voi käyttää testioraakkelia, erityisesti jos se on automatisoitu.

Kattavuuden mittaustyökalut (K)
Nämä työkalut, jotka voivat käyttää joko tunkeutuvia tai ei-tunkeutuvia menetelmiä, mittaavat testijoukon läpikäymien määrätyntyyppisen koodirakenteiden prosenttiosuutta (esim. lauseet, haarat tai päätökset ja moduuli- tai funktiokutsut).

Tietoturvatestauksen työkalut
Näitä työkaluja käytetään ohjelmiston tietoturvaominaisuuksien arvioinnissa. Näitä ominaisuuksia ovat mm. ohjelmiston kyky suojella aineiston luottamuksellisuutta, yhtenäisyyttä, oikeellisuutta, käyttövaltuuksia, saatavuutta ja kiistämättömyyttä. Tietoturvatestauksen työkalut keskittyvät pääasiassa määrättyyn teknologiaan, alustaan ja tarkoitukseen.

6.1.7 Suorituskyky- ja monitorointityökalut (K1)

Dynaamisen analyysin työkalut (K)
Dynaamisen analyysin työkalut löytävät vikoja, jotka ilmenevät vain ohjelmiston suorituksen aikana, kuten aikariippuvuudet tai muistivuodot. Niitä käytetään tyypillisesti komponenttitestauksessa ja komponentti-integrointitestauksessa sekä väliohjelmiston testauksessa.

Suorituskyky-/kuormitus-/rasitustestaustyökalut
Suorituskykytestaustyökalut monitoroivat ja raportoivat, kuinka järjestelmä käyttäytyy erilaisissa simuloiduissa käyttötilanteissa yhtäaikaisten käyttäjien määrän, käyttäjämäärän kasvatuskuvion, säännöllisyyden ja tapahtumien suhteellisen prosenttiosuuden suhteen. Kuorman simulointi toteutetaan luomalla määrättyjä toimintoja suorittavia virtuaalikäyttäjiä, jotka sijoitetaan useille testikoneille, joita kutsutaan tavallisesti kuormageneraattoreiksi.

Monitorointityökalut
Monitorointityökalut analysoivat ja todentavat jatkuvasti määrättyjen järjestelmäresurssien käyttöä ja raportoivat siitä, ja varoittavat mahdollisista palvelun ongelmista.

6.1.8 Erityissovellusalueiden työkalutuki (K1)

Aineiston laadun arviointi
Aineisto on keskeinen tekijä esimerkiksi konversio- tai migraatioprojekteissa sekä tietovarastoissa ja vastaavissa laajoissa sovelluksissa. Joskus testausaineisto on toiminnan keskeinen tekijä, kuten esimerkiksi konversio- tai migraatioprojekteissa tai esimerkiksi tietovarastosovelluksissa, ja sen ominaisuudet voivat vaihdella kriittisyyden ja määrän suhteen. Tällaisissa tilanteissa työkaluja tarvitaan aineiston laadun arvioimiseksi aineiston katselmoinnissa ja konversio- ja migraatiosääntöjen todentamisessa sen varmistamiseksi, että käsitelty aineisto on virheetön, prosessoitu kokonaan ja vastaa ennalta määritettyjä tilannekohtaisia standardeja.

Muita testaustyökaluja on olemassa käytettävyystestausta varten.

6.2 Työkalujen tehokas käyttö: mahdolliset edut ja riskit (K2)	20 minuuttia

Termit

Aineisto-ohjattu testaus, avainsanaohjattu testaus, skriptauskieli.

6.2.1 Mahdolliset edut ja riskit työkaluja käytettäessä (kaikille työkaluille) (K2)

Pelkästään työkalun ostaminen tai vuokraaminen ei takaa menestystä työkalun kanssa. Jokainen työkalutyyppi voi vaatia lisätyötä todellisten ja kestävien etujen saavuttamiseksi. Työkalujen käyttämisessä testauksessa on mahdollisia hyötyjä ja mahdollisuuksia, mutta myös riskejä.

Mahdollisia etuja työkalujen käytöstä ovat mm.

- toistuvan työn väheneminen (esim. regressiotestien ajaminen, saman testiaineiston uudelleensyöttäminen ja ohjelmointistandardien tarkistaminen)
- parempi yhdenmukaisuus ja toistettavuus (esim. työkalulla samassa järjestyksessä ja samaa suoritustiheyttä noudattaen suoritettavat testit sekä vaatimuksista johdetut testit)
- puolueeton arviointi (esim. staattiset mittarit, kattavuus)
- helppo tiedonsaanti testeistä tai testauksesta (esim. tilastot ja kaaviot testauksen edistymisestä, havaintomääristä ja suorituskyvystä).

Työkalujen käytön riskeihin kuuluvat:

- epärealistiset oletukset työkalulle (mukaan luettuna toiminnallisuus ja helppokäyttöisyys)
- työkalun käyttöönottoon tarvittavan ajan, kulujen ja työmäärän (mukaan luettuna koulutus ja ulkopuolinen asiantuntemus) aliarviointi
- työkalusta saatavien merkittävien ja jatkuvien hyötyjen saamiseksi tarvittavan ajan ja panostuksen aliarviointi (esim. testausprosessin muutostarpeet ja jatkuvat parannukset työkalun käyttötapoihin)
- työkalun tuomien testaushyötyjen ylläpidon vaatiman työmäärän aliarviointi
- liian suuri luottamus työkaluun (työkalu testisuunnittelun korvaajana tai työkalun käyttö, kun manuaalinen testaus olisi soveltuvampaa)
- työkalun sisältämän testausmateriaalin versionhallinnan laiminlyönti
- Yhteensopivuus- ja yhdessätoimivuusasioiden laiminlyönti kriittisten työkalujen kohdalla, joita ovat esimerkiksi vaatimustenhallintatyökalut, versionhallintatyökalut, havaintojenhallintatyökalut, virheidenhallintatyökalut, sekä usean toimittajan toimittamat työkalut
- työkalutoimittajan toiminnan lakkaaminen, välineen poistuminen markkinoilta tai myynti toiselle toimittajalle
- työkalutoimittajan huono vaste tukipyyntöihin, päivityksiin ja vikojen korjauksiin
- vapaan koodin tai ilmaisohjelmaprojektien lykkääntyminen
- odottamattomat tilanteet, kuten toimimattomuus uudelle käyttöalustalle siirron jälkeen.

6.2.2 Huomioon otettavia asioita joillekin työkalutyypeille (K1)

Testien suoritustyökalut
Testien suoritustyökalut suorittavat testejä käyttämällä automatisoituja testiskriptejä. Tällaiset työkalut vaativat usein huomattavan panostuksen, jotta niillä saadaan merkittäviä hyötyjä.

Testien tallentaminen nauhoittamalla testaajan manuaalisia toimenpiteitä voi vaikuttaa houkuttelevalta, mutta tämä menetelmä ei skaalaudu suureen määrään automatisoituja testejä. Nauhoitettu skripti on lineaarinen

esitys suoritetusta testistä aineistoineen ja toimenpiteineen. Tällainen skripti voi olla epävakaa odottamattomien tapahtumien sattuessa.

Aineisto-ohjatussa lähestymistavassa testisyötteet (testiaineisto) talletetaan erikseen, tavallisesti laskenta-taulukkoon, ja siinä käytetään yleisempää skriptiä, joka pystyy lukemaan testiaineiston ja suorittamaan saman testin eri aineistoilla. Testaajat, jotka eivät tunne skriptauskieltä, voivat laatia testiaineistoa näille ennalta tehdyille skripteille.

Aineisto-ohjatuissa tekniikoissa käytetään myös tekniikoita, missä taulukkoon tallennettujen kovakoodattujen aineistoyhdistelmien sijaan aineisto luodaan muokattavien parametrien perusteella algoritmien avulla suorituksen aikana ja välitetään sovellukselle. Työkalu voi käyttää esimerkiksi algoritmia, joka luo satunnaisen käyttäjätunnuksen, ja satunnaisuuden muoto perustuu alussa määritettyyn alkuarvoon.

Avainsanaohjatussa lähestymistavassa taulukko sisältää avainsanat (joita kutsutaan myös toimisanoiksi), jotka kuvaavat suoritettavat toimenpiteet, sekä testiaineiston. Testaajat (vaikka eivät tuntisikaan skriptaus-kieltä) voivat luoda testit käyttämällä avainsanoja, joita voidaan räätälöidä testattavan sovelluksen mukaisesti.

Skriptauskielen teknistä osaamista tarvitaan kaikissa lähestymistavoissa (joko testaajilta tai testaus-automaation asiantuntijoilta).

Käytetystä skriptaustekniikasta riippumatta jokaisen testin odotetut tulokset täytyy tallentaa myöhempää vertailua varten.

Staattisen analyysin työkalut
Staattisen analyysin työkalut voivat valvoa koodausstandardien noudattamista, mutta olemassa olevaan koodin käytettyinä ne voivat luoda paljon ilmoituksia. Varoitukset eivät estä koodin kääntämistä ajettavaksi ohjelmaksi, mutta ne pitäisi kuitenkin ottaa huomioon tulevan ylläpidon helpottamiseksi. Suodattimien asteittainen käyttöönotto joidenkin viestien poissulkemiseksi voisi olla tehokas lähestymistapa.

Testauksenhallintatyökalut
Testauksenhallintatyökaluissa pitää olla rajapinta toisiin työkaluihin tai taulukkolaskelmaohjelmiin, jotta hyödyllistä tietoa voidaan tuottaa muodossa, joka sopii organisaation tarpeisiin.

6.3 Työkalun käyttöönotto organisaatiossa (K1)	15 minuuttia

Termit

Ei erityistermejä

Tausta

Seuraavassa on lueteltu tärkeimpiä asioita, joita on otettava huomioon, kun organisaatioon ollaan valitsemassa testaustyökalua:

- organisaation kypsyyden, vahvuuksien ja heikkouksien arviointi sekä oikean hetken tunnistaminen työkalujen tukeman parannetun testausprosessin käyttöönotolle
- arviointi selkeitä vaatimuksia vastaan ja objektiivisten kriteereiden perusteella
- proof-of-concept -menettely eli välineen käyttö arviointivaiheessa sen määrittämiseksi, toimiiko se tehokkaasti testattavan ohjelmiston kanssa kyseisessä ympäristössä tai mahdollisten ympäristöön tarvittavien muutosten tunnistamiseksi, jotta välinettä voidaan käyttää tehokkaasti
- välinetoimittajan arviointi (mukaan luettuna tarjottu koulutus, tuki sekä kaupalliset näkökulmat) tai ei-kaupallisten työkalujen kohdalla muiden tukipalvelutarjoajien arviointi
- työkalun käytön tuomien sisäisten valmennus- ja ohjausvaatimusten tunnistaminen
- koulutustarpeiden arviointi ottaen huomioon nykyisen testaustiimin automatisointitaidot
- panos-tuotos-suhteen arviointi todellisen liiketoimintacasen perusteella.

Työkalun käyttöönotto organisaatiossa alkaa pilottiprojektilla, jolla on seuraavat tavoitteet:

- oppia lisää yksityiskohtia työkalusta
- arvioida, kuinka työkalu soveltuu olemassa oleviin prosesseihin ja käytäntöihin, ja päättää, mitä pitäisi muuttaa
- päättää työkalun ja testimateriaalin käytössä, hallinnassa, tallentamisessa ja ylläpidossa noudatettavista vakiokäytännöistä (esim. tiedostojen ja testien nimeämiskäytännöt, kirjastojen luominen ja testijoukkojen modulaarisuuden määrittäminen)
- sen arvioiminen, saadaanko työkalusta hyötyä kohtuullisilla kustannuksilla.

Työkalun käyttöönoton menestystekijöihin organisaatiossa kuuluvat:

- vaiheittainen työkalun käyttöönotto läpi organisaation
- prosessien sovittaminen ja parantaminen yhteensopivaksi työkalun käytön kanssa
- koulutuksen, valmennuksen ja ohjauksen tarjoaminen uusille käyttäjille
- käyttösuositusten määrittäminen
- toimintatapojen määrittäminen tiedon keräämiseksi välineen todellisesta käytöstä
- työkalun käytön ja hyötyjen seuranta
- välinekohtaisen tuen tarjoaminen testitiimille
- kokemusten kerääminen kaikilta tiimeiltä.

Viitteet

6.2.2 Buwalda, 2001, Fewster, 1999

6.3 Fewster, 1999

7. Viitteet

Standardit

ISTQB Glossary of terms used in Software Testing Version 2.1

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

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

[IEEE 1028] IEEE Std 1028™ (2008) IEEE Standard for Software Reviews and Audits.
Katso luku 3.2.

[IEEE 12207] IEEE 12207/ISO/IEC 12207-2008, Software life cycle processes.
Katso luku 2.1.

[ISO 9126] ISO/IEC 9126-1:2001, Software Engineering – Software Product Quality.
Katso luku 2.3.

Kirjat

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

[Black, 2001] Black, R. (2001) *Managing the Testing Process* (2nd edition), John Wiley & Sons: New York
Katso luvut 1.1, 1.2, 1.4, 1.5, 2.3, 2.4, 5.1, 5.2, 5.3, 5.5, 5.6

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

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

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

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

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

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

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

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

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

8. Liite A – Sertifikaattisisällön tausta

Tämän dokumentin historia

Tämä dokumentti valmisteltiin vuosien 2004 ja 2007 välillä työryhmässä, joka koostui International Software Testing Qualifications Board (ISTQB) -organisaation nimeämistä jäsenistä. Se katselmoitiin ensin valitussa katselmointiryhmässä ja sitten kansainvälisestä ohjelmistotestausyhteisöstä valittujen edustajien kanssa. Dokumentin laatimisessa noudatetut säännöt on esitetty liitteessä C.

Tämä dokumentti on perussisältö kansainväliselle ohjelmistotestauksen perustason sertifikaatille, joka muodostaa ensimmäisen tason ISTQB:n hyväksymässä kansainvälisessä pätevöitymisessä (www.istqb.org).

Perussertifikaatin pätevöitymisen tavoitteet

- Saada tunnustusta testaukselle oleellisena ja ammattimaisena ohjelmistosuunnittelun alueena.
- Tarjota standardirunko testaajien urakehitykselle.
- Luoda ammatillisesti pätevöityneille testaajille tilaisuus tulla työnantajien, asiakkaiden ja kollegoiden tunnustamiksi ja nostaa testaajien profiilia.
- Kannustaa yhdenmukaisia ja hyviä testauskäytäntöjä kaikilla ohjelmistokehityksen tieteenaloilla.
- Tunnistaa testausaiheet, jotka ovat merkityksellisiä ja arvokkaita toimialalle.
- Luoda ohjelmistotoimittajille mahdollisuus palkata sertifioituja testaajia ja siten saada kaupallista hyötyä kilpailijoihin nähden mainostamalla testaajien rekrytointikäytäntöä.
- Antaa testaajille ja testauksesta kiinnostuneille mahdollisuus hankkia kansainvälisesti tunnustettu pätevöityminen aiheessa.

Tavoitteet kansainväliselle pätevöitymiselle (muokattu ISTQB-kokouksesta Sollentunassa marraskuussa 2001)

- Pystyä vertailemaan testaustaitoja eri maissa.
- Mahdollistaa testaajien siirtyminen helpommin maiden rajojen yli.
- Mahdollistaa monikansallisten/kansainvälisten projektien yhteinen ymmärrys testauskysymyksistä.
- Lisätä pätevöityneiden testaajien määrää maailmanlaajuisesti.
- Saada enemmän vaikutusta/arvoa kansainvälisenä aloitteena kuin maakohtaisen lähestymisen pohjalta
- Kehittää yhteinen kansainvälisen testausymmärryksen ja osaamisen ryhmä sertifikaattisisällön ja sanaston kautta, sekä kasvattaa kaikkien osallistujien testausosaamisen tasoa.
- Edistää testausta ammattina useammissa maissa.
- Luoda testaajille mahdollisuus saada tunnustettu pätevöityminen omalla äidinkielellään
- Mahdollistaa osaamisen ja resurssien jakaminen maiden välillä.
- Tuoda useiden maiden osallistumisen myötä testaajille ja tälle pätevöitymiselle kansainvälistä tunnustusta.

Aloitusvaatimukset tälle sertifikaatille

Aloitusvaatimuksena ISTQB Ohjelmistotestauksen Perustason sertifikaattikokeelle on, että kokelailla on mielenkiintoa ohjelmistotestaukseen. Kuitenkin vahvasti suositellaan, että

- kokelailla on vähintään minimitausta joko ohjelmistokehityksestä tai ohjelmistotestauksesta, kuten kuuden kuukauden kokemus joko järjestelmä- tai hyväksymistestaajana tai ohjelmistokehittäjänä
- kokelaat käyvät ISTQB:n standardien mukaan akkreditoidun kurssin (jonkin ISTQB:n tunnustaman kansallisen hallituksen akkreditoima).

Ohjelmistotestauksen perussertifikaatin tausta ja historia

Ohjelmistotestaajien riippumaton sertifiointi alkoi Yhdistyneessä Kuningaskunnassa (UK) British Computer Society:n Information Systems Examination Board (ISEB) -organisaatiossa, kun Software Testing Board perustettiin 1998 (www.bcs.org.uk/iseb). Vuonna 2002, saksalainen ASQF alkoi tukea saksalaisten testaajien pätevöitymissuunnitelmaa (www.asqf.de). Tämä sertifikaattisisältö perustuu sekä ISEB:n että ASQF:n sertifikaattisisältöihin. Se sisältää uudelleenjärjesteltyä, päivitettyä ja uutta aineistoa, ja sen painopiste on suunnattu aiheisiin, joista on eniten käytännön apua testaajille.

Olemassa oleva Ohjelmistotestauksen Perustason sertifikaatti (esim. ISEB:n, ASQF:n tai ISTQB:n tunnustaman kansallisen hallituksen myöntämä), joka on myönnetty ennen tämän Kansainvälisen Sertifikaatin julkaisua, vastaa Kansainvälistä Sertifikaattia. Perustason sertifikaatti ei vanhene eikä sitä tarvitse uusia. Myöntämispäivämäärä löytyy sertifikaatista.

Jokaisessa osallistuvassa maassa paikallisia näkökohtia hallinnoidaan kansallisessa ISTQB:n tunnustamassa Ohjelmistotestauksen hallituksessa. ISTQB määrittää kansallisten hallitusten tehtävät, mutta ne pannaan täytäntöön jokaisessa maassa. Maiden johtokuntien tehtäviin tulevat kuulumaan koulutuksen tarjoajien akkreditointi ja kokeiden järjestäminen.

9. Liite B - Oppimistavoitteet/osaamistaso

Seuraavia oppimistavoitteita sovelletaan tähän sertifikaattisisältöön. Jokainen sertifikaattisisällön aihe kuulustellaan kokeessa sen oppimistavoitteen mukaisesti.

Taso 1: Muistaminen (K1)

Kokelas tunnistaa, muistaa ja pystyy palauttamaan mieleensä termin tai käsitteen.

Avainsanat: Muistaa, palauttaa mieleen, tunnistaa, tietää.

Esimerkki
Tunnistaa "häiriön" määritelmän:

- "palvelun toimimattomuus loppukäyttäjän tai muun sidosryhmän jäsen näkökulmasta" tai
- "ohjelmiston poikkeama odotetusta toimituksesta, palvelusta tai tuloksesta".

Taso 2: Ymmärtäminen (K2)

Kokelas osaa valita syyt tai perustelut aiheeseen liittyville väitteille ja osaa esittää lyhyesti, vertailla, luokitella, ryhmitellä ja antaa esimerkkejä asiasta.

Avainsanat: Vetää yhteen, yleistää, tehdä yhteenveto, luokitella, vertailla, sijoittaa, asettaa vastakkain, valaista esimerkeillä, tulkita, selittää, kuvata, päätellä, tehdä johtopäätös, ryhmitellä, laatia malleja.

Esimerkkejä
Osaa selittää syyt, miksi testit pitäisi suunnitella niin aikaisin kuin mahdollista:

- Jotta löydetään viat silloin, kun niiden poistaminen on halvempaa
- Jotta löydetään tärkeimmät viat ensin.

Osaa selittää yhtäläisyydet ja erot integrointi- ja järjestelmätestauksen välillä:

- Yhtäläisyydet: useamman kuin yhden komponentin testaus, ja voidaan testata ei-toiminnallisia piirteitä.
- Erot: integrointitestaus keskittyy rajapintoihin ja vuorovaikutuksiin, kun taas järjestelmätestaus keskittyy koko järjestelmän piirteisiin, kuten päästä-päähän prosessointiin.

Taso 3: Soveltaminen (K3)

Kokelas osaa valita oikean toimintamallin tai tekniikan soveltamistavan ja käyttää sitä annetussa yhteydessä.

Avainsanat: soveltaa, suorittaa, käyttää, noudattaa proseduuria, soveltaa proseduuria.

Esimerkki

- Tunnistaa raja-arvot kelvollisille ja epäkelvollisille osioille.
- Osaa valita testitapaukset annetusta tilasiirtymäkaaviosta niin, että kaikki siirtymät saadaan katettua.

Taso 4: Analysointi (K4)

Kokelas osaa jakaa proseduuriin tai tekniikkaan liittyvän tiedon pienempiin osiin ymmärtääkseen sen paremmin, ja pystyy erottamaan tosiasiat ja päätelmät. Tyypillinen käyttötapa on esimerkiksi asiakirjan,

ohjelmiston tai projektin tilanteen analysointi ja sopivien toimenpiteiden ehdottaminen ongelman tai tehtävän ratkaisemiseksi.

Avainsanat: analysoida, organisoida, löytää yhdenmukaisuuksia, yhdistää, jäsentää, rakentaa, päätellä johonkin kuuluvaksi, purkaa osiin, erotella, erottaa, määritellä, tarkentaa, valita.

Esimerkki

- Projektiriskien analysointi ja ennaltaehkäisevien ja korjaavien toimenpiteiden ehdottaminen.
- Kuvata, mitkä havaintoraportin osat perustuvat tosiasioihin ja mitkä ovat tulosten perusteella tehtyjä päätelmiä.

Viite

(Oppimistavoitteiden kognitiivisia tasoja varten)

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

10. Liite C – ISTQB:hen sovellettavat säännöt

Perustason sertifikaattisisältö (syllabus)

Tässä mainittuja sääntöjä käytettiin tämän oppimissisällön laatimisessa ja katselmoinnissa. (Jokaisen säännön lopussa on "TUNNUS" säännön lyhennyksenä).

Yleiset säännöt

SG1. Sertifikaattisisällön pitää olla sellaisten henkilöiden ymmärrettävissä ja sisäistettävissä, joilla on 0 - 6:n (tai useamman) kuukauden testauskokemus. (6-MONTH)
SG2. Sertifikaattisisällön pitää olla ennemmin käytännöllinen kuin teoreettinen. (PRACTICAL)
SG3. Sertifikaattisisällön pitää olla aiottujen lukijoiden kannalta selkeä ja yksiselitteinen. (CLEAR)
SG4. Sertifikaattisisällön pitää olla eri maista tulevien ihmisten ymmärrettävissä, ja helposti käännettävissä eri kielille. (TRANSLATABLE)
SG5. Alkuperäisen sertifikaattisisällön pitää olla amerikanenglantia. (AMERICAN-ENGLISH)

Nykyinen sisältö

SC1. Sertifikaattisisällön pitää sisältää uusimmat testauskäsitteet ja sen pitää heijastaa ohjelmistotestauksen nykyisiä parhaita käytäntöjä, joista on yleisesti sovittu. Sertifikaattisisältö katselmoidaan 3-5 vuoden välein. (RECENT)
SC2. Sertifikaattisisällössä pitää minimoida aikariippuvaiset asiat, kuten tämänhetkiset markkinatilanteet, jotta sillä on 3 - 5 vuoden käyttöaika. (SHELF-LIFE).

Oppimistavoitteet

LO1. Oppimistavoitteiden pitää erotella asiat, jotka kokelaan pitää tunnistaa/muistaa (kognitiivinen taso K1), asiat, jotka pitää ymmärtää käsitteellisesti (K2), ne, joita kokelaan pitäisi pystyä hyödyntämään/käyttämään (K3), ja ne, joita kokelaan pitäisi pystyä hyödyntämään dokumentin, ohjelmiston tai projektin tilanteen analysoimiseksi kyseisessä tilanteessa (K4). (KNOWLEDGE-LEVEL).
LO2. Sisällön kuvauksen pitää olla yhdenmukainen oppimistavoitteiden kanssa. (LO-CONSISTENT)
LO3. Oppimistavoitteiden havainnollistamiseksi sertifikaattisisällön mukana pitää toimittaa esimerkkikoekysymyksiä kaikista pääosa-alueista. (LO-EXAM)

Yleinen rakenne

ST1. Sertifikaattisisällön rakenteen pitää olla selkeä ja sallia ristiviittaukset muihin osiin, kokeen kysymyksiin ja muihin asiaankuuluviin dokumentteihin. (CROSS-REF)
ST2. Sertifikaattisisällön osa-alueiden päällekkäisyys pitää minimoida. (OVERLAP)
ST3. Sertifikaattisisällön jokaisella osa-alueella on oltava sama rakenne. (STRUCTURE-CONSISTENT)
ST4. Sertifikaattisisällössä pitää mainita versio, julkaisupäivä ja sivunumero jokaisella sivulla. (VERSION)
ST5. Sertifikaattisisällön pitää sisältää suositukset siitä, kuinka paljon aikaa tulisi käyttää kuhunkin osa-alueeseen (kuvastaa kunkin aiheen suhteellista tärkeyttä). (TIME-SPENT)

Viitteet

SR1. Sertifikaattisisällön asioista kerrotaan lähteet ja viitteet, jotta koulutuksen tarjoajat pystyvät löytämään lisää tietoa aihealueesta. (REFS)
SR2. Jos valmiiksi tunnistettuja ja selkeitä lähteitä ei ole, sertifikaattisisällössä pitää tarjota enemmän yksityiskohtaista tietoa. Esimerkiksi määritelmät löytyvät sanastosta, joten vain termit on lueteltu sertifikaattisisällössä.(NON-REF DETAIL)

Tiedon lähteet

Sertifikaattisisällössä käytetyt termit on määritelty ISTQB Testaussanastossa. Sanasto on saatavissa ISTQB:lta.
Lista suositelluista ohjelmistotestauksen kirjoista julkaistaan myös tämän sertifikaattisisällön kanssa. Pääkirjalista on kohdassa Viitteet.

11. Liite D – Huomioita koulutuksen tarjoajille

Jokainen sertifikaattisisällön pääaiheen otsikossa on mainittu sille varattu aika minuutteina. Tämän tarkoituksena on sekä ohjeistaa jokaisen osa-alueen suhteellista kestoa valtuutetulla kurssilla että kertoa jokaisen osa-alueen opetukseen käytettävä arvioitu vähimmäisaika. Koulutusorganisaatiot voivat käyttää enemmän aikaa kuin osoitettu ja kokelaat voivat käyttää enemmän aikaa lukemiseen ja asiaan perehtymiseen. Kurssin opetussuunnitelman ei tarvitse seurata sertifikaattissisällön järjestystä.

Sertifikaattisisältö sisältää viitteitä vakiintuneisiin standardeihin, joita täytyy käyttää koulutusmateriaalin valmistelussa. Jokaisesta standardista täytyy käyttää sitä versiota, joka mainitaan sen hetkisessä sertifikaattisisällössä. Muita julkaisuja, mallipohjia tai standardeja, joihin ei viitata sertifikaattisisällössä, on mahdollista käyttää ja niihin voi viitata, mutta niitä ei kuulustella.

Sertifikaattisisällön erityisalueet, jotka vaativat käytännön harjoituksia, ovat seuraavat:

4.3 Määrittelypohjaiset eli mustalaatikkotekniikat

Koulutuksen pitää sisältää käytännön töitä (lyhyitä harjoituksia), jotka kattavat neljä tekniikkaa: ekvivalenssiositus, raja-arvoanalyysi, päätöstaulutestaus ja tilasiirtymätestaus. Näihin tekniikoihin liittyvien luentojen ja harjoitusten pitää perustua tekniikoiden yhteydessä mainittuihin viitteisiin.

4.4 Rakenteeseen perustuvat eli lasilaatikkotekniikat

Mukana on oltava käytännön työtä (lyhyitä harjoituksia), joissa arvioidaan, saavuttaako testijoukko 100% lause- ja 100% päätöskattavuuden, samoin kuin tehtäviä, joissa suunnitellaan testitapauksia annetuista kontrollivirroista.

5.6 Havaintojen hallinta

Käytännön työ (lyhyt harjoitus), johon kuuluu havaintoraportin kirjoittaminen ja/tai arviointi.

12. Liite E – Julkaisuseloste Sertifikaattisisältö 2010

1. Oppimistavoitteisiin tehdyt muutokset sisältävät tarkennuksia.
    a. Seuraaviin oppimistavoitteisiin on tehty sanajärjestysmuutoksia (sisältö ja oppimistaso ovat muuttumattomia): LO-1.2.2,LO-1.4.1, LO-2.1.1, LO-2.1.3, LO-4.6.1, LO-6.3.2
    b. K4 on lisätty. Syy: jotkut vaatimukset (LO-4.4.4 ja LO-5.6.2) on jo kirjoitettu K4-muodossa, ja LO-4.6.1:n kysymykset on helpompi kirjoittaa ja tenttiä K4-tasolla.
    c. LO-1.1.5 on muotoiltu uudelleen ja nostettu tasolle K2 koska virheisiin liittyvien termien vertailua voi odottaa.
    d. LO-1.2.3 Debuggauksen ja testauksen eron selittäminen on uusi oppimistavoite. Sisältö oli jo katettu.
    e. LO-3.1.3 Vertailuasiat käsitelty.
    f. LO-3.1.4 poistettu. Osittain päällekkäinen tavoitteen LO-3.1.3 kanssa.
    g. LO-3.2.1 yhdenmukaisuus sisällön kanssa.
    h. LO-3.3.2 nostettu tasolle K2, jotta se on yhdenmukainen LO-3.1.2:n kanssa.
    i. LO-6.1.2 poistettu, koska on osa LO-6.1.3:a, joka on muotoiltu uudelleen väärän K2-avainsanan käytön vuoksi.
2. Yhdenmukaistettu termin Testauksen lähestymistapa käyttö sanaston määritelmän mukaiseksi. Termiä testausstrategia ei vaadita muistiinpalautettavaksi.
3. Luku 1.4 sisältää nyt testauksen lähdedokumentaation ja testitapausten välisen jäljitettävyyden käsitteen.
4. Luku 2.x sisältää nyt testauskohteet ja testauksen lähdedokumentaation.
5. Uudelleentestaus on nyt sanaston päätermi varmistustestauksen sijaan.
6. Käsite aineiston laatu ja aineistotestaus on lisätty moniin paikkoihin sertifikaattisisältöön. Aineiston laatu ja riski luvuissa 2.2, 5.5, 6.1.8
7. Luku 5.2.3 Aloituskriteerit on lisätty uudeksi alakohdaksi. Syy: yhdenmukaisuus lopetuskriteerien kanssa (-> aloituskriteerit lisätty LO-5.2.9:ään)
8. Yhdenmukainen termien testausstrategia ja testauksen lähestymistapa niiden sanaston mukaisen määritelmän mukaisesti.
9. Luku 6.1 lyhennetty, koska työkalujen kuvaukset olivat liian laajoja 45 minuutin luennoksi.
10. IEEE Std 829:2008 on julkaistu. Tämä sertifikaattisisällön versio ei vielä ota uutta versiosta huomioon. Luvussa 5.2 viivataan kokonaistestaussuunnitelmaan. Kokonaistestaussuunnitelman sisältö katetaan ajatuksella, että dokumentti ”Testaussuunnitelma” kattaa suunnittelun eri tasot. Voidaan siis luoda testaussuunnitelma eri testaustasoille yhtä hyvin kuin projektitasoinen testaussuunnitelma, joka kattaa monta testaustasoa. Jälkimmäistä nimitetään kokonaistestaussuunnitelmaksi tässä sertifikaattisisällössä ja ISTQB:n sanastossa.
11. Eettiset säännöt on siirretty jatkotason sertifikaattisisällöstä perustason sertifikaattisisältöön.