# Sertifioitu Testaaja

# Jatkotason sertifikaattisisältö Testauspäällikkö

Versio 2012
Käännösversio 2015

---

International Software Testing Qualifications Board

---

Jatkotaso, Testauspäällikkö alatyöryhmä Rex Black (puheenjohtaja), Judy McKay (varapuheenjohtaja), Graham Bath, Debra Friedenberg, Bernard Homès, Kenji Onishi, Mike Smith, Geoff Thompson, Tsuyoshi Yumoto; 2010-2012.

# Muutoshistoria

## Englanninkielinen versio:

| Version | Date | Remarks |
|---|---|---|
| ISEB v1.1 | 04SEP01 | ISEB Practitioner Syllabus |
| ISTQB 1.2E | SEP03 | ISTQB Advanced Level Syllabus from EOQ-SG |
| V2007 | 12OCT07 | Certified Tester Advanced Level syllabus version 2007 |
| D100626 | 26JUN10 | Incorporation of changes as accepted in 2009, separation of each chapters for the separate modules |
| D101227 | 27DEC10 | Acceptance of changes to format and corrections that have no impact on the meaning of the sentences. |
| D2011 | 31OCT11 | Change to split syllabus, re-worked LOs and text changes to match LOs. Addition of BOs. |
| Alpha 2012 | 09Feb12 | Incorporation of all comments from NBs received from October release. |
| Beta 2012 | 26Mar12 | Incorporation of comments from NBs received on time from Alpha release. |
| Beta 2012 | 07APR12 | Beta version submitted to GA |
| Beta 2012 | 08JUN12 | Copy edited version released to NBs |
| Beta 2012 | 27JUN12 | EWG and Glossary comments incorporated |
| RC 2012 | 15AUG12 | Release candidate version - final NB edits included |
| GA 2012 | 19OCT12 | Final edits and cleanup for GA release |

## Suomenkielinen versio:

| Version | Date | Remarks |
|---|---|---|
| 1.0 | | Ensimmäinen hyväksytty suomenkielinen versio |
| | | |
| | | |

# Sisällysluettelo

# Kiitokset

Tämän dokumentin on tuottanut International Software Testing Qualifications Boardin Jatkotason työryhmän Testauspäällikkö –alatyöryhmä: Rex Black (puheenjohtaja), Judy McKay (varapuheenjohtaja), Graham Bath, Debra Friedenberg, Bernard Homès, Paul Jorgensen, Kenji Onishi, Mike Smith, Geoff Thompson, Erik van Veenendaal, Tsuyoshi Yumoto.

Ydintiimi kiittää katselmointitiimiä ja kansallisia hallituksia näiden ehdotuksista ja työpanoksesta.

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

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

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

Chris van Bael, Graham Bath, Kimmo Hakala, Rob Hendriks, Marcel Kwakernaak, Rik Marselis, Don Mills, Gary Mogyorodi, Thomas Mueller, Ingvar Nordstrom, Katja Piroué, Miele Posthuma, Nathalie Rooseboom de Vries, Geoff Thompson, Jamil Wahbeh, Hans Weiberg.

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

Seuraavat henkilöt osallistuivat suomenkielisen sertifikaattisisällön kääntämiseen ja katselmointeihin: Minna Aalto, Kimmo Hakala, Jani Haukinen, Kari Kakkonen, Juha Pomppu, Timo Salonen, Laura Selonen.

# 0. Johdatus tähän sertifikaattisisältöön

## 0.1 Tämän dokumentin tarkoitus

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

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

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

## 0.2 Yleiskatsaus

Jatkotaso muodostuu kolmesta erillisestä sertifikaattisisällöstä:

- Testauspäällikkö
- Testausasiantuntija
- Tekninen testausasiantuntija

Dokumentti Jatkotason yleiskatsaus sisältää seuraavat tiedot:

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

## 0.3 Kuulusteltavat oppimistavoitteet

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

# 1. Testausprosessi – 420 min.

## Avainsanat

lopetusehdot, testattava tilanne, testauksen kontrollointi, testauksen päättäminen, testauksen yhteenvetoraportti, testaussuunnittelu, testiloki, testin suorittaminen, testin toteutus, testiproseduuri, testiskripti, testisuunnittelu, testitapaus

## Oppimistavoitteet: Testausprosessi

### 1.2 Testauksen suunnittelu, seuranta ja valvonta

TM-1.2.1 (K4) Analysoida järjestelmään liittyvät testaustarpeet, jotta voidaan suunnitella testaustehtävät ja tuotokset, joiden avulla saavutetaan testaukselle asetut tavoitteet

### 1.3 Testien analysointi

TM-1.3.1 (K3) Hyödyntää jäljitettävyyttä, kun tarkistetaan määriteltyjen testattavien tilanteiden täydellisyyttä ja yhdenmukaisuutta testauksen kohteeseen, testausstrategiaan ja testaussuunnitelmaan verrattuna.

TM-1.3.2 (K2) Selittää tekijät, jotka saattavat vaikuttaa yksityiskohtaisuuden tasoon, jolla testattavat tilanteet määritetään, sekä hyödyt ja haitat, jotka liittyvät testattavien tilanteiden yksityiskohtaiseen määrittelyyn.

### 1.4 Testisuunnittelu

TM-1.4.1 (K3) Käyttää jäljitettävyyttä suunniteltujen testitapausten täydellisyyden ja yhdenmukaisuuden tarkistamiseen verrattuna määritettyihin testattaviin tilanteisiin.

### 1.5 Testien toteutus

TM-1.5.1 (K3) Käyttää tietoa riskeistä, priorisoinnista, testiympäristön ja testiaineiston riippuvuuksista sekä rajoitteista, kun laaditaan testien suoritusaikataulua, joka on täydellinen ja yhdenmukainen testauksen tavoitteiden, testistrategian ja testaussuunnitelman kanssa.

### 1.6 Testin suoritus

TM-1.6.1 (K3) Käyttää jäljitettävyyttä, kun seurataan testauksen etenemisen valmistumista ja yhdenmukaisuutta testauksen tavoitteiden, testausstrategian ja testaussuunnitelman kanssa.

### 1.7 Lopetusehtojen arviointi ja raportointi

TM-1.7.1 (K2) Selittää, miksi tarkan ja oikea-aikaisen tiedon keruu testausprosessin aikana on tärkeää tarkan raportoinnin ja päätöskriteerien arvioinnin tukemiseksi.

### 1.8 Testauksen päätöstehtävät

TM-1.8.1 (K2) Vetää yhteen tiedot neljästä päätöstehtävien ryhmästä.

TM-1.8.2 (K3) Toteuttaa projektin jälkipalaveri prosessien arvioimiseksi ja kehitysalueiden löytämiseksi.

## 1.1 Esittely

ISTQB® Perustason sertifikaattisisältö kuvaa perustestausprosessin, johon kuuluvat seuraavat tehtävät:

- Suunnittelu ja valvonta
- Analysointi ja suunnittelu
- Toteutus ja suoritus
- Lopetusehtojen arviointi ja raportointi
- Testauksen päätöstehtävät.

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

Jatkotasolla joitakin näistä tehtävistä käsitellään erillisinä, jotta prosesseista saadaan tarkempaa tietoa ja niitä voidaan optimoida, jotta tehtävät sopivat paremmin ohjelmistokehityksen elinkaareen, sekä tehokkaan testauksen monitoroinnin ja hallinnan helpottamiseksi. Tehtävät on nyt jaoteltu seuraavasti:

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

## 1.2 Testauksen suunnittelu, seuranta ja valvonta

Tämä kappale keskittyy testauksen suunnittelun, seurannan ja hallinnan prosesseihin. Kuten Perustasolla on tuotu esiin, nämä tehtävät ovat testauksen hallinnan rooleja.

### 1.2.1 Testauksen suunnittelu

Testauksen suunnittelu alkaa kaikilla testaustasolla, kun kyseisen tason testausprosessi käynnistetään, ja se jatkuu läpi projektin kyseisen tason päätöstehtävien valmistumiseen asti. Siihen kuuluu testauksen mission ja testausstrategiassa määriteltyjen tavoitteiden saavuttamiseksi tarvittavien tehtävien ja resurssien tunnistaminen. Testaussuunnitteluun kuuluu myös niiden menetelmien määrittäminen, joilla voidaan kerätä ja seurata mittaritietoja, joita käytetään projektin ohjaamiseen, suunnitelmien noudattamisen seuraamiseen sekä tavoitteiden saavuttamisen arviointiin. Hyödyllisten metriikoiden määrittäminen suunnitteluvaiheessa mahdollistaa työkalujen valinnan, koulutuksen aikatauluttamisen sekä dokumentoinnin ohjeistuksen laatimisen.

Testausprojektille valittu strategia (tai strategiat) auttaa määrittämään tehtävät, jotka pitäisi suorittaa suunnitteluvaiheessa. Kun esimerkiksi käytetään riskipohjaista testausstrategiaa (ks. luku 2), riskianalyysiä käytetään ohjaamaan testisuunnitteluprosessia, joka liittyy tunnistettujen tuoteriskien pienentämiseksi tarvittavien toimenpiteiden suunnitteluun sekä riskeihin liittyvien varasuunnitelmien laatimiseen. Mikäli tunnistetaan useita todennäköisiä ja vakavia tietoturvaan liittyviä mahdollisia vikoja, merkittävä osuus työmäärästä pitäisi käyttää tietoturvatestien laatimiseen ja suorittamiseen. Yhtä lailla, jos huomataan, että suunnittelukuvauksista löytyy yleensä vakavia vikoja, testisuunnitteluprosessiin voidaan päätyä lisäämään enemmän suunnittelukuvausten staattista testausta (katselmointeja).

Tietoa riskeistä voidaan käyttää myös, kun päätetään eri testaustehtävien kiireellisyydestä. Esimerkiksi silloin, jos järjestelmän suorituskyky muodostaa suuren riskin, suorituskykytestaus voidaan suorittaa heti, kun integroitu koodi on saatavilla. Samoin, jos käytetään reaktiivista strategiaa, testausohjeen laatimisen ja dynaamisessa testauksessa, kuten tutkivassa testauksessa, käytettävien työkalujen suunnittelu voi olla aiheellista.

Tämän lisäksi testaussuunnitteluvaiheessa Testauspäällikön tulee määrittää selkeästi testauksen lähestymistavat, mukaan luettuna käytettävät testaustasot, jokaisen testaustason tavoitteet ja päämäärät, sekä mitä testaustekniikoita käytetään milläkin testaustasolla. Esimerkiksi tiettyjen ilmailualan järjestelmien riskipohjaisessa testauksessa riskiarviointi määrittää, minkä tason koodikattavuus vaaditaan ja näin ollen mitä testaustekniikoita pitäisi käyttää.

Testauksen pohjamateriaalin (esim. tiettyjen vaatimusten tai riskien), testattavien tilanteiden ja niitä kattavien testien välillä saattaa esiintyä monimutkaisia yhteyksiä. Näiden tuotosten välillä esiintyy usein monen-suhde-moneen -yhteyksiä. Ne on ymmärrettävä, jotta testien suunnittelu, seuranta ja hallinta voidaan toteuttaa tehokkaasti. Myös työkaluihin liittyvät päätökset voivat riippua tuotosten välisten yhteyksien ymmärtämisestä.

Yhteyksiä saattaa esiintyä myös kehitystiimin ja testaustiimin laatimien tuotosten välillä. Voi olla, että tarvitaan esimerkiksi jäljitettävyysmatriiseja, jotta voidaan seurata järjestelmäsuunnittelijoiden laatimien yksityiskohtaisten suunnittelukuvausten, liiketoiminta-asiantuntijoiden laatimien liiketoimintavaatimusten ja testaustiimin määrittämien testauksen tuotosten välisiä yhteyksiä. Mikäli aiotaan suunnitella ja käyttää alemman tason testitapauksia, voidaan suunnitteluvaiheessa määritellä vaatimus, että toteutustiimin yksityiskohtaiset suunnittelukuvaukset on hyväksytettävä ennen kuin testitapausten luominen voi alkaa. Ketterää elinkaarimallia käytettäessä voidaan pitää epämuodollisia tiedonsiirtopalavereita tiedon välittämiseksi tiimiltä toiselle ennen testauksen aloitusta.

Testaussuunnitelmassa voidaan myös luetella ohjelmiston tietyt piirteet, jotka kuuluvat testauksen piiriin (riskianalyysin perusteella, jos sitä on käytetty), ja siinä kuvataan yksiselitteisesti myös piirteet, jotka jäävät testauksen ulkopuolelle. Projektin muodollisuudesta ja siinä noudatettavasta dokumentointitasosta riippuen jokaiseen testauksen piiriin kuuluvaan ominaisuuteen voidaan liittää vastaava testisuunnitelma.

Tässä vaiheessa voidaan myös edellyttää, että Testauspäällikkö työskentelee yhdessä projektiarkkitehtien kanssa alustavien testiympäristömäärittelyiden laatimiseksi, tarvittavien resurssien saatavuuden varmistamiseksi, sen varmistamiseksi, että ympäristön laatimiseen tarvittavat henkilöt ovat sitoutuneet tehtävään, ja jotta hän ymmärtää testiympäristön laatimiseen ja toimittamiseen tarvittavat kustannukset, aikataulun ja tarvittavan työn määrän.

Lopuksi kaikki ulkoiset riippuvuudet ja projektiin liittyvät palvelutasosopimukset (SLA, Service Level Agreement) pitäisi tunnistaa ja tarvittaessa niiden osalta on tehtävä ensimmäiset yhteydenotot vastakkaisiin osapuoliin. Esimerkkejä riippuvuuksista ovat ulkopuolisiin ryhmiin kohdistuvat resurssipyynnöt sekä riippuvuudet muista projekteista (jos kyseessä on ohjelmistoprojekti), ohjelmistotoimittajista tai toteutuskumppaneista, julkaisutiimistä ja tietokannanhoitajista.

### 1.2.2 Testauksen seuranta ja hallinta

Jotta testauspäällikkö pystyy tuottavasti valvomaan testauksen etenemistä, on suunniteltava testausaikataulu ja menetelmät, joilla voidaan seurata testauksen tuotoksia ja resursseja suunnitelmaa vastaan. Näiden menetelmien pitäisi sisältää yksityiskohtaiset mittarit ja tavoitteet, joita tarvitaan testauksen tuotoksien ja tehtävien tilan vertaamiseksi testaussuunnitelmaan ja strategisiin tavoitteisiin.

Pienissä ja yksinkertaisemmissa projekteissa testauksen tuotoksien ja tehtävien vertaaminen suunnitelmaan ja strategisiin tavoitteisiin voi olla suhteellisen helppoa, mutta yleensä tämän saavuttamiseksi on määritettävä yksityiskohtaisemmat tavoitteet. Näihin voivat sisältyä testauksen tavoitteiden ja testauksen pohjamateriaalin kattavuuden saavuttamiseen liittyvät mittarit ja tavoitteet.

Erityisen tärkeää on yhdistää testauksen tuotoksien ja tehtävien tila testauksen pohjamateriaaliin tavalla, joka on ymmärrettävä ja merkittävä projektin ja liiketoiminnan sidosryhmien kannalta. Tähän voidaan päästä määrittelemällä tavoitteet ja seuraamalla edistymistä testattavien tilanteiden ja niiden muodostamien ryhmien perusteella, jos muut testauksen tuotokset yhdistetään testauksen pohjamateriaaliin testattavien tilanteiden avulla. Oikein suunniteltuna jäljitettävyys antaa mahdollisuuden myös jäljitettävyyden tilan raportointiin ja tekee kehityksen tuotoksien, testauksen pohjamateriaalin ja testauksen tuotoksien välisistä monimutkaisista suhteista läpinäkyvämpiä ja ymmärrettävämpiä.

Joskus monimutkaiset mittaritiedot ja tavoitteet, joita sidosryhmät vaativat seurattaviksi, eivät liity suoraan järjestelmän toiminnallisuuteen tai määrityksiin, erityisesti jos muodollista dokumentaatiota on vain vähän tai ei ollenkaan. Esimerkiksi liiketoiminnan sidosryhmät voivat olla kiinnostuneempia saamaan aikaan toiminnallisen liiketoimintakierroksen kattavuuden vaikka määrittely on kirjoitettu järjestelmän toiminnallisuuden näkökulmasta. Liiketoiminnan sidosryhmien mukanaolo projektin aikaisessa vaiheessa voi auttaa määrittämään nämä mittarit ja tavoitteet, joita ei pelkästään käytetä projektin etenemisen paremmassa hallinnassa, vaan ne voivat myös auttaa ohjaamaan testaustehtäviä ja vaikuttamaan niihin läpi projektin. Sidosryhmien mittarit ja tavoitteet voivat johtaa esimerkiksi siihen, että testisuunnittelun ja testien toteutustyön tuotokset ja/tai testien suoritusaikataulu rakennetaan niin, että ne helpottavat testauksen etenemisen tarkkaa seurantaa näitä mittareita vasten. Nämä tavoitteet auttavat myös tuottamaan jäljitettävyyden tietyllä testaustasolla ja niitä voidaan käyttää myös tuottamaan jäljitettävyyttä läpi eri testaustasojen.

Testauksen hallinta on jatkuva tehtävä. Siihen kuuluu todellisen edistymisen seuranta suunnitelmaa vasten ja tarpeen vaatiessa korjaavien toimenpiteiden käynnistäminen . Testauksen hallinta ohjaa testausta niin, että testauksen missio, strategiat ja tavoitteet saavutetaan, ja siihen kuuluu suunnittelutehtävien uudelleen läpikäyminen tarpeen mukaan. Se, miten hallinnasta saatuihin tietoihin reagoidaan, riippuu yksityiskohtaisista suunnittelutiedoista.

Testaussuunnitteluun liittyvien dokumenttien sisältö ja testauksen hallintaan liittyvät tehtävät käsitellään luvussa 2.

## 1.3 Testien analysointi

Sen sijaan, että testien analysointia ja suunnittelua käsiteltäisiin yhdessä, kuten Perustason sertifikaattisisällössä on kuvattu, Jatkotason sertifikaattisisältö käsittelee niitä erillisinä tehtävinä, vaikka tiedostaakin, että ne voidaan suorittaa rinnakkaisina, integroituina tai iteratiivisina tehtävinä testisuunnittelun tuotoksien laatimisen helpottamiseksi.

Testianalyysi on tehtävä, joka määrittää testattavien tilanteiden muodossa sen, “mitä” testataan. Testattavat tilanteet voidaan tunnistaa analysoimalla testauksen pohjamateriaalia, testauksen tavoitteita ja tuoteriskejä. Niitä voidaan pitää onnistumisen yksityiskohtaisina mittareina ja tavoitteina (esim. osana lopetuskriteereitä) ja ne pitäisi pystyä jäljittämään takaisinpäin testauksen pohjamateriaaliin ja määriteltyihin strategisiin tavoitteisiin, mukaan luettuna testauksen tavoitteet ja muut mittarit, joita projekti tai sidosryhmä on asettanut onnistumiselle. Testattavia tilanteita pitäisi myös pystyä jäljittämään eteenpäin testisuunnitelmiin ja muihin testauksen tuotoksiin sitä mukaa, kun niitä luodaan.

Tietyllä testaustasolla testianalysointi voidaan tehdä heti, kun kyseisen testaustason pohjamateriaali on laadittu ja vakaa. Testattavien tilanteiden tunnistamiseen voidaan käyttää muodollisia testaustekniikoita ja muita yleisiä analyyttisiä tekniikoita (esim. analyyttiset riskipohjaiset lähestymistavat ja analyyttiset vaatimuspohjaiset lähestymistavat). Testattavissa tilanteissa voidaan määrittää arvoja tai muuttujia, riippuen testaustasosta, analyysin suoritushetkellä käytettävissä olevasta tiedosta ja valitusta yksityiskohtaisuuden (eli dokumentaation tarkkuuden) tasosta.

Kun tehdään päätöstä testattavien tilanteiden määrittelyssä käytettävästä yksityiskohtaisuuden tasosta, on mietittävä monia asioita, mm.

- Testaustaso
- Testauksen pohjamateriaalin yksityiskohtaisuus ja laatu
- Järjestelmän/ohjelmiston monimutkaisuus
- Projekti- ja tuoteriskit
- Suhde testauksen pohjamateriaalin ja sen välillä, mitä testataan ja miten.
- Käytettävä ohjelmistokehitysmalli
- Käytettävä testauksenhallintaväline
- Testisuunnitelmien ja muiden testauksen tuotoksien määrittelyssä ja dokumentoinnissa käytettävä taso
- Testausasiantuntijoiden tiedot ja taidot

- Testausprosessin ja itse organisaation kypsyystaso (huomaa, että suurempi kypsyys voi vaatia korkeampaa yksityiskohtaisuuden tasoa tai mahdollistaa matalamman yksityiskohtaisuuden tason)
- Projektin muiden sidosryhmien edustajien käytettävyys konsultoinnissa.

Testattavien tilanteiden määrittäminen yksityiskohtaisesti johtaa yleensä suurempaan testattavien tilanteiden määrään. Voi olla, että esimerkiksi verkkokauppasovellukseen liittyy yksi yleinen testattava tilanne, “Testaa uloskirjautuminen”. Jos testattavia tilanteita dokumentoidaan yksityiskohtaisesti, tämä voidaan kuitenkin jakaa useisiin testattaviin tilanteisiin, jolloin jokaista käytettävistä maksutapaa kohti on yksi tilanne, jokaista kohdemaata kohti yksi tilanne, ja niin edelleen.

Yksityiskohtaiseen testattavien tilanteiden kuvaamiseen liittyy mm. seuraavia etuja:

- Lisää joustavuutta suhteessa muihin testauksen tuotoksiin (esim. testitapaukset), testauksen pohjamateriaalin sekä testauksen tavoitteisiin ja tarjoaa näin Testauspäällikölle paremmat mahdollisuudet testauksen seurantaan ja hallintaan.
- Edesauttaa vikojen ehkäisemistä, kuten on todettu jo Perustasolla, kun toteutetaan projektin aikaisessa vaiheessa testauksen ylemmillä tasoilla, niin pian kun testauksen pohjamateriaali on määritelty ja mahdollisesti jo ennen kuin järjestelmän arkkitehtuuri ja yksityiskohtaiset suunnittelukuvaukset ovat saatavilla.
- Esittää testauksen tuotokset sidosryhmien edustajille käyttämällä käsitteitä, jotka ovat heille ymmärrettäviä (usein testitapaukset ja muut testauksen tuotokset eivät merkitse mitään liiketoiminnan edustajille ja yksinkertaiset mittaritiedot, kuten suoritettujen testitapausten määrä, eivät tarkoita mitään sidosryhmien edustajien kattavuusvaatimuksien kannalta).
- Auttaa vaikuttamaan ei pelkästään muihin testaustehtäviin vaan myös muihin toteutuksen tehtäviin ja auttaa ohjaamaan niitä.
- Mahdollistaa testien suunnittelun, toteutuksen ja suorituksen sekä tuloksena syntyvien tuotosten optimoinnin yksityiskohtaisten mittaritietojen ja tavoitteiden paremman kattavuuden perusteella.
- Luo perustan selkeämmälle jäljitettävyydelle testaustasolla vaakasuunnassa.

Yksityiskohtaiseen testattavien tilanteiden kuvaamiseen liittyy mm. seuraavia haittoja:

- Saattaa olla aikaavievää
- Ylläpidettävyys voi olla vaikeaa muuttuvassa ympäristössä
- Muodollisuuden taso täytyy määrittää ja koko tiimin pitää noudattaa sitä.

Yksityiskohtainen testattavien tilanteiden määrittely voi olla erityisen tehokasta seuraavissa tilanteissa:

- Valitun ohjelmistokehityksen elinkaarimallin, kustannuksiin ja/tai aikaan liittyvien tai muiden rajoitteiden vuoksi käytetään kevyttä testisuunnitteludokumentointia, kuten tarkistuslistoja.
- Testauksen pohjamateriaalina on käytettävissä vain vähän muodollisia vaatimuksia tai muita kehityksen tuotoksia, tai niitä ei ole lainkaan.
- Kyseessä on laaja, monimutkainen tai korkean riskin projekti, joka vaatii seurantaa ja valvontaa tasolla, jota ei voida saavuttaa pelkästään kehityksen tuotoksiin liitettyjen testitapausten pohjalta.

Testattavat tilanteet voidaan määritellä karkeammalla tasolla, kun testauksen perusta voidaan liittää helposti ja suoraan testisuunnittelun tuotoksiin. Näin on todennäköisemmin seuraavissa tilanteissa:

- Yksikkötason testauksessa
- Yksinkertaisemmissa projekteissa, joissa on yksinkertaiset hierarkkiset suhteet sen välillä, mitä pitää testata ja kuinka se pitää testata.
- Hyväksymistestauksessa, jossa käyttötapauksia voidaan käyttää apuna testien määrittelyssä.

## 1.4 Testisuunnittelu

Testisuunnittelu on tehtävä, jossa määritellään “miten” jokin asia testataan. Siihen kuuluu testitapausten luominen tunnistettujen testattavien tilanteiden tai testauksen pohjamateriaalin perusteella tarkentamalla niitä askelittain testausstrategiassa ja/tai testaussuunnitelmassa lueteltuja testaustekniikoita käyttämällä.

Testitapaukset voivat liittyä suoraan (tai epäsuorasti toisten testattavien tilanteiden kautta) testauksen pohjamateriaaliin ja asetettuihin tavoitteisiin, riippuen testauksen seurantaan, hallintaan ja jäljitettävyyteen käytettävistä lähestymistavoista. Näihin tavoitteisiin voivat kuulua strategiset tavoitteet, testauksen tavoitteet ja muut projektin tai sidosryhmien asettamat onnistumiskriteerit.

Testisuunnittelu voidaan tehdä kullakin testaustasolla, kun sen testattavat tilanteet on tunnistettu ja saatavilla on riittävästi tietoa joko korkean tai matalan tason testitapausten luomiseksi sovittua testisuunnittelun lähestymistapaa noudattamalla. Ylemmillä testaustasoilla on todennäköisempää, että testisuunnittelu on erillinen tehtävä, joka seuraa aiemmin tehtyä testianalyysia. Alemmilla testaustasoilla on todennäköistä, että testien analysointi ja suunnittelu tehdään yhteensovitettuna tehtävänä.

On myös todennäköistä, että jotkut normaalisti testien toteutusvaiheessa suoritettavat tehtävät integroidaan testien suunnitteluprosessiin, kun käytetään iteratiivista lähestymistapaa suoritettavaksi tarvittavien testien laatimisessa; tällainen tehtävä on esimerkiksi testiaineiston laatiminen. Tämä lähestymistapa voi itse asiassa optimoida testattavien tilanteiden kattamisen, kun prosessin aikana luodaan joko matalan tason tai korkean tason testitapauksia.

## 1.5 Testien toteutus

Testien toteutus on tehtävä, jonka aikana Testausasiantuntijat järjestävät ja priorisoivat testit. Silloin, kun käytetään muodollista dokumentointia, testien toteutus on tehtävä, jolloin testisuunnitelmien pohjalta laaditaan konkreettiset testitapaukset, testiproseduurit ja testiaineisto. Jotkut organisaatiot, jotka noudattavat IEEE 829 [IEEE829] standardia, määrittelevät testien syötteet ja niihin liittyvät odotetut tulokset testitapaussuunnitelmaan ja testiaskeleet testiproseduurin kuvaukseen. On kuitenkin yleisempää, että jokaisen testin syötteet, odotetut tulokset ja testiaskeleet dokumentoidaan yhdessä. Testien toteutukseen kuuluu myös tallennettavan testiaineiston luominen (esim. tiedostot tai tietokannan taulut).

Testien toteutusvaiheessa tehdään myös viimeiset tarkistukset sen varmistamiseksi, että testaustiimi on valmis aloittamaan testien suorituksen. Tarkistuksiin voi kuulua myös sen varmistaminen, että tarvittava testiympäristö, testiaineisto ja ohjelmakoodi on toimitettu (mahdollisesti suorittamalla ympäristön ja/tai koodin hyväksymistestejä) ja että kaikki testitapaukset on kirjoitettu, katselmoitu ja että ne ovat valmiita suoritettavaksi. Se voi sisältää myös kyseisen testaustason eksplisiittisten ja implisiittisten aloituskriteerien täyttymisen tarkistuksen (ks. kappale 1.7). Testien toteutus voi myös sisältää yksityiskohtaisen testiympäristön ja testiaineiston kuvauksen laatimisen.

Testauksen tuotosten (esim. testitapausten ja testattavien tilanteiden) yksityiskohtaisuus voi vaikuttaa testien toteutukseen liittyvän työn yksityiskohtaisuuteen ja monimutkaisuuteen. Erityisesti, kun testit on tarkoitus arkistoida pitkään uudelleenkäytettäviksi regressiotesteiksi, testit saattavat sisältää yksityiskohtaisen kuvauksen testin suorittamiseksi tarvittavista askelista, jotta varmistetaan luotettava, yhdenmukainen testin suoritus riippumatta siitä, kuka testin suorittaa. Jos on noudatettava erilaisia säädöksiä tai sääntöjä, testien pitäisi tuottaa todisteet siitä, että soveltuvia standardeja on noudatettu (ks. kappale 2.9).

Manuaalisten ja automatisoitujen testien suoritusjärjestys pitäisi sisällyttää testien toteutuksen aikana testien suoritusaikatauluun. Testauspäälliköiden on tarkistettava huolellisesti, onko olemassa rajoitteita, riskit ja prioriteetit mukaan luettuna, jotka saattavat edellyttää testien ajamista tietyssä järjestyksessä tai tietyllä laitteistolla. Testiympäristön tai testiaineiston riippuvuuksien on oltava tiedossa ja ne pitää tarkistaa.

Aikaiseen testien toteutukseen saattaa liittyä joitakin haittoja. Esimerkiksi ketterässä ohjelmistokehityksessä koodi saattaa muuttua merkittävästi iteraatiokierroksesta toiseen, minkä seurauksena suuri osa toteutustyöstä voi tulla tarpeettomaksi. Jopa ilman niinkin muutosaltista elinkaarimallia kuin Ketterä ohjelmistokehitys, mikä tahansa iteratiivinen tai inkrementaalinen elinkaarimalli voi johtaa huomattaviin muutoksiin iteraatioiden välillä, mikä tekee skriptatuista testeistä epäluotettavia tai aiheuttaa niihin kohdistuvia merkittäviä ylläpitotarpeita. Sama pätee huonosti johdettuun peräkkäiselinkaarimalliin, jossa

vaatimukset muuttuvat usein, jopa myöhään projektissa. Ennen kuin aloitetaan merkittävä testien toteutusponnistus, on viisasta ymmärtää ohjelmistokehityksen elinkaarimalli ja testaukseen saatavilla olevien ohjelmiston ominaisuuksien ennakoitavuus.

Aikaiseen testien toteutukseen saattaa liittyä joitakin etuja. Esimerkiksi testauksen pohjamateriaalin kanssa yhdenmukaiset konkreettiset testit tarjoavat läpikäytyjä esimerkkejä siitä, kuinka ohjelmiston pitäisi käyttäytyä. Liiketoiminnan asiantuntijat kokevat todennäköisesti konkreettisten testien todentamisen abstraktien liiketoimintasääntöjen todentamista helpommaksi, ja siksi he saattavat myös tunnistaa lisää heikkouksia ohjelmiston määrittelyissä. Tällaiset todennetut testit voivat tarjota ohjelmistosuunnittelijoille ja toteuttajille valaisevia esimerkkejä ohjelmalta vaaditusta käyttäytymisestä.

## 1.6 Testin suoritus

Testien suoritus alkaa, kun testauksen kohde on toimitettu ja suorituksen aloituskriteerit on täytetty. Testit pitäisi suunnitella tai ainakin määritellä ennen testien suoritusta. Työkalujen pitäisi olla valmiina, erityisesti testauksen hallintaa, havaintojen seurantaa ja testien suorituksen automatisointia (mikäli sitä käytetään) varten. Testitulosten seurannan, mukaan luettuna mittareiden seuranta, pitäisi olla käynnissä ja kaikkien tiimin jäsenten tulisi ymmärtää, mitä seurattavat tiedot tarkoittavat. Tulosten kirjaamista ja vikojen raportointia varten pitäisi olla olemassa standardit, jotka on julkistettu. Varmistamalla, että nämä asiat ovat kunnossa ennen testien suoritusta, suoritus voi edetä tuottavasti.

Testit pitäisi suorittaa testitapausten mukaisesti, vaikka Testauspäällikön kannattaa harkita pienen jouston sallimista, jotta testaaja voi tutkia muita kiinnostavia testiskenaarioita ja käyttäytymistä, jota huomataan testauksen aikana. Kun noudatetaan testausstrategiaa, joka on ainakin osittain reaktiivinen, pitäisi jonkin verran aikaa varata testaussessioille, joissa käytetään kokemuspohjaisia ja vikaperusteisia tekniikoita. Minkä tahansa tällaisen ei-skriptatun testauksen yhteydessä havaitun häiriön osalta tulee tietysti kuvata poikkeamat kirjatusta testitapauksesta, jotta häiriö voidaan toistaa. Automatisoidut testit noudattavat niille määrättyjä ohjeita ilman poikkeuksia.

Testauksen suorituksen aikana Testauspäällikön päätehtävä on seurata edistymistä verrattuna testaussuunnitelmaan ja mikäli tarpeen, käynnistää ja suorittaa korjaavat toimenpiteet testauksen ohjaamiseksi mission, tavoitteiden ja strategian kannalta onnistuneeseen päätökseen. Tähän Testauspäällikkö voi käyttää jäljitettävyyttä testituloksista takaisin testattaviin tilanteisiin, testauksen pohjamateriaalin ja viime kädessä testauksen tavoitteisiin, sekä myös testauksen tavoitteista testauksen tuloksiin päin. Tämä prosessi on kuvattu yksityiskohtaisesti kappaleessa 2.6.

## 1.7 Lopetusehtojen arviointi ja raportointi

Testauksen edistymisen seurannan ja hallinnan dokumentointia ja raportointia käsitellään yksityiskohtaisesti kappaleessa 2.6.

Testausprosessin kannalta on tärkeää varmistaa, että on olemassa tehokkaat prosessit, joiden avulla saadaan tarpeellista tietoa testauksen päätöskriteerien arviointia ja raportointia varten.

Kerättävien tietojen vaatimusten ja keruumenetelmien määrittäminen ovat osa testauksen suunnittelua, seurantaa ja hallintaa. Testien analysoinnin, suunnittelun, toteutuksen ja suorituksen aikana Testauspäällikön pitäisi varmistaa että kyseisistä vaiheista vastuussa olevat testaustiimin jäsenet tuottavat vaaditut tiedot tarkasti ja oikea-aikaisesti tehokkaan arvioinnin ja raportoinnin helpottamiseksi.

Raportoinnin tiheys ja yksityiskohtaisuuden taso riippuvat projektista ja organisaatiosta. Tästä pitäisi sopia testauksen suunnitteluvaiheessa yhdessä keskeisten projektin sidosryhmien kanssa.

## 1.8 Testauksen päätöstehtävät

Sen jälkeen kun testaus on todettu päättyneeksi, avaintuotokset pitäisi tallentaa ja joko siirtää eteenpäin asiaankuuluvalle henkilölle tai arkistoida. Kokonaisuutena näitä tehtäviä kutsutaan testauksen päätöstehtäviksi. Testauksen päätöstehtävät jakautuvat neljään pääryhmään:

1. Testauksen valmistumisen tarkastaminen – sen varmistaminen, että kaikki testaustyö on todella saatu tehtyä loppuun. Esimerkiksi kaikkien suunniteltujen testien pitäisi olla joko suoritettu tai tarkoituksella ohitettu, ja kaikkien tunnettujen vikojen pitää olla joko korjattu ja uudelleentestattu, siirretty tulevaan julkaisuun tai hyväksytty pysyvänä rajoitteena.
2. Testausmateriaalin luovutus – hyödylliset tuotokset luovutetaan niitä tarvitseville. Esimerkiksi tunnetuista lykätyistä tai hyväksytyistä vioista pitää viestittää niille, jotka käyttävät järjestelmää ja tukevat sen käyttöä. Testit ja testiympäristöt pitää luovuttaa niille, jotka ovat vastuussa ylläpitotestauksesta. Regressiotestijoukot (joko automatisoidut tai manuaaliset) pitää dokumentoida ja luovuttaa ylläpitotiimille.
3. Kokemuksista oppiminen – pidetään tai osallistutaan jälkipalaveriin, jossa voidaan dokumentoida tärkeät opitut asiat (sekä testausprosessin että koko ohjelmistokehityksen elinkaaren näkökulmasta). Näissä kokouksissa laaditaan suunnitelmat sen varmistamiseksi, että hyvät käytännöt voidaan toistaa ja että huonoja käytäntöjä ei joko toisteta tai ongelmiin varaudutaan projektisuunnitelmissa, mikäli niitä ei voida ratkaista. Huomioon otettaviin asioihin kuuluvat mm. seuraavat:
    a. Oliko käyttäjien edustus laaturiskien analysointipalavereissa liittävän laaja-alainen? Esimerkiksi myöhään löydettyjen ennakoimattomien vikaryppäiden vuoksi tiimi on voinut todeta, että tulevissa projekteissa laaturiskien analysointipalavereihin pitäisi osallistua laaja-alaisempi joukko käyttäjien edustajia.
    b. Olivatko arviot paikkansapitäviä? Arvioita on voitu esimerkiksi tulkita merkittävästi väärin ja siksi tulevissa arvioinneissa on otettava huomioon tämä ja tulkintavirheiden taustalla olevat syyt, esim. oliko testaus tuottamatonta vai oliko arvio todellisuudessa pienempi kuin sen olisi pitänyt olla.
    c. Mitkä ovat vikojen trendit ja syy-seuraus-analyysissä havaitut tulokset? Arvioi esimerkiksi, vaikuttivatko myöhäiset muutospyynnöt analyysin ja kehityksen laatuun, etsi trendejä, jotka antavat viitteitä huonoista käytännöistä, esim. jätettiinkö ajan säästämiseksi väliin testaustaso, jolla viat olisi voitu löytää aikaisemmin ja kustannustehokkaammin. Tarkista, onko vikatrendeillä yhteyksiä esim. uusiin teknologioihin, henkilöstömuutoksiin tai osaamiseen puutteeseen.
    d. Onko havaittavissa mahdollisia prosessin kehittämistilaisuuksia?
    e. esiin suunnitelmiin nähden odottamattomia poikkeamia, joihin pitäisi varautua tulevissa suunnitelmissa?
4. Tulosten, lokien, raporttien sekä muiden dokumenttien ja tuotosten arkistoiminen kokoonpanonhallintajärjestelmään. Esimerkiksi testaussuunnitelma ja projektisuunnitelma sekä tiedot, missä järjestelmässä ja versiossa niitä käytettiin, pitäisi tallentaa suunnitelma-arkistoon.

Nämä tehtävät ovat tärkeitä, ne jäävät usein tekemättä, ja niiden pitäisi olla eksplisiittisesti osa testaussuunnitelmaa.

On tyypillistä, että yksi tai useampia näistä tehtävistä jätetään pois yleensä siksi, että projektitiimin jäsenet siirretään uusiin tehtäviin tai tiimi lakkautetaan liian aikaisin, seuraavien projektien resurssi- tai aikataulupaineiden vuoksi tai tiimin loppuun palamisen vuoksi. Sopimuspohjaisissa projekteissa, kuten esimerkiksi tilaustyönä tehtävässä kehitystyössä, vaaditut tehtävät pitäisi määritellä sopimuksessa.

# 2. Testauksenhallinta– 750 min.

## Avainsanat

kokonaistestaussuunnitelma, laaturiski, projektiriski, riski, riskianalyysi, riskiarviointi, riskien tunnistaminen, riskienhallinta, riskien lieventäminen, riskipohjainen testaus, riskitaso, tasokohtainen testaussuunnitelma, testattavat tilanteet, testauksen kontrollointi, testauksen lähestymistapa, testauksen seuranta, testauksen työmäärän arviointi, testauksenhallinta, testausjohtaja, testauspäällikkö, testitaso, tuoteriski

## Oppimistavoitteet: Testauksenhallinta

### 2.2 Testauksenhallinnan taustaa

TM-2.2.1 (K4) Analysoida ohjelmistoprojektin tai ohjelman sidosryhmiä, olosuhteita ja tarpeita, mukaan luettuna ohjelmistokehityksen elinkaarimalli, ja tunnistaa optimaaliset testaustehtävät.
TM-2.2.2 (K2) Ymmärtää, kuinka ohjelmistokehityksen elinkaaren tehtävät ja tuotokset vaikuttavat testaukseen, ja kuinka testaus vaikuttaa ohjelmistokehityksen elinkaaren tehtäviin ja tuotoksiin.
TM-2.2.3 (K2) Selittää tavat, joilla voidaan hallita kokemuspohjaiseen testaukseen ja ei-toiminnalliseen testaukseen liittyviä testauksen hallintaan kuuluvia asioita.

### 2.3 Riskipohjainen testaus ja muita lähestymistapoja testien priorisointiin ja työmäärien kohdentamiseen

TM-2.3.1 (K2) Selittää, millä eri tavoin riskipohjainen testaus vastaa riskeihin
TM-2.3.2 (K2) Selittää esimerkkien avulla tuoteriskianalyysissä käytettäviä eri tekniikoita.
TM-2.3.3 (K4) Analysoida, tunnistaa ja arvioida tuotelaaturiskejä, ja vetää yhteen riskit ja niiden arvioidut riskitasot projektin avainsidosryhmien näkökulmista.
TM-2.3.4 (K2) Kuvata, kuinka tunnistettuja tuotelaaturiskejä voidaan pienentää ja hallita niiden arvioidun riskitason mukaisesti läpi ohjelmiston elinkaaren ja testausprosessin.
TM-2.3.5 (K2) Antaa esimerkkejä testien valintaan ja priorisointiin sekä työmäärien kohdentamiseen liittyvistä vaihtoehdoista.

### 2.4 Testausdokumentaatio ja muut tuotokset

TM-2.4.1 (K4) Analysoida näytteitä testauspolitiikoista ja testistrategioista, sekä luoda kokonaistestaussuunnitelmia, tasokohtaisia testaussuunnitelmia ja muita testaustuotoksia, jotka ovat täydellisiä ja yhdenmukaisia edellä mainittujen dokumenttien kanssa.
TM-2.4.2 (K4) Analysoida määrätyn projektin projektiriskejä ja valita sopivat riskienhallintavaihtoehdot (lieventäminen, varautuminen, siirtäminen ja/tai hyväksyminen).
TM-2.4.3 (K2) Kuvata esimerkkien avulla, kuinka testauksen lähestymistavat (testausstrategiat) vaikuttavat testaustehtäviin.
TM-2.4.4 (K3) Määritellä mahdollisuuksien mukaan eri standardointiorganisaatioilta saatavilla olevia mallipohjia hyödyntämällä organisaation, elinkaaren ja projektin tarpeisiin sopivat testauksen tuotosten dokumentaation normit ja mallipohjat.

### 2.5 Testauksen työmääräarviointi

TM-2.5.1 (K3) Laatia osoitetun projektin kaikille testausprosessiin liittyville tehtäville työmääräarviot kaikkia soveltuvia arviointitekniikoita käyttämällä.
TM-2.5.2 (K2) Ymmärtää ja kuvata esimerkkien avulla tekijät, jotka saattavat vaikuttaa testauksen työmääräarvioihin.

### 2.6 Testauksen mittaritietojen määrittäminen ja käyttö

TM-2.6.1 (K2) Kuvata ja vertailla tyypillisiä testaukseen liittyviä metriikoita.
TM-2.6.2 (K2) Vertailla testauksen edistymisen seurannan eri ulottuvuuksia.
TM-2.6.3 (K4) Analysoida ja raportoida testitulokset jäljellä olevien riskien, vikojen, testien suorituksen ja testien kattavuuden tilan sekä luottamuksen perusteella ja tuottaa näin näkemyksiä ja suosituksia, joiden avulla projektin sidosryhmien edustajat voivat tehdä julkaisupäätöksiä.

## 2.7 Testauksen liiketoiminnallinen arvo

TM-2.7.1 (K2) Antaa esimerkkejä jokaisesta neljästä laatukustannuksia määrittävästä kategoriasta.

TM-2.7.2 (K3) Arvioida testauksen arvo laatukustannusten perusteella, ottaen huomioon myös määrälliset ja laadulliset seikat, ja välittää arvio testauksen sidosryhmille.

## 2.8 Hajautettu, ulkoistettu ja paikallisesti ulkoistettu testaus

TM-2.8.1 (K2) Ymmärtää tekijät, joita tarvitaan, kun halutaan onnistuneesti käyttää hajautettua, ulkoistettua ja sisäisesti ulkoistettua testauksen henkilöstöstrategiaa.

## 2.9 Teollisuusstandardien soveltamisen hallinta

TM-2.9.1 (K2) Vetää yhteen ohjelmistotestaukseen liittyvien standardien lähteet ja käyttötavat.

## 2.1 Esittely

Jatkotasolla testauksen ammattilainen on alkanut erikoistua urallaan. Tämä luku keskittyy alueisiin, joilta testauksen ammattilaisilta oletetaan olevan osaamista, kun he siirtyvät Testauspäällikön ja Testausjohtajan tehtäviin. Tässä sertifikaattisisällössä käytämme näistä kaikista ammattilaisista yhteisesti nimitystä Testauspäällikkö, vaikka on selvää, että eri organisaatioissa näille nimikkeille on erilaisia määritelmiä ja tehtävissä toimivilla henkilöillä on eritasoisia vastuita.

## 2.2 Testauksenhallinnan taustaa

Esimiesten keskeinen vastuu on varmistaa resurssit (ihmiset, ohjelmisto, laitteisto, ympäristö jne.) lisäarvoa tuovien prosessien suorittamiseksi ja hyödyntää niitä. Ohjelmisto- ja IT-johtajien kohdalla prosessit ovat usein osa projektia tai ohjelmaa, joka tähtää sisäiseen tai ulkoiseen käyttöön tarkoitetun ohjelmiston tai järjestelmän toimittamiseen. Testauspäällikön kohdalla prosessit liittyvät testaukseen, erityisesti testauksen perusprosessiin, joka on kuvattu Perustason sertifikaattisisällössä ja tämän sertifikaattisisällön luvussa 1. Koska testausprosessit tuovat lisäarvoa vain tukemalla projektin tai ohjelman yleistä onnistumista (tai estämällä vakavammat epäonnistumiset), Testauspäällikön täytyy suunnitella ja hallita testausprosesseja sen mukaisesti. Toisin sanoen, Testauspäällikön täytyy järjestää testausprosessit ja niihin liittyvät tehtävät ja tuotokset tilanteen mukaan muiden sidosryhmien, niiden tehtävien (esim. ohjelmistokehityksen elinkaari, jossa testaus tapahtuu) ja tuotosten (esim. vaatimusmäärittelyt) edellyttämällä tavalla.

### 2.2.1 Testauksen sidosryhmien ymmärtäminen

Henkilöt kuuluvat testauksen sidosryhmiin silloin, kun testaustehtävillä, testauksen tuotoksilla tai lopullisen järjestelmän tai tuotoksen laadulla on heille merkitystä. Sidosryhmien kiinnostus voi kohdistua suoraan tai epäsuoraan testaustehtäviin osallistumiseen, suoraan tai epäsuoraan testauksen tuotoksien vastaanottoon, tai projektin tai ohjelman aikaansaamien tuotosten suoraan tai epäsuoraan laatuvaikutukseen.

Vaikka testauksen sidosryhmät vaihtelevatkin projektin, tuotteen, organisaation ja muiden tekijöiden mukaan, seuraavat roolit voivat sisältyä niihin:

- Toteuttajat, kehityspäälliköt ja kehitysjohtajat. Näiden sidosryhmien edustajat toteuttavat testattavan ohjelmiston, ottavat vastaan testitulokset, ja usein heidän on ryhdyttävä toimenpiteisiin raportoitujen tulosten perusteella (esim. korjattava raportoidut viat).
- Tietokanta-arkkitehdit, järjestelmäarkkitehdit ja suunnittelijat. Nämä sidosryhmien edustajat suunnittelevat ohjelmiston, ottavat vastaan testitulokset ja usein heidän on ryhdyttävä toimenpiteisiin raportoitujen tulosten perusteella.
- Markkinointi- ja liiketoiminta-asiantuntijat. Nämä sidosryhmien edustajat määrittelevät ominaisuudet ja niihin liittyvän laatutason, jotka ohjelmiston täytyy sisältää. He ovat myös usein mukana määrittelemässä tarvittavaa testikattavuutta, katselmoimassa testituloksia ja tekemässä päätöksiä testitulosten perusteella.
- Ylempi johto, tuotepäälliköt ja projektin tukihenkilöt. He ovat usein mukana määrittelemässä tarvittavaa testikattavuutta, katselmoimassa testituloksia ja tekemässä päätöksiä testitulosten perusteella.
- Projektipäälliköt. Nämä sidosryhmien edustajat ovat vastuussa projektiensa onnistuneesta läpiviennistä, mikä vaatii laadun, aikataulun, ominaisuuksien ja budjetin tasapainottamista. He hankkivat usein testaustehtävissä tarvittavat resurssit ja tekevät yhteistyötä Testauspäällikön kanssa testauksen suunnittelussa ja hallinnassa.
- Tekninen tuki, asiakastuki ja help deskin henkilöstö. Nämä sidosryhmien edustajat tukevat käyttäjiä ja asiakkaita, jotka hyötyvät toimitetun ohjelmiston ominaisuuksista ja laadusta.
- Suorat ja epäsuorat käyttäjät. Nämä sidosryhmien edustajat käyttävät suoraan ohjelmistoa (eli he ovat loppukäyttäjiä) tai he ottavat vastaan ohjelmiston tuottamia tai tukemia tuotoksia tai palveluita.

Lisää testauksen sidosryhmistä: ks. [Goucher09] luku 2.

Tämä sidosryhmien lista ei ole tyhjentävä. Testauspäälliköiden täytyy tunnistaa heidän projektinsa tai ohjelmistonsa kannalta merkittävät testauksen sidosryhmät. Testauspäällikön täytyy myös ymmärtää, millainen suhde sidosryhmällä tarkalleen on testaukseen ja kuinka testaustiimi voi palvella sidosryhmien tarpeita. Yllä kuvattujen testauksen sidosryhmien tunnistamisen lisäksi Testauspäällikön tulee tunnistaa muut ohjelmistokehityksen elinkaaren tehtävät ja tuotokset, jotka vaikuttavat testaukseen ja/tai joihin testaus vaikuttaa. Ilman tätä testausprosessi ei ehkä saavuta ihanteellista sisäistä ja kokonaistehokkuutta. (ks. kappale 2.2.3).

### 2.2.2 Muita ohjelmistokehityksen elinkaaren tehtäviä ja tuotoksia

Koska ohjelmistotestauksessa arvioidaan yhden tai useamman testaustehtävien ulkopuolella toteutetun tuotoksen laatua, testaus kuuluu yleensä laajempaan ohjelmistokehityksen elinkaaren tehtäväkokonaisuuteen. Testauspäällikön pitää suunnitella testaustehtävät ja ohjata niitä, ja siksi hänen täytyy ymmärtää, kuinka nämä muut tehtävät ja niiden tulokset vaikuttavat testaukseen, kuten todettiin jo Perustason sertifikaattisisällössä, ja kuinka testaus vaikuttaa muihin tehtäviin ja niiden tuloksiin.

Esimerkiksi Ketteriä menetelmiä käyttävissä organisaatioissa kehittäjät tekevät usein testiohjattua toteutusta, luovat automatisoituja yksikkötestejä ja integroivat jatkuvasti koodia (sekä koodin testaamiseen tarvittavia testejä) kokoonpanonhallintajärjestelmään. Testauspäällikön pitää työskennellä kehityspäällikön kanssa sen varmistamiseksi, että testaajat ovat mukana näissä tehtävissä ja toimivat niiden kanssa yhdenmukaisesti. Testaajat voivat katselmoida yksikkötestejä ja tehdä ehdotuksia näiden testien kattavuuden ja tehokkuuden parantamiseksi sekä saadakseen paremman käsityksen ohjelmistosta ja sen toteutuksesta. Testaajat voivat arvioida tapoja, joilla he voivat integroida omia automatisoituja testejään, erityisesti toiminnallisia regressiotestejä, kokoonpanonhallintajärjestelmään. [Crispin09]

Vaikka testaustehtävien, muiden testauksen sidosryhmien sekä ohjelmistokehityksen elinkaaren tehtävien ja tuotosten väliset suhteet vaihtelevat projektin, valitun ohjelmistokehityksen elinkaaren ja monien muiden tekijöiden mukaan, testauksella on keskinäinen yhteys seuraavien tekijöiden kanssa, joihin se liittyy läheisesti:

- Vaatimusten laatiminen ja hallinta. Testauspäällikön täytyy testauksen työmäärää rajatessaan ja arvioidessaan ottaa vaatimukset huomioon sekä pysytellä selvillä vaatimusten muutoksista ja toteuttaa testauksen hallintatehtäviä näiden muutosten vaatimalla tavalla. Teknisen testausasiantuntijan ja Testausasiantuntijan tulisi osallistua vaatimusten katselmointeihin.
- Projektinhallinta. Testauspäällikön tulee Testausasiantuntijan ja Teknisen testausasiantuntijan avustuksella laatia aikataulu ja resurssivaatimukset Projektipäällikköä varten. Testauspäällikön täytyy työskennellä yhdessä Projektipäällikön kanssa, jotta hän ymmärtää projektisuunnitelman muutokset ja voi toteuttaa näiden muutosten vaatimat testauksen hallintatehtävät.
- Kokoonpanonhallinta, julkaisujen hallinta ja muutoksenhallinta. Testauspäällikön tulee yhdessä testaustiimin kanssa laatia testattavan kohteen toimitusprosessit ja –menetelmät sekä kirjata ne testaussuunnitelmaan. Testauspäällikkö voi pyytää Testausasiantuntijaa ja Teknistä testausasiantuntijaa laatimaan testit koonnin todentamiseksi ja varmistamaan versionhallinnan testien suorituksen aikana.
- Ohjelmistokehitys ja ylläpito. Testauspäällikön pitäisi työskennellä yhdessä kehityspäälliköiden kanssa testauskohteiden toimituksen koordinoimiseksi, mukaan lukien jokaisen testijulkaisun sisältö ja päivämäärät, sekä osallistua Vianhallintaan (ks. luku 4).
- Tekninen tuki. Testauspäällikön pitäisi työskennellä yhdessä Teknisen tuen johtajan kanssa, jotta testauksen päätöstehtävien aikana varmistetaan testitulosten asianmukainen toimitus niin, että julkaisun jälkeiseen tuotetukeen osallistuvat henkilöt ovat tietoisia tunnetuista häiriöistä ja kiertoteistä. Tämän lisäksi Testauspäällikön pitäisi osallistua Teknisen tuen johtajan kanssa tuotantohäiriöiden analysointiin, jotta hän voi toteuttaa testausprosessin parannuksia.
- Teknisen dokumentaation toteuttaminen. Testauspäällikön pitäisi työskennellä Teknisen dokumentaation johtajan kanssa oikea-aikaisen testausdokumentaation toimittamisen sekä kyseisistä dokumenteista löytyneiden vikojen hallinnoinnin varmistamiseksi.

Yllä kuvattujen testauksen sidosryhmien tunnistamisen lisäksi Testauspäällikön täytyy tunnistaa muut ohjelmistokehityksen elinkaaren tehtävät ja tuotokset, jotka vaikuttavat testaukseen ja/tai joihin testaus vaikuttaa. Mikäli näin ei tehdä, testausprosessi ei saavuta optimaalista sisäistä ja kokonaistehokkuutta.

### 2.2.3 Testaustehtävien ja muiden elinkaaren tehtävien yhteensovittaminen

Testauksen pitäisi olla keskeinen osa projektia riippumatta siitä, mitä ohjelmistokehitysmallia käytetään. Näihin kuuluvat:

- peräkkäismallit, kuten vesiputousmalli, V-malli ja W-malli. Peräkkäismallissa kaikki tietyn vaiheen tuotokset ja tehtävät (esim. vaatimukset, suunnittelu, toteutus, yksikkötestaus, integraatiotestaus, järjestelmätestaus ja hyväksymistestaus) tehdään loppuun ennen seuraavan vaiheen alkua. Testauksen suunnittelu, analysointi, testien suunnittelu ja toteutus etenevät samanaikaisesti projektisuunnittelun, liiketoiminta-/vaatimusanalyysin, ohjelmisto- ja tietokantasuunnittelun ja ohjelmoinnin kanssa, ja päällekkäisyyden osuus riippuu kyseessä olevasta testaustasosta. Testauksen suoritus etenee testausvaiheittain Perustason sertifikaattisisällössä ja tässä sertifikaattisisällössä kuvatulla tavalla.
- Iteratiiviset tai inkrementaaliset mallit, kuten Rapid Application Development (RAD) ja Rational Unified Process (RUP). Iteratiivisessa tai inkrementaalisessa mallissa toteutettavat ominaisuudet ryhmitellään (esim. liiketoiminnan prioriteetin tai riskin mukaan) ja sen jälkeen jokainen ryhmä käy läpi projektin eri vaiheet, mukaan luettuna niiden tuotokset ja tehtävät. Vaiheet voidaan suorittaa joko peräkkäin tai toisensa kanssa rinnakkain, ja itse iteraatiot voivat olla peräkkäisiä tai päällekkäin meneviä. Projektin käynnistyksen aikana tehdään korkean tason testauksen suunnittelu sekä analysointi samaan aikaan projektisuunnittelun ja liiketoiminta-/vaatimusanalyysin kanssa. Yksityiskohtainen testauksen suunnittelu, testien analysointi sekä testien suunnittelu ja toteutus tapahtuvat joka iteraation alussa samanaikaisesti. Eri testaustasojen testien suoritus tapahtuu usein samanaikaisesti. Jokainen testaustaso käynnistetään niin aikaisin kuin mahdollista ja se voi jatkua, vaikka sitä seuraavat ylemmät testaustasot ovat alkaneet.
- Ketterät menetelmät, kuten scrum ja Extreme Programming (XP). Nämä ovat iteratiivisia elinkaarimalleja, joissa iteraatiot ovat hyvin lyhyitä (usein kahdesta neljään viikkoon). Jokaisen iteraation tuotokset ja tehtävät viedään loppuun ennen seuraavan iteraation alkua (eli iteraatiot ovat peräkkäisiä). Testaus etenee samalla tavalla kuin iteratiivisissa malleissa mutta eri testaustehtävät suoritetaan enemmän päällekkäin kehitystehtävien kanssa, mikä tarkoittaa myös testauksen suorituksen huomattavaa samanaikaisuutta kehitystehtävien kanssa. Kaikkien iteraation tehtävien, testaustehtävät mukaan luettuna, pitäisi olla valmiita ennen seuraavan iteraation aloitusta. Ketterässä projektissa Testauspäällikön rooli vaihtelee usein suorasta esimiesroolista teknisen asiantuntijan/neuvonantajan rooliin.
- Spiraalimalli. Spiraalimallissa käytetään prototyyppejä aikaisin projektissa sekä varmistamaan järjestelmän toteutuskelpoisuus että suunnittelu- ja toteutuspäätösten testaamiseksi, ja liiketoiminnan prioriteetteja ja teknisiä riskejä käytetään apuna, kun valitaan, missä järjestyksessä prototyyppikokeilut suoritetaan. Nämä prototyypit testataan sen määrittämiseksi, mitkä teknisten ongelmien osat ovat vielä ratkaisematta. Kun tärkeimmät tekniset ongelmat on ratkaistu, projekti etenee joko peräkkäismallin tai iteratiivisen mallin mukaan.

Voidakseen sovittaa testaustehtävät kunnolla järjestelmän elinkaareen Testauspäällikön täytyy ymmärtää yksityiskohtaisesti organisaatiossa käytettäviä elinkaarimalleja. Esimerkiksi V-mallissa järjestelmään sovellettava ISTQB:n perustestausprosessi voidaan linjata seuraavasti:

- Järjestelmätestauksen suunnittelutehtävät tapahtuvat yhtäaikaisesti projektisuunnittelun kanssa, ja testauksen seuranta jatkuu, kunnes järjestelmätestauksen suoritus ja päätöstehtävät on saatu loppuun.
- Järjestelmätestauksen analysointi ja testien suunnittelutehtävät tapahtuvat yhtäaikaisesti vaatimusmäärittelyjen, järjestelmän ja arkkitehtuurin (korkean tason) suunnittelukuvausten ja yksikkötestauksen (matalan tason) suunnittelukuvausten laatimisen kanssa.
- Järjestelmätestauksen toteutustehtävät voivat alkaa järjestelmäsuunnittelun aikana, vaikka suurin osa näistä tehtävistä tapahtuu tyypillisesti samaan aikaan koodauksen ja yksikkötestauksen kanssa niin, että järjestelmätestauksen toteutustehtävät jatkuvat usein aivan viime päiviin asti ennen järjestelmätestauksen suoritusta.
- Järjestelmätestauksen suoritustehtävät alkavat, kun järjestelmätestauksen aloituskriteerit on kaikki täytetty (tai niistä on luovuttu), mikä tyypillisesti tarkoittaa, että ainakin yksikkötestaus ja usein myös komponentti-integraatiotestaus ovat valmiita. Järjestelmätestauksen suoritus jatkuu, kunnes järjestelmätestauksen päätöskriteerit täyttyvät.

- Järjestelmätestauksen päätöskriteerien arviointia ja testitulosten raportointia tehdään läpi järjestelmätestauksen suorituksen, yleensä sitä tiheämmin ja nopeammin mitä lähemmäksi projektin aikaraja tulee.
- Järjestelmätestauksen päätöstehtävät suoritetaan sen jälkeen, kun päätöskriteerit ovat täyttyneet ja järjestelmätestaus on vahvistettu suoritetuksi, vaikka niitä voidaan joskus lykätä, kunnes hyväksymistestaus on suoritettu ja kaikki projektin tehtävät on saatu valmiiksi.

Iteratiivisessa tai inkrementaalisissa elinkaarimallissa on suoritettava samat tehtävät, mutta niiden ajoitus ja laajuus voivat vaihdella. Esimerkiksi sen sijaan, että projektin alussa asennettaisiin koko testausympäristö valmiiksi, voi olla tuottavampaa toteuttaa vain se osa, jota tarvitaan kyseisessä iteraatiossa. Mitä pidemmälle suunnittelu tapahtuu kaikissa iteratiivisissa tai inkrementaalisissa malleissa, sitä pidemmälle voidaan laajentaa perustestausprosessin soveltamista.

Kaikissa projektissa tapahtuvien suunnitteluvaiheiden lisäksi tiimin käyttämä elinkaarimalli voi vaikuttaa myös testien suoritukseen ja raportointiin. Esimerkiksi iteratiivisessa elinkaarimallissa voi olla tehokasta laatia täydelliset raportit ja suorittaa iteraation jälkeisiä katselmointi-istuntoja ennen seuraavan iteraation aloittamista. Kun jokaista iteraatiota käsitellään pienoisprojektina, tiimi saa mahdollisuuden tehdä korjauksia ja muokkauksia sen perusteella, mitä tapahtui edellisessä iteraatiossa. Koska iteraatiot voivat olla lyhyitä ja ajallisesti rajattuja, voi olla järkevää pienentää raportointiin ja arviointiin käytettävää aikaa ja työmäärää, mutta nämä tehtävät pitäisi suorittaa niin, että voidaan seurata testauksen kokonaisedistymistä ja tunnistaa ongelma-alueet mahdollisimman pian. Iteraatiossa koetut prosessiin liittyvät ongelmat voivat helposti vaikuttaa seuraavaan iteraatioon ja jopa toistua sen aikana, mikäli korjaaviin toimenpiteisiin ei ryhdytä.

Yleistä tietoa siitä, kuinka testaus sovitetaan yhteen muiden elinkaaren tehtävien kanssa, voidaan kirjata Testausstrategiaan (ks. kappale 2.4.2). Testauspäällikön pitäisi testaussuunnittelun ja/tai projektisuunnittelun aikana suorittaa projektikohtainen yhteensovittaminen joka testaustason osalta sekä valituille ohjelmistokehityksen elinkaaren ja testausprosessin tehtävien yhdistelmille.

Organisaation, projektin ja tuotteen tarpeista riippuen voidaan tarvita muitakin testaustasoja Perustason sertifikaattisisällössä määriteltyjen lisäksi, kuten:

- Laitteiston ja ohjelmiston integraatiotestaus
- Järjestelmäintegraatiotestaus
- Ominaisuuksien vuorovaikutuksen testaus
- Asiakastuotteen integraatiotestaus.

Jokaiselle testaustasolle pitää selkeästi määrittää seuraavat asiat:

- Testauksen tavoitteet ja saavutettavissa olevat päämäärät
- Testauksen laajuus ja testattavat nimikkeet
- Testauksen pohjamateriaali sekä keinot, jolla arvioidaan pohjamateriaalin kattavuus (eli jäljitettävyys)
- Aloitus- ja lopetuskriteerit
- Testauksen tuotokset, mukaan luettuna tulosten raportointi
- Soveltuvat testaustekniikat sekä keinot, joilla varmistetaan, että kyseisillä tekniikoilla saavutetaan riittävä kattavuus
- Testauksen tavoitteiden, aloitus- ja päätöskriteereiden sekä tulosten raportoinnin kannalta keskeiset mittarit ja metriikat (mukaan luettuna kattavuusmittarit)
- Eri testaustehtävissä käytettävät testaustyökalut (jos käytetään)
- Resurssit (esim. testiympäristö)
- Tiimiin kuuluvat ja tiimin ulkopuoliset vastuuhenkilöt ja -ryhmät
- Yhdenmukaisuus organisaation, säädöstenmukaisten tai muiden standardien kanssa (jos tarvitaan).

Kuten tässä luvussa myöhemmin todetaan, paras tapa on määritellä nämä asiat yhdenmukaisesti läpi kaikkien testaustasojen, jotta vältetään turhat ja vaaralliset aukot eri tasoilla tehtävien samanlaisten testien välillä.

### 2.2.4 Ei-toiminnallisen testauksen hallinta

Ei-toiminnallisten testien suunnittelematta jättäminen voi johtaa vakavien, joskus tuhoisien laatuongelmien löytymiseen järjestelmästä julkaisun jälkeen. Monet ei-toiminnallisen testaustyypit ovat kuitenkin kalliita, joten Testauspäällikön on valittava riskien ja rajoitteiden perusteella, mitä ei-toiminnallisia testejä suoritetaan. Tämän lisäksi on olemassa monenlaisia ei-toiminnallisia testejä, joista jotkut eivät välttämättä sovi kyseisen sovelluksen testaukseen.

Koska Testauspäälliköllä ei ehkä ole riittävästi asiantuntemusta kaikkien suunnittelussa huomioon otettavien asioiden hallitsemiseksi, hänen täytyy delegoida osa testisuunnittelun vastuusta Tekniselle Testausasiantuntijalle (ja joissain tapauksissa Testausasiantuntijalle), jolle ei-toiminnallisen testauksen tehtävät on osoitettu tehtäväksi. Testauspäällikön pitäisi pyytää asiantuntijoita ottamaan huomioon seuraavat yleiset tekijät:

- Sidosryhmien vaatimukset
- Tarvittavat työkalut
- Testiympäristö
- Organisatoriset tekijät
- Tietoturva.

Tarkempia tietoja: Ks. Jatkotason sertifikaattisisältö, Tekninen testausasiantuntija [ISTQB ATTA SYL]

Toinen tärkeä asia Testauspäällikölle huomioon otettavaksi on se, kuinka ei-toiminnalliset testit integroidaan ohjelmistokehityksen elinkaareen. Tyypillinen virhe on odottaa ei-toiminnallisten testien aloittamista, kunnes kaikki toiminnalliset testit ovat valmiit, mikä voi johtaa kriittisten ei-toiminnallisten vikojen myöhäiseen löytymiseen. Sen sijaan ei-toiminnalliset testit pitäisi priorisoida ja ryhmitellä riskin perusteella. Usein on olemassa tapoja pienentää ei-toiminnallisia riskejä testauksen varhaisessa vaiheessa tai jopa toteutuksen aikana. Esimerkiksi järjestelmän suunnittelun aikana tehtävät käyttöliittymäprototyyppien käytettävyyskatselmoinnit voivat olla varsin tehokkaita paljastamaan käytettävyysvikoja, jotka voisivat aiheuttaa merkittäviä aikatauluongelmia, mikäli ne löydettäisiin järjestelmätestauksen loppuvaiheessa.

Iteratiivisissa elinkaarimalleissa muutosten ja iteraatioiden nopeus voi hankaloittaa keskittymistä tiettyihin ei-toiminnallisiin testeihin, jotka vaativat monimutkaisten testikehysten rakentamista. Yksittäistä iteraatiota pidemmän ajan vaativat testien suunnittelu- ja toteutustehtävät pitäisi järjestää erillisiksi iteraatioiden ulkopuolisiksi tehtäviksi.

### 2.2.5 Kokemuspohjaisen testauksen hallinta

Vaikka kokemuspohjainen testaus tuottaakin hyötyjä paljastamalla tuottavasti vikoja, jotka voivat muita tekniikoita käytettäessä jäädä huomaamatta, ja toimimalla muiden tekniikoiden kattavuuden täydentäjänä, se luo myös haasteita testauksen hallinnalle. Testauspäällikön pitäisi olla tietoinen kokemuspohjaisten tekniikoiden, erityisesti tutkivan testauksen, haasteista samoin kuin hyödyistä. Tämän tyyppisen testauksen aikana on vaikea määritellä saavutettu kattavuus testauksen tyypillisesti keveän dokumentoinnin ja testien minimaalisen etukäteisvalmistelun vuoksi. Testitulosten toistettavuus vaatii erityistä hallinnollista huomiota erityisesti silloin, kun mukana on useita testaajia.

Yksi tapa kokemuspohjaisen testauksen ja erityisesti tutkivan testauksen hallintaan on jakaa työ pieniin, 30 – 120 minuutin jaksoihin, joita joskus kutsutaan testisessioiksi. Tämä aikarajoite rajoittaa ja keskittää sessiossa tehtävän työn ja tuo tietyn tasoista monitorointia ja aikataulutusta. Jokaiseen session kuuluu testiohje, jonka Testauspäällikkö antaa testaajalle joko kirjallisesti tai suullisesti. Testiohje sisältää testaussessiossa käsiteltävän tai käsiteltävät testattavat tilanteet, mikä myös helpottaa keskittymistä ja vähentää päällekkäisyyttä, jos useita ihmisiä tekee samaan aikaan tutkivaa testausta.

Yksi tapa hallita kokemuspohjaista testausta on integroida tämä itseohjautuva ja spontaani testaus perinteisempiin, ennalta suunniteltuihin testausistuntoihin. Testaajille voidaan esimerkiksi antaa lupa (ja määrätty aika) tutkia ennalta määritellyissä testeissä kuvattujen askelien, syötteiden ja odotettujen tulosten ulkopuolelta. Testaajille voidaan myös antaa suoritettavaksi tällaisia itseohjautuvia testaussessioita osana heidän päivittäistä testaustaan ennen tai jälkeen ennalta määriteltyjen testien suorituksen tai

sen aikana. Jos tällaisissa testausistunnoissa tunnistetaan vikoja tai jatkotestauksen kannalta kiinnostavia alueita, aiemmin määriteltyjä testejä voidaan päivittää.

Tutkivan testauksen istunnon alussa testaaja selvittää ja suorittaa testauksen kannalta tarvittavat alkujärjestelyt. Istunnon aikana testaaja oppii testattavasta järjestelmästä, suunnittelee ja suorittaa oppimansa perusteella testejä valittua testaustekniikkaa käyttäen, tutkii mahdollisia vikoja ja kirjaa testin tulokset testipäiväkirjaan. (Jos testien toistettavuus on tarpeen, testaajan pitää kirjata ylös myös testin syötteet, toimenpiteet ja tapahtumat.) Istunnon jälkeen voidaan pitää jälkipalaveri, joka antaa suunnan seuraaville istunnoille.

## 2.3 Riskipohjainen testaus ja muita lähestymistapoja testien priorisointiin ja työmäärien kohdentamiseen

Hyvin yleinen testauksen hallintaan liittyvä haaste on testien oikea valinta, kohdentaminen ja priorisointi. Katettavaksi tarkoitetusta testattavien tilanteiden ja niiden yhdistelmien käytännössä lähes rajattomasta joukosta testaustiimin pitää valita rajallinen joukko tilanteita, määrittää sopiva työmäärä kohdennettavaksi niin, että jokainen näistä tilanteista katetaan testitapauksilla, ja järjestää tuloksena syntyvät testitapaukset priorisoituun järjestykseen, jossa tehtäväksi tarkoitetun yleinen ja toiminnallinen testaustyön tehokkuus on optimoitu. Testauspäällikkö voi käyttää tämän ongelman ratkaisemiseen riskien tunnistamista ja analysointia sekä muita menetelmiä, vaikka monet asiaan liittyvät rajoitteet ja muuttujat voivat vaatia ratkaisuksi kompromissia.

### 2.3.1 Riskipohjainen testaus

Riski tarkoittaa negatiivisen tai ei-toivotun lopputuloksen tai tapahtuman mahdollisuutta. Riski on olemassa aina, kun esiin voi tulla joku ongelma, joka heikentää asiakkaan, käyttäjän, asianosaisen tai sidosryhmän näkemystä tuotteen laadusta tai projektin onnistumisesta. Kun mahdollisen ongelman päävaikutus liittyy tuotteen laatuun, ongelmaa kutsutaan laaturiskiksi, tuoteriskiksi tai tuotelaaturiskiksi. Kun mahdollisen ongelman päävaikutus liittyy projektin onnistumiseen, ongelmaa kutsutaan projektiriskiksi tai suunnitteluriskiksi.

Riskipohjaisessa testauksessa tunnistetaan laaturiskit ja ne analysoidaan tuotteen laaturiskianalyysissä yhdessä sidosryhmien edustajien kanssa. Tämän jälkeen testaustiimi suunnittelee, toteuttaa ja suorittaa testit laaturiskien pienentämiseksi. Laatu tarkoittaa ominaisuuksien, käyttäytymisen, piirteiden ja tunnusmerkkien kokonaisuutta, joka vaikuttaa asiakkaan, käyttäjän ja sidosryhmien tyytyväisyyteen. Siksi laaturiski tarkoittaa tilannetta, jolloin tuotteessa saattaa olla laatuongelmia. Järjestelmän laaturiskeistä ovat esimerkkejä virheelliset laskutoimitukset raporteissa (toiminnallinen tarkkuuteen liittyvä riski), hidas vaste käyttäjän antamaan syötteeseen (ei-toiminnallinen tehokkuuteen ja vasteaikaan liittyvä riski) sekä vaikeudet ymmärtää näyttöjä ja kenttiä (ei-toiminnallinen käytettävyyteen ja ymmärrettävyyteen liittyvä riski). Kun testit paljastavat vikoja, testaus on pienentänyt laaturiskejä tuottamalla tietoa vioista ja antamalla mahdollisuuden käsitellä ne ennen julkaisua. Kun testaus ei löydä vikoja, testaus on pienentänyt laaturiskejä varmistamalla että järjestelmä toimii oikein testauksenaikaisissa olosuhteissa.

Riskipohjainen testaus käyttää tuoteriskejä testattavien tilanteiden valitsemisessa, työmäärien kohdentamisessa näihin tilanteisiin ja tuloksena syntyvien testitapausten priorisointiin. Riskipohjaista testausta varten on olemassa joukko erilaisia tekniikoita, jotka vaihtelevat merkittävästi sekä kerätyn dokumentaation tyypin ja sen tason että käytettävän muodollisuuden tason suhteen. Ajatellaanpa eksplisiittisesti tai implisiittisesti, riskipohjaisen testauksen tavoitteena on käyttää testausta vähentämään laaturiskien kokonaistasoa ja erityisesti pienentää kyseistä riskitasoa hyväksyttävälle tasolle.

Riskipohjainen testaus käsittää seuraavat neljä päätehtävää:

- riskien tunnistaminen
- riskien arviointi
- riskien lieventäminen
- riskien hallinta.

Nämä tehtävät menevät limittäin toistensa kanssa. Seuraavat alakappaleet käsittelevät jokaista näistä tehtävistä erikseen.

Jotta riskien tunnistaminen ja arviointi toimisivat tehokkaimmin, niihin pitäisi osallistua edustajia kaikista projektin ja tuotteen sidosryhmistä, vaikka joskus projektin realiteetit johtavat siihen, että jonkun sidosryhmän edustajat toimivat toisen sidosryhmän edustajina. Kun esimerkiksi kehitetään massamarkkinoille tarkoitettua ohjelmistoa, pientä potentiaalisten asiakkaiden ryhmää voidaan pyytää auttamaan sellaisten mahdollisten vikojen tunnistamisessa, jotka heidän kohdallaan vaikuttavat ohjelmiston käyttöön kaikkein eniten; tässä tapauksessa tämä pieni ryhmä toimii koko lopullisen asiakaskunnan edustajana. Koska testaajilla on erityisasiantuntemusta tuotteen laaturiskeistä ja häiriöistä, heidän pitäisi osallistua aktiivisesti riskien tunnistamis- ja arviointiprosessiin.

#### 2.3.1.1 Riskien tunnistaminen

Sidosryhmien edustajat voivat tunnistaa riskejä käyttämällä yhtä tai useampia seuraavista tekniikoista:

- Asiantuntijahaastattelut
- Riippumattomat arvioinnit
- Riskiluetteloiden käyttö
- Projektin jälkipalaverit
- Riskityöpajat
- Aivoriihet
- Tarkistuslistat
- Aikaisempien kokemusten hyödyntäminen.

Kun mukaan otetaan laajin mahdollinen sidosryhmien edustajien joukko, löydetään riskien tunnistamisprosessissa todennäköisesti suurin osa merkittävistä tuotelaaturiskeistä.

Riskien tunnistaminen tuottaa usein sivutuotteita, eli löydetään ongelmia, jotka eivät ole tuotelaaturiskejä. Esimerkkejä näistä ovat tuotteeseen tai projektiin liittyvät yleiset kysymykset tai ongelmat, tai ongelmat viitedokumenteissa, kuten vaatimus- ja suunnittelukuvauksissa. Laaturiskien tunnistamisen sivutuotteena tunnistetaan usein myös projektiriskejä, mutta ne eivät ole riskipohjaisen testauksen pääkohde. Projektiriskien hallinta on kuitenkin tärkeää koko testauksen kannalta, ei vain riskipohjaisessa testauksessa, ja sitä käsitellään tarkemmin kappaleessa 2.4.

#### 2.3.1.2 Riskien arviointi

Riskien tunnistamisen jälkeen voi alkaa riskien arviointi, joka tarkoittaa näiden tunnistettujen riskien tutkimista. Riskien arviointiin kuuluu erityisesti jokaisen riskin luokittelu ja riskeihin liittyvän todennäköisyyden ja vaikutuksen määrittäminen. Riskien arviointiin voi liittyä myös muiden riskiin liittyvien seikkojen, kuten esimerkiksi riskin omistajan, arviointi tai määrittäminen.

Riskin luokittelu tarkoittaa jokaisen riskin sijoittamista sopivaan riskityyppien ryhmään, kuten esimerkiksi suorituskyky, luotettavuus, toiminnallisuus jne. Jotkut organisaatiot käyttävät luokittelussa ISO 9126 standardia ([ISO9126] (joka on korvautumassa ISO 25000 standardilla [ISO25000]), mutta monet organisaatiot käyttävät muita luokittelumalleja. Riskien tunnistamisessa käytettävää tarkistuslistaa käytetään usein myös riskien luokittelussa. On olemassa yleisiä laaturiskien tarkistuslistoja, joita monet organisaatiot räätälöivät omaan käyttöönsä. Kun tarkistuslistoja käytetään riskien tunnistamisen perustana, riskin luokittelu tapahtuu usein tunnistamisen yhteydessä.

Riskitason määrittämiseen kuuluu tyypillisesti riskin toteutumisen todennäköisyyden sekä toteutumisesta seuraavan vaikutuksen arviointi, joka tehdään jokaiselle riskiaiheelle. Toteutumisen todennäköisyydellä tarkoitetaan sitä, kuinka todennäköisesti testattavassa järjestelmässä on mahdollisia ongelmia. Todennäköisyys tarkoittaa toisin sanoen teknisen riskin tason arviointia. Tuote- ja projektiriskien todennäköisyyteen vaikuttavia seikkoja ovat mm. seuraavat:

- teknologian ja tiimien monimutkaisuus
- henkilöstöön ja koulutukseen liittyvät seikat liiketoiminta-asiantuntijoiden, suunnittelijoiden ja toteuttajien osalta
- tiimin sisäiset ristiriidat
- toimittajiin liittyvät sopimukselliset ongelmat

- maantieteellisesti hajautettu tiimi
- perinteiset vs. uudet lähestymistavat
- työkalut ja teknologia
- heikko hallinnollinen tai tekninen johto
- aikaan, resursseihin, budjettiin ja johtoon liittyvät paineet
- aiempien laadunvarmistustehtävien puute
- suuri muutosten määrä
- suuri aiemmin löytyneiden vikojen määrä
- rajapintoihin ja integraatioihin liittyvät ongelmat.

Toteutumisen vaikutus tarkoittaa sitä, kuinka vakavasti tapahtuma vaikuttaa käyttäjiin, asiakkaisiin tai muihin sidosryhmien edustajiin. Projekti- ja tuoteriskien vaikutukseen liittyviä seikkoja ovat mm. seuraavat:

- kohteena olevan ominaisuuden käyttötiheys
- ominaisuuden kriittisyys liiketoimintatavoitteiden saavuttamisen kannalta
- maineelle aiheutuva haitta
- liiketoiminnan menetys
- mahdolliset taloudelliset, ekologiset tai sosiaaliset menetykset tai korvausvelvollisuudet
- siviili- tai rikosoikeudelliset rangaistukset
- lisenssin menetys
- järkevien vaihtoehtoisten toimintatapojen puute
- negatiiviseen julkisuuteen johtavien häiriöiden näkyvyys
- turvallisuus.

Riskitasoa voidaan arvioida numeerisesti tai laadullisesti. Jos todennäköisyydelle ja vaikutukselle voidaan antaa numeeriset arvot, kaksi arvoa voidaan kertoa keskenään ja laskea näin numeerinen riskin prioriteettiarvo. Tyypillisesti riskitaso voidaan kuitenkin arvioida vain laadullisesti. Tämä tarkoittaa, että todennäköisyys voidaan kuvata hyvin korkeaksi, korkeaksi, keskitasoiseksi, matalaksi tai hyvin matalaksi, mutta todennäköisyyttä ei voida esittää todellista tarkkuutta ilmaisevana prosenttiosuutena; samalla tavalla vaikutuksesta voidaan puhua hyvin korkeana, korkeana, keskitasoisena, matalana tai hyvin matalana, mutta sitä ei voida ilmaista taloudellisin termein kattavasti tai täsmällisesti. Riskitasojen laadullista arviointia ei pitäisi pitää määrällistä huonompana; itse asiassa kun riskitasojen määrällisiä arvioita käytetään ei-tarkoituksenmukaisella tavalla, tulokset antavat sidosryhmien edustajille harhaanjohtavaa tietoa siitä, missä määrin riskit on ymmärretty ja niitä voidaan hallita. Riskitasojen laadulliset arviot yhdistetään usein kertomalla tai laskemalla ne yhteen koostetuksi riskiarvioksi. Tätä koostearviota voidaan pitää riskien prioriteettiarvona, mutta sitä pitäisi käsitellä järjestysasteikkoon pohjautuvana laadullisena, suhteellisena arviona.

Jos riskianalyysi ei perustu laajaan ja tilastollisesti merkittävään riskiaineistoon, riskianalyysi pohjautuu sidosryhmien edustajien subjektiiviseen näkemykseen todennäköisyydestä ja vaikutuksesta. Projektipäälliköillä, toteuttajilla, käyttäjillä. liiketoiminta-asiantuntijoilla, arkkitehdeillä ja testaajilla on tyypillisesti erilaiset näkemykset ja näin ollen mahdollisesti erilaiset mielipiteet kunkin riskikohteen riskitasosta. Riskianalyysiprosessin pitäisi sisältää keinot yhteisen näkemyksen aikaansaamiseksi tai, pahimmassa tapauksessa, ennalta sovitun (esim. johdon ohjeistuksen mukaisen tai riskikohteen keskiarvon, mediaanin tai moodin) riskitason määrittämiseksi. Tämän lisäksi pitäisi tarkistaa, että riskitasojen jakauma käytetyllä asteikolla on hyvä sen varmistamiseksi, että riskiarviot tuottavat käyttökelpoista ohjeistusta testien järjestyksen, priorisoinnin ja työmäärien kohdentamisen suunnittelua varten. Muussa tapauksessa riskitasoja ei voi käyttää ohjaamaan riskien lieventämiseen liiittyviä tehtäviä.

#### 2.3.1.3 Riskien lieventäminen

Riskipohjainen testaus alkaa laaturiskien analyysilla (tuotelaaturiskien tunnistaminen ja arviointi). Tämä analyysi on kokonaistestaussuunnitelman ja muiden testaussuunnitelmien perusta. Testit suunnitellaan, toteutetaan ja suoritetaan suunnitelmassa (tai suunnitelmissa) kuvatulla tavalla riskien kattamiseksi. Testin laatimiseen ja suorittamiseen tarvittava työmäärä on suhteessa riskin tasoon, mikä tarkoittaa että yksityiskohtaisempia testisuunnittelutekniikoita (kuten esimerkiksi syötteiden pareittainen testaus) käytetään korkeampiin riskeihin, kun taas vähemmän yksityiskohtaisempia testisuunnittelutekniikoita (kuten

esimerkiksi ekvivalenssiositus tai aikarajattu tutkiva testaus) käytetään matalampiin riskeihin. Tämän lisäksi testien suunnittelun ja suorituksen priorisointi perustuu riskitasoon. Jotkut turvallisuuteen liittyvät standardit (esim. FAA DO-178B/ED 12B, IEC 61508) määräävät käytettävät testisuunnittelutekniikat ja kattavuustason riskitason perusteella. Tämän lisäksi riskitaso pitäisi ottaa huomioon, kun tehdään päätöksiä projektin tuotosten katselmoinnista (mukaan luettuna testitapaukset), riippumattomuuden tasosta, testaajien kokemuksen tasosta sekä suoritettavan varmistustestauksen (uudelleentestauksen) ja regressiotestauksen määrästä.

Projektin aikana testaustiimin pitäisi pysytellä tietoisena uusista tiedoista, jotka muuttavat laaturiskien joukkoa ja/tai tunnettuihin laaturiskeihin liittyvää riskitasoa. Laaturiskien analyysiin pitäisi tietyin ajoin tehdä muutoksia, jotka johtavat testitapausten muutoksiin. Näitä muutoksia pitäisi tehdä ainakin silloin, kun saavutetaan merkittävämpiä tarkastuspisteitä. Muutoksiin kuuluvat uusien riskien tunnistaminen, olemassa olevien riskien tason uudelleenarviointi ja riskien lieventämistehtävien tehokkuuden arviointi. Jos esimerkiksi riskien tunnistaminen ja arviointi tehtiin vaatimusmäärittelyvaiheen aikana niin, että se perustui senhetkisin vaatimuksiin, pitää riskit arvioida uudelleen sen jälkeen kun suunnittelukuvaukset on saatu valmiiksi. Vastaavasti jos testauksen aikana komponentista löytyy huomattavasti odotettua enemmän vikoja, voidaan päätellä, että vikojen todennäköisyys kyseisellä alueella oli odotettua korkeampi ja tämän perusteella muokata todennäköisyyden arviota ja kokonaisriskitasoa ylöspäin. Tämä voi johtaa kyseisen komponentin testausmäärän kasvattamiseen.

Tuotelaaturiskejä voidaan pienentää jo ennen kuin testauksen suoritus alkaa. Jos esimerkiksi riskien tunnistamisen aikana havaitaan ongelmia vaatimuksissa, projektitiimi voi riskejä pienentävänä toimenpiteenä suorittaa perusteellisia vaatimuskuvausten katselmointeja. Tämä voi pienentää riskitasoa, mikä saattaa tarkoittaa, että tarvitaan vähemmän testejä jäljellä olevien laaturiskien pienentämiseen.

#### 2.3.1.4 Riskien hallinta ohjelmiston elinkaaressa

Ihannetilanteessa riskienhallintaa tehdään koko ohjelmiston elinkaaren aikana. Mikäli organisaatiolla on testauspolitiikka ja/tai testausstrategia, näissä pitäisi määritellä yleinen prosessi, jonka mukaan tuote- ja projektiriskejä hallitaan testauksessa, ja kuvata kuinka riskienhallinta on sulautettu kaikkiin testausvaiheisiin ja kuinka se vaikuttaa niihin.

Kypsässä organisaatiossa, jossa koko projektitiimi on tietoinen riskeistä, riskienhallintaa tapahtuu monilla eri tasoilla eikä vain testauksessa. Tärkeisiin riskeihin ei pelkästään puututa aikaisemmin jollakin tietyllä testaustasolla, vaan myös jo aiemmilla testaustasoilla. (Jos esimerkiksi suorituskyky on tunnistettu laaturiskien avainalueeksi, suorituskykytestaus ei ala pelkästään aikaisin järjestelmätestauksessa, vaan suorituskykytestejä suoritetaan yksikkö- ja integrointitestauksessa.) Kypsät organisaatiot eivät tunnista pelkästään riskejä, vaan myös riskien lähteet ja riskien seuraukset, jos riskit toteutuvat. Toteutuneiden vikojen osalta käytetään perussyyanalyysiä riskien lähteiden perusteellisempaan ymmärtämiseen ja prosessin parantamiseen niin, että viat pyritään jo alun perin estämään. Riskien lieventämistä tapahtuu läpi koko ohjelmistokehityksen elinkaaren. Riskianalyysi käyttää kaikkia tietoja ja ottaa huomioon tilanteeseen liittyvät työtehtävät, järjestelmän käyttäytymisen analyysin, kustannuspohjaisen riskiarvioinnin, tuoteriskianalyysin, loppukäyttäjien riskianalyysin ja korvausvelvollisuuksien riskianalyysin. Riskianalyysi ylittää testauksen rajat ja testaustiimi osallistuu koko ohjelmiston riskianalyysiin ja vaikuttaa siihen.

Useimmat riskipohjaiset testausmenetelmät käyttävät myös riskitasoon pohjautuvia tekniikoita testauksen järjestämiseen ja priorisointiin, jolla varmistetaan tärkeimpien alueiden kattaminen sekä tärkeimpien vikojen löytäminen aikaisin testauksen suorituksessa. Joissain tapauksissa kaikki korkeimman tason riskien testit suoritetaan ennen alemman tason riskien testejä ja testit suoritetaan tiukasti riskijärjestyksessä (kutsutaan usein “syvyysjärjestykseksi”); toisissa tapauksissa käytetään näytteenottolähestymistapaa, jolloin kaikkien tunnistettujen riskiaiheiden joukosta valitaan joukko testejä käyttämällä riskiä painottamassa valintaa samalla kuitenkin varmistamalla, että jokainen riskiaihe tulee katettua ainakin kerran (kutsutaan usein ”leveysjärjestykseksi”).

Käytetäänpä riskipohjaisessa testauksessa ensimmäisenä sitten syvyys- tai leveysjärjestystä, on mahdollista, että testaukseen varattu aika loppuu ennen kuin kaikki testit on suoritettu. Riskipohjainen tes-

taus luo testaajalle mahdollisuuden raportoida johdolle senhetkisen riskitason perusteella ja antaa johdolle mahdollisuuden päättää, laajennetaanko testausta vai siirretäänkö jäljellä olevat riskit käyttäjien, asiakkaan, help deskin/teknisen tuen ja/tai operaattoreiden hoidettavaksi.

Testien suorituksen aikana kaikkein edistyneimmät riskipohjaiset testaustekniikat – joiden ei tarvitse olla kaikkein muodollisimpia tai raskaimpia – antavat projektin jäsenille, projekti- ja tuotepäälliköille, esimiehille, johtajille ja projektin sidosryhmille mahdollisuuden seurata ja hallita ohjelmistokehityksen elinkaarta jäljellä olevan riskitason pohjalta ja tehdä sen perusteella mm. julkaisupäätöksiä. Tämä edellyttää, että Testauspäällikkö raportoi testauksen tulokset riskien suhteen tavalla, jonka kaikki testauksen sidosryhmät ymmärtävät.

### 2.3.2 Riskipohjaiset testaustekniikat

Riskipohjaisia testaustekniikoita on paljon. Jotkut näistä tekniikoista ovat hyvin epämuodollisia. Esimerkkinä voidaan mainita lähestymistavat, joissa testaaja analysoi laaturiskejä tutkivan testauksen aikana [Whittaker09]. Tämä voi auttaa ohjaamaan testausta, mutta voi johtaa myös siihen, että keskitytään liikaa vikojen todennäköisyyteen, ei niiden vaikutukseen, eikä tässä tavassa oteta huomioon organisaation poikittaisilta sidosryhmiltä tulevaa panosta. Lisäksi tällaiset lähestymistavat ovat subjektiivisia ja riippuvaisia yksittäisen testaajan taidoista, kokemuksista ja mieltymyksistä. Sellaisenaan näillä lähestymistavoilla harvoin saavutetaan riskipohjaisen testauksen täysiä etuja.

Yrittäessään saavuttaa riskipohjaisen testauksen hyödyt mutta samalla minimoida kustannukset monet ammattilaiset käyttävät kevyitä riskipohjaisia lähestymistapoja. Nämä yhdistävät vapaamuotoisten lähestymistapojen nopean vasteajan ja joustavuuden muodollisempien lähestymistapojen tehokkuuteen ja yksimielisyyden saavuttamiseen. Esimerkkeihin kevyemmistä lähestymistavoista kuuluvat Pragmatic Risk Analysis and Management (PRAM) [Black09], Systematic Software Testing (SST) [Craig02] ja Product Risk Management (PRisMa) [vanVeenendaal12]. Tavanomaisten riskipohjaisen testauksen ominaisuuksien lisäksi näillä tekniikoilla on tyypillisesti seuraavat ominaisuudet:

- Ne ovat kehittyneet ajan myötä riskipohjaisesta testauksesta eri teollisuudenaloilla saatujen kokemusten perusteella, erityisesti tehokkuutta korostavien alojen kokemuksien pohjalta.
- Perustuvat sekä liiketoimintaa että teknisiä näkökulmia edustavista sidosryhmien edustajista kootun tiimin laajaan mukanaoloon erityisesti riskien tunnistamisessa ja arvioinnissa.
- Ovat parhaimmillaan, kun ne otetaan käyttöön projektin aikaisimmissa vaiheissa, kun laaturiskien minimoinnin vaihtoehdot ovat laajimmillaan, ja kun riskianalyysin päätuotokset ja sivutuotteet voivat auttaa vaikuttamaan tuotteen määrittelyyn ja toteutukseen tavalla, joka minimoi riskit.
- Käyttävät generoituja lopputuloksia (riskimatriisia tai riskianalyysitaulukkoa) testaussuunnittelun ja testattavien tilanteiden ja sitä kautta kaikkien testauksen hallinnan ja analysoinnin tehtävien pohjana.
- Tukevat testitulosten raportointia jäljellä olevien riskien osalta kaikille testauksen sidosryhmätasoille.

Jotkut näistä tekniikoista (esim. SST) vaativat vaatimusmäärittelyitä riskianalyysin syötteeksi ja niitä voi käyttää vain silloin, kun vaatimusmäärittelyt ovat saatavilla. Toiset tekniikat (esim. PRisMA ja PRAM) kannustavat yhdistetyn riski- ja vaatimuspohjaisen strategian käyttöön, jossa vaatimuksia ja/tai muita määrittelyitä käytetään riskianalyysin syötteenä, mutta ne voivat toimia myös pelkästään sidosryhmien syötteen perusteella. Vaatimusten käyttö syötteenä auttaa varmistamaan hyvän vaatimuskattavuuden, mutta Testauspäällikön pitää varmistaa, että tärkeitä riskejä, jotka eivät tule esiin vaatimusten kautta – erityisesti ei-toiminnallisilla alueilla – ei jää huomioimatta. Kun hyviä, priorisoituja vaatimuksia on käytettävissä syötteiksi, nähdään tyypillisesti vahva korrelaatio riskitasojen ja vaatimusten prioriteetin välillä.

Monet näistä tekniikoista rohkaisevat myös käyttämään riskien tunnistamis- ja arviointiprosessia keinona sidosryhmien välisen yksimielisyyden luomiseen testauksen lähestymistapoihin liittyen. Tämä on iso hyöty, mutta edellyttää, että sidosryhmät käyttävät aikaa osallistuakseen joko ryhmän aivoriihi-istuntoihin tai henkilökohtaisiin haastatteluihin. Riittämätön sidosryhmien osallistuminen johtaa puutteisiin riskianalyysissä. Sidosryhmät eivät tietenkään aina ole yhtä mieltä riskin tasosta, joten laaturiskianalyysiä johtavan henkilön täytyy työskennellä luovasti ja positiivisesti sidosryhmien kanssa parhaan mahdollisen yhteisymmärryksen aikaansaamiseksi. Kaikki katselmointipalavereita vetävän koulutetun puheenjohtajan taidot sopivat myös laaturiskianalyysiä johtavalle henkilölle.

Kuten muodollisemmat tekniikat, kevyemmätkin tekniikat sallivat todennäköisyyden ja vaikutuksen painottamisen liiketoiminta- tai teknisten riskien korostamiseksi (esimerkiksi riippuen testauksen tasosta). Toisin kuin muodollisemmat tekniikat, kevyemmät tekniikat kuitenkin

- käyttävät vain kahta tekijää, todennäköisyyttä ja vaikutusta
- käyttävät yksinkertaisia, laadullisia arvioita ja asteikkoja.

Näiden lähestymistapojen keveys tuo joustavuutta ja mahdollistaa soveltuvuuden monille eri toimialoille sekä läpi tiimien, joiden kokemus ja osaaminen voivat olla kaikentasoisia (mukana voi olla jopa ei-teknisiä ja aloittelevia henkilöitä).

Asteikon muodollisessa, raskaammassa päässä Testauspäälliköllä on käytettävissään joukko vaihtoehtoja:

- Vaara-analyysi, joka laajentaa analyyttistä prosessia ylöspäin ja yrittää tunnistaa jokaisen riskin taustalla olevia vaaratilanteita.
- Riskien toteutumisen kustannus, jossa riskiarviointiprosessi keskittyy määrittämään jokaisen laaturiskin osalta kolme asiaa: 1) joka riskiaiheeseen liittyvä häiriön todennäköisyys (ilmaistuna prosentteina); 2) riskiaiheeseen liittyvän tyypillisen häiriön aiheuttaman menetyksen kustannukset, mikäli häiriö tapahtuu tuotantokäytössä (ilmaistuna rahallisena määränä); ja 3) tällaisten häiriöiden testaamisen kustannukset.
- Vikavaikutusanalyysi (Failure Mode and Effect Analysis, FMEA) ja sen muunnelmat [Stamatis03], jossa tunnistetaan laaturiskit, niiden mahdolliset syyt ja todennäköiset vaikutukset, ja sen jälkeen niille annetaan vakavuus-, kiireellisyys- ja löytöarviot.
- QFC (Quality Function Deployment), joka on laaturiskien hallintamenetelmä, joka keskittyy väärästä tai riittämättömästä asiakkaan tai käyttäjän vaatimusten ymmärtämisestä syntyviin laaturiskeihin ja erityisesti niiden testaamiseen.
- Vikapuuanalyysi (Fault Tree Analysis, FTA), jossa erilaisiin joko testauksessa tai tuotannossa todettuihin häiriöihin tai mahdollisiin häiriöihin (laaturiskeihin) kohdistetaan perussyyanalyysi, joka lähtee liikkeelle vioista, jotka voivat aiheuttaa häiriön, etenee sitten virheisiin tai vikoihin, jotka voivat aiheuttaa aiemmin tutkitut viat, ja jatkuu eteenpäin, kunnes erilaiset alkuperäissyyt on tunnistettu.

Riskipohjaisessa testauksessa käytettävät yksittäiset tekniikat ja kunkin tekniikan muodollisuuden taso riippuvat projektin, prosessin ja tuotteen kannalta huomioon otettavista asioista. Esimerkiksi vapaamuotoinen lähestymistapa, kuten Whittakerin tutkiva tekniikka voi sopia korjauspakettiin tai pikakorjaukseen. Ketterissä projekteissa laaturiskianalyysi integroidaan täysin joka sprintin alkuvaiheeseen ja riskien dokumentointi sulautuu käyttäjätarinan seurantaan. Järjestelmien järjestelmät vaativat erikseen jokaisen järjestelmän riskianalyysin sekä koko järjestelmäkokonaisuuden riskianalyysin. Turvallisuuskriittiset ja tehtäväkriittiset projektit vaativat tiukempaa muodollisuutta ja laajempaa dokumentaatiota.

Valitut tekniikat määräävät yleensä riskipohjaisen testauksen syötteet, prosessit ja tulokset. Tyypillisiin syötteisiin kuuluvat sidosryhmien näkemykset, määritykset ja historiatieto. Tyypillisiin prosesseihin kuuluvat riskien tunnistaminen, riskien arviointi ja riskien hallinta. Tyypillisiin tuloksiin kuuluvat lista laaturiskeistä (sisältää myös riskitasot sekä suosituksen käytettävästä testauksen työmäärästä), syötedokumenteista, kuten määrityksistä, löydetyistä vioista, riskiaiheisiin liittyvistä kysymyksistä tai ongelmista sekä testaukseen tai koko projektiin vaikuttavista projektiriskeistä.

Yleensä riskipohjaisen testauksen kaikkein tärkein menestystekijä on oikeista sidosryhmistä muodostetun tiimin osallistuminen riskien tunnistamiseen ja arviointiin. Kaikilla sidosryhmillä on oma näkemyksensä siitä, mistä tuotteen laatu muodostuu, sekä omat prioriteettinsa ja huolensa laatuun liittyen. Sidosryhmät jakaantuvat kuitenkin yleensä kahteen isompaan ryhmään: liiketoiminnan sidosryhmät ja tekniset sidosryhmät.

Liiketoiminnan sidosryhmiin kuuluvat muun muassa asiakkaat, käyttäjät, käyttöhenkilöstö, help desk ja teknisen tuen henkilöstö. Nämä sidosryhmät ymmärtävät asiakasta ja käyttäjää, ja pystyvät näin tunnistamaan riskejä ja arvioimaan niiden vaikutusta liiketoiminnan näkökulmasta.

Teknisiin sidosryhmiin kuuluvat muun muassa kehittäjät, suunnittelijat, tietokannanvalvojat ja tietoverkonhoitajat. Nämä sidosryhmät ymmärtävät taustalla olevat tavat, jotka voivat aiheuttaa ohjelmistohäiriön, ja voivat näin tunnistaa riskejä ja arvioida niiden todennäköisyyttä teknisestä näkökulmasta.

Joillakin sidosryhmillä on sekä liiketoiminnan että tekninen näkökulma. Esimerkiksi asiantuntijoilla, jotka toimivat testaus- tai liiketoiminta-asiantuntijoiden rooleissa, on usein laajempi näkemys riskeistä heidän teknisen ja liiketoiminta-asiantuntemuksensa vuoksi.

Riskiaiheiden tunnistamisprosessissa syntyy melkoinen riskien lista. Sidosryhmien ei tarvitse kinastella riskiaiheista; niin kauan, kun jokin sidosryhmä kokee jonkin asian järjestelmän laatuun kohdistuvana riskinä, se on riskiaihe. On kuitenkin tärkeää, että sidosryhmät pääsevät yksimielisyyteen riskitason arviosta. Esimerkiksi kevyemmässä lähestymistavassa, joka käyttää arviointikohteina todennäköisyyttä ja vaikutusta, osana prosessia on oltava yhteisen arviointimallin löytäminen sekä todennäköisyydelle että vaikutukselle. Kaikkien sidosryhmien, testausryhmä mukaan luettuna, on käytettävä samaa asteikkoa ja niiden on pystyttävä sopimaan yhdestä todennäköisyyden sekä vaikutuksen arvosta kunkin laaturiskiaiheen kohdalla.

Jos riskipohjaista testausta aiotaan käyttää pidemmän aikaa, Testauspäällikön on vietävä riskipohjaista testausta eteenpäin ja käynnistettävä se onnistuneesti sidosryhmien parissa. Organisaation rajat ylittävän ryhmän on ymmärrettävä riskianalyysin arvo, jotta varmistetaan tekniikan jatkuva käyttö. Tämä edellyttää, että Testauspäällikkö ymmärtää sidosryhmien tarpeet ja odotukset sekä prosessiin osallistumiseen käytettävissä olevan ajan.

Hyvä sidosryhmien osallistuminen laaturiskien analysointiprosessiin tarjoaa Testauspäällikölle tärkeän hyödyn. Niissä projekteissa, joiden määritykset ovat puutteelliset ja vaatimukset heikkoja tai niitä ei ole, sidosryhmät voivat silti tunnistaa riskejä sopivan tarkistuslistan ohjaamana. Tämä hyöty voidaan todeta, kun vikojen löytötehokkuus testaustiimissä paranee riskipohjaisen testauksen käyttöönoton jälkeen. Näin tapahtuu, koska käytetään kattavampaa testauksen pohjamateriaalia – tässä tapauksessa laaturiskiaiheiden listaa.

Riskipohjaisen testauksen päätöstehtävien aikana testaustiimien pitäisi mitata, kuinka laajasti he kokivat näiden hyötyjen toteutuneen. Monissa tapauksissa tähän liittyy vastaaminen joihinkin tai kaikkiin seuraavista neljästä kysymyksestä käyttämällä mittareita ja keskustelemalla testaustiimin kanssa:

- Löysikö testaustiimi prosentuaalisesti enemmän merkittäviä vikoja kuin vähemmän tärkeitä vikoja?
- Löysikö testaustiimi suurimman osan tärkeistä vioista aikaisessa testauksen suorituksen vaiheessa?
- Pystyikö testaustiimi selittämään testitulokset sidosryhmille riskeihin suhteutettuna?
- Liittyikö testeihin, jotka testaustiimi jätti väliin (jos sellaisia oli) matalamman tason riski kuin niihin, jotka suoritettiin?

Useimmissa tapauksissa onnistunut riskipohjainen testaus tuottaa myöntävän vastauksen kaikkiin neljään kysymykseen. Pidemmällä tähtäimellä Testauspäällikön pitäisi asettaa näille neljälle mittarille prosessinkehitystavoitteet ja pyrkiä parantamaan laaturiskianalyysiprosessin tehokkuutta. Tietenkin riskipohjaiseen testaukseen voidaan soveltaa muita metriikoita ja onnistumiskriteereitä, ja Testauspäällikön pitää huolellisesti miettiä näiden metriikoiden välisiä suhteita, strategisia tavoitteita, joita testaustiimi palvelee, sekä tiettyihin metriikoihin ja onnistumiskriteereihin pohjautuvasta johtamisesta seuraavaa käyttäytymistä.

### 2.3.3 Muita testien valinnan tekniikoita

Vaikka monet Testauspäälliköt käyttävät riskipohjaista testausta yhtenä testausstrategian osana, monet käyttävät myös muita tekniikoita.

Yksi eniten näkyvillä olevista vaihtoehtoisista tekniikoista testattavien tilanteiden laatimiseen ja priorisointiin on vaatimuspohjainen testaus. Vaatimuspohjainen testaus voi hyödyntää useita tekniikoita, kuten monitulkintaisuuskatselmointeja, testattavien tilanteiden analysointia sekä syy-seuraus-kaavioita.

Monitulkintaisuuskatselmoinneissa tunnistetaan ja poistetaan moniselitteisyyksiä vaatimuksista (jotka toimivat testauksen pohjamateriaalina) usein käyttämällä tyypillisistä vaatimusvioista laadittua tarkistuslistaa (ks. [Wiegers03]).

Kuten on kuvattu [Craig02]ssa, testattavien tilanteiden analyysiin kuuluu vaatimusmäärittelyiden tarkka läpikäynti katettavien testattavien tilanteiden tunnistamiseksi. Jos näille vaatimuksille on määritelty prioriteetti, sitä voidaan käyttää testitapausten työmäärän arvioinnissa ja priorisoinnissa. Prioriteettitiedon puuttuessa on vaikea määritellä sopiva työmäärä ja testausjärjestys sekoittamatta vaatimuspohjaista testausta riskipohjaiseen testaukseen.

Syy-seuraus-kaavioita käsitellään Jatkotason sertifikaattisisällön Testausasiantuntija-osiossa, kun käsitellään testattavien tilanteiden yhdistelmien kattamista osana testien analysointia. Sillä on kuitenkin laajempi käyttötarkoitus siinä, että se voi auttaa pienentämään äärettömän laajan testausongelman hallittavissa olevaksi testitapausten joukoksi ja tuottaa silti testauksen pohjamateriaalin 100 %:n toiminnallisen kattavuuden. Syy-seuraus-kaaviot paljastavat myös testisuunnittelun aikana testauksen pohjamateriaalissa aukkoja, jotka voivat paljastaa vikoja aikaisessa ohjelmiston elinkaaren vaiheessa, mikäli testien suunnittelu aloitetaan vaatimusluonnoksia vasten. Yksi iso este syy-seuraus-kaavioiden käytölle on niiden luomisen monimutkaisuus. On olemassa työkaluja, jotka tukevat tämän menetelmän käyttöä, joka manuaalisesti voi olla monimutkaista.

Yleinen vaatimuspohjaisen testauksen este on, että usein vaatimusmäärittelyt ovat moniselitteisiä, ei-testattavia, epätäydellisiä tai niitä ei ole. Kaikki organisaatiot eivät ole halukkaita ratkaisemaan näitä ongelmia, joten tällaisiin tilanteisiin joutuvien testaajien on valittava toinen testausstrategia.

Toinen menetelmä, jota joskus käytetään laajentamaan olemassa olevien vaatimusten käyttöä, on käyttö- tai toiminnallisten profiilien luominen. Se on mallipohjainen lähestymistapa, joka käyttää käyttötapausten, käyttäjien (joita kutsutaan usein persooniksi), syötteiden ja tulosten sekoitusta kuvaamaan tarkasti järjestelmän käyttöä todellisessa maailmassa. Tämän avulla voidaan testata toiminnallisuuden lisäksi myös käytettävyyttä, yhteentoimivuutta, luotettavuutta, tietoturvaa ja suorituskykyä.

Testien analyysin ja suunnittelun aikana testaustiimi tunnistaa käyttäjäprofiilit ja yrittää kattaa ne testitapauksilla. Käyttäjäprofiili on arvio, joka perustuu saatavilla olevaan tietoon tai ohjelmiston todenmukaiseen käyttöön. Se tarkoittaa, että riskipohjaisessa testauksessa käyttäjäprofiilit eivät ehkä täydellisesti mallinna lopullista todellisuutta. Jos kuitenkin saatavilla on riittävästi tietoa ja panostusta sidosryhmiltä, mallista saadaan riittävä (ks. [Musa04]).

Jotkut Testauspäälliköt käyttävät myös menetelmällisiä lähestymistapoja kuten tarkistuslistoja sen määrittämiseen, mitä testataan, kuinka paljon ja missä järjestyksessä. Hyvin vakaiden tuotteiden kohdalla merkittävimmät toiminnallisuudet ja ei-toiminnalliset testattavat alueet sisältävä tarkistuslista yhdessä olemassa olevien testitapausten kanssa voi olla riittävä. Tarkistuslista tarjoaa testien työmäärän ja järjestyksen suhteen heuristiikkoja, jotka pohjautuvat yleensä tehtyjen muutosten tyyppiin ja määrään. Tällaiset lähestymistavat muuttuvat yleensä vähemmän käyttökelpoisiksi, kun testataan muita kuin vain vähäisiä muutoksia.

Lopuksi, eräs tavallinen menetelmä on käyttää reaktiivista lähestymistapaa. Reaktiivisessa lähestymistavassa vain vähän testien analysointiin, suunnitteluun tai valmisteluun liittyviä tehtäviä tapahtuu ennen testien suoritusta. Testaustiimi keskittyy reagoimaan oikeasti toimitettavaan tuotteeseen. Kun löydetään vikaryppäitä, jatkotestaus keskittyy niihin. Priorisointi ja allokointi ovat täysin dynaamisia. Reaktiivinen lähestymistapa voi toimia muiden lähestymistapojen täydentäjänä, mutta jos sitä käytetään yksistään, sovelluksen keskeiset ja tärkeät alueet, joissa ei kuitenkaan ole paljon vikoja, jäävät usein huomiotta.

### 2.3.4 Testien priorisointi ja työmäärän jakaminen testausprosessissa

Mitä hyvänsä tekniikkaa – tai, vielä parempi, tekniikoiden sekoitusta – Testauspäällikkö käyttääkin, hänen on sovitettava kyseinen tekniikka projektin ja testauksen prosesseihin. Esimerkiksi peräkkäiselinkaarimalleissa (esim. V-malli) testaustiimi valitsee testit, määrittää testauksen työmäärän ja tekee ensimmäisen testien priorisoinnin vaatimusvaiheessa, ja näitä muokataan tietyin väliajoin, kun taas iteratiiviset tai Ketterät elinkaaret vaativat iteraatio-iteraatiolta-lähestymistapaa. Testauksen suunnittelussa

ja hallinnassa on mietittävä, missä määrin riskit, vaatimukset ja/tai käyttöprofiilit kehittyvät, ja niihin on reagoitava vastaavasti.

Testien analysoinnin, suunnittelun ja toteutuksen aikana on käytettävä testauksen suunnitteluvaiheessa määriteltyjä työmääriä ja priorisointia. On yleistä, että testausprosessia jaetaan pienempiin osiin tarkkaa analysointia ja/tai mallintamista varten, mutta näitä tietoja ei saa käyttää ohjaamaan testausprosessin etenemistä. Tämä osiin jakaminen tapahtuu tyypillisesti suunnittelun ja toteutuksen aikana.

Testien suorituksessa pitäisi noudattaa suunnittelun aikana määriteltyä priorisointia, vaikka on tärkeä päivittää prioriteetteja säännöllisesti suunnitelman laatimisen jälkeen saadun tiedon perusteella. Kun Testauspäällikkö arvioi testituloksia ja päätöskriteerien tilaa ja raportoi niistä, hänen on myös arvioitava ja raportoitava niistä riskien, vaatimuksien, käyttöprofiilien, tarkistuslistojen ja muiden testien valintaa ja priorisointia ohjanneiden seikkojen näkökulmasta. Mikäli tarpeellista, testien luokittelu kiireellisyyden mukaan pitäisi tehdä testien priorisointimallin pohjalta.

Testauspäällikkö voi osana tulosten raportointia ja päätöskriteerien arviointia mitata myös testauksen valmistumisastetta. Tähän pitäisi kuulua testitapausten ja löydettyjen vikojen jäljittäminen takaisin niihin liittyvään pohjamateriaaliin. Esimerkiksi riskipohjaisessa testauksessa kun testit suoritetaan ja vikoja löydetään, testaajat voivat tutkia jäljellä olevien riskien riskitasoa. Tämä tukee riskipohjaisen testauksen käyttöä oikean julkaisuhetken määrittämisessä. Testauksen raportoinnissa pitäisi ottaa kantaa katettuihin ja vielä avoinna oleviin riskeihin sekä saavutettuihin ja vielä saavuttamattomiin hyötyihin. Testitulosten raportoinnista riskikattavuuden perusteella löytyy esimerkki ks. [Black03].

Lopuksi, testauksen päätösvaiheen aikana Testauspäällikön on tarkasteltava mittareita ja onnistumiskriteereitä, jotka ovat oleellisia testauksen sidosryhmien tarpeiden ja odotusten kannalta, mukaan luettuna asiakkaiden ja käyttäjien laatuun liittyvät tarpeet ja odotukset. Vain silloin, kun testaus täyttää nämä tarpeet ja odotukset, voi testaustiimi väittää olevansa aidosti tehokas.

## 2.4 Testausdokumentaatio ja muut tuotokset

Dokumentaatio tuotetaan usein osana testauksen hallinnan tehtäviä. Vaikka testauksenhallinnan dokumenttien tarkat nimet ja dokumenttien laajuus vaihtelevat, seuraavassa on lueteltu organisaatioissa ja projekteissa tyypillisesti esiintyviä testauksenhallinnan dokumentteja:

- Testauspolitiikka – kuvaa organisaation testaukselle asettamat pyrkimykset ja tavoitteet
- Testausstrategia – kuvaa organisaation yleiset, projekteista riippumattomat testaustavat
- Kokonaistestaussuunnitelma (tai Projektin testaussuunnitelma) – kuvaa, kuinka testausstrategia pannaan käytäntöön tietyssä projektissa
- Tasokohtainen testaussuunnitelma – kuvaa tehtävät, jotka suoritetaan nimenomaan kyseisellä testaustasolla.

Näiden dokumenttien konkreettinen toteutustapa voi vaihdella tilanteesta toiseen. Joissakin organisaatioissa ja joissakin projekteissa ne voidaan yhdistää yhdeksi ainoaksi dokumentiksi, toisissa ne voivat olla erillisiä dokumentteja ja joissain niiden sisältö voi tulla esiin intuitiivisesti, kirjoittamattomana tai testauksen perinteisten menetelmien käyttönä. Isommissa ja muodollisemmissa organisaatioissa ja projekteissa kaikki nämä dokumentit löytyvät yleensä kirjallisina tuotoksina, kun taas pienemmissä ja vähemmän muodollisissa organisaatioissa ja projekteissa on yleensä vähemmän kirjallista dokumentaatiota. Tämä sertifikaattisisältö kuvaa jokaisen näistä dokumenteista erikseen, vaikka käytännössä organisaation ja projektin sisältö määrittää kunkin dokumentin oikean käytön.

### 2.4.1 Testauspolitiikka

Testauspolitiikka kuvaa, miksi organisaatiossa testataan. Se määrittää yleiset testauksen tavoitteet, jotka organisaatio haluaa saavuttaa. Tämän politiikan laatimiseen pitäisi osallistua organisaation ylemmän testausjohdon yhdessä testauksen sidosryhmien ylempien johtajien kanssa.

Joissain tapauksissa testauspolitiikka voi täydentää tai olla osa laajempaa laatupolitiikkaa. Tämä laatupolitiikka kuvaa johdon yleiset laatuun liittyvät arvot ja tavoitteet.

Silloin, kun kirjallinen testauspolitiikka on olemassa, sen pitäisi olla lyhyt, ylemmän tason dokumentti, joka

- vetää yhteen organisaation testauksesta saaman hyödyn
- määrittää testauksen tavoitteet, kuten luottamuksen saaminen ohjelmistoon, vikojen löytäminen ohjelmistosta ja laaturiskien riskitason vähentäminen (ks. kappale 2.3.1)
- kuvaa, kuinka voidaan arvioita testauksen yleistä ja toiminnallista tehokkuutta näiden tavoitteiden saavuttamisessa
- kuvaa tyypillisen testausprosessin, käyttäen ehkä perustana ISTQB:n perustestausprosessia
- määrittää, kuinka organisaatio pyrkii parantamaan testausprosessejaan (ks. luku 5).

Testauspolitiikan pitäisi käsitellä sekä uuskehitykseen että ylläpitoon liittyviä testaustehtäviä. Se voi myös viitata sisäisiin ja ulkoisiin standardeihin koko organisaatiota koskevien testauksen tuotosten ja terminologian osalta.

### 2.4.2 Testausstrategia

Testausstrategia kuvaa organisaation yleisen testausmetodologian. Tähän kuuluvat tapa, jolla testausta käytetään tuote- ja projektiriskien hallintaan, kuinka testaus on jaettu tasoihin, ja testaukseen liittyvät ylemmän tason tehtävät. (Organisaatiolla voi olla erilaisia strategioita eri tilanteisiin, kuten erilaisiin ohjelmistokehityksen elinkaariin tai erilaisia riskitasoja tai säädöksiin pohjautuvia vaatimuksia varten.) Testausstrategian ja siinä kuvattujen prosessien ja tehtävien pitää olla yhdenmukaisia testauspolitiikan kanssa. Sen pitää kuvata organisaation tai yhden tai useamman hankkeen yleiset testauksen aloitus- ja lopetuskriteerit.

Kuten yllä on mainittu, testausstrategiat kuvaavat yleiset testausmenetelmät, joihin tyypillisesti kuuluvat

- analyyttiset strategiat, kuten riskipohjainen testaus, jossa testaustiimi analysoi testauksen pohjamateriaalin tunnistaakseen katettavat testattavat tilanteet. Esimerkiksi vaatimuspohjaisessa testauksessa testattavat tilanteet johdetaan testianalyysissa vaatimuksista ja sen jälkeen suunnitellaan ja toteutetaan testit, joilla kyseiset tilanteet katetaan. Tämän jälkeen testit suoritetaan käyttämällä usein testien kattamien vaatimusten tärkeysjärjestystä määrittämään, missä järjestyksessä testit ajetaan. Testitulokset raportoidaan vaatimusten tilan mukaan, esimerkiksi testatut ja hyväksytyt vaatimukset, testatut ja hylätyt vaatimukset, ei vielä kokonaan testatut vaatimukset, vaatimukset, joiden testaus on estynyt, jne.
- mallipohjaiset strategiat, kuten käyttöprofiilit, jolloin testaustiimi laatii (todellisten tai ennakoitujen tilanteiden pohjalta) mallin ympäristöstä, jossa järjestelmä tulee toimimaan, syötteistä ja tilanteista, joita järjestelmä tulee kohtaamaan, sekä siitä, kuinka järjestelmän pitäisi käyttäytyä. Esimerkiksi nopeasti kasvavan mobiilisovelluksen mallipohjaisessa suorituskykytestauksessa voidaan kehittää mallit saapuvasta ja lähtevästä verkkoliikenteestä, aktiivisista ja passiivisista käyttäjistä sekä syntyvästä käsittelykuormasta senhetkisen käytön pohjalta ja ennakoida ajan myötä tapahtuvaa kasvua. Lisäksi malleja voidaan kehittää nykyisen tuotantoympäristön laitteiston, ohjelmiston, aineistokapasiteetin, verkkoympäristön ja infrastruktuurin perusteella. Malleja voidaan kehittää myös kuvaamaan ihanteellista, odotettua ja minimiläpäisynopeutta, vasteaikoja ja resurssien käyttöä.
- menetelmälliset, kuten laatuominaisuuksiin pohjautuvat, strategiat, joissa testaustiimi käyttää ennalta määrättyä testattavien tilanteiden joukkoa, kuten esimerkiksi laatustandardia (esim. ISO 25000 [ISO25000], joka korvaa ISO9126:n [ISO9126]) tai yleistettyä, loogisia tiettyyn toimialaan, sovellukseen tai testaustyyppiin (esim. tietoturvatestaus) kohdistuvia testattavien tilanteiden joukkoa kuvaavaa tarkistuslistaa tai kokoelmaa, ja käyttää sitä iteraatiosta tai julkaisusta toiseen. Esimerkiksi yksinkertaisen, vakaan verkkokauppasivuston ylläpitotestauksessa testaajat voivat käyttää tarkistuslistaa, johon on koottu avaintoiminnot, ominaisuudet ja linkit joka sivulle. Testaajat käyvät läpi tarkistuslistan keskeiset elementit joka kerta, kun sivustoon tehdään muutoksia.
- Prosesseja tai standardeja noudattavat strategiat, kuten lääketieteelliset järjestelmät, joiden on noudatettava Yhdysvaltain elintarvike- ja lääkeviraston (FDA) standardeja, jolloin testaustiimi noudattaa standardointikomitean tai muun asiantuntijaraadin laatimia prosesseja, jotka käsittelevät dokumentaatiota, testauksen pohjamateriaalin ja testioraakkeleiden oikeaa tunnistamista ja käyttöä, sekä testaustiimin organisaatiota. Esimerkiksi projekteissa, jotka noudattavat kette-

rien menetelmien Scrum-tekniikkaa, joka iteraation aikana testaajat analysoivat käyttäjätarinoita, jotka kuvaavat eri ominaisuuksia, arvioivat jokaisen ominaisuuden vaatiman testaustyömäärän osana koko iteraation suunnitteluprosessia, tunnistavat jokaisen käyttäjätarinan testattavat tilanteet (kutsutaan usein hyväksymiskriteereiksi), suorittavat testit, joilla kyseiset tilanteet katetaan, ja raportoivat joka käyttäjätarinan tilasta (ei testattu, hylätty, läpäisty) testin suorituksen aikana.

- Reaktiiviset strategiat, kuten vikapohjaisten hyökkäysten käyttäminen, jolloin testaustiimi odottaa testien suunnittelussa ja toteutuksessa siihen asti, kunnes ohjelmisto on toimitettu heille, ja pohjaa toimintansa vasta varsinaiseen testattavaan järjestelmään. Esimerkiksi, kun käytetään tutkivaa testausta valikkopohjaisen sovelluksen testauksessa, voidaan laatia joukko testiohjeita, jotka pohjautuvat sovelluksen ominaisuuksiin, valikkovalintoihin ja näyttöihin. Jokaiselle testaajalle annetaan joukko testiohjeita, joita he sitten käyttävät rakentaessaan tutkivan testauksen istuntonsa. Testaajat raportoivat säännöllisesti testaussessioiden tuloksista Testauspäällikölle, joka voi muokata testiohjeita löydösten perusteella.
- Konsultatiiviset strategiat, kuten esimerkiksi käyttäjien ohjaama testaus, missä testaustiimi pohjaa työnsä yhden tai useamman avainsidosryhmän palautteeseen määritellessään testattavia kohteita, jotka pitää kattaa. Esimerkiksi web-pohjaisen sovelluksen ulkoistetussa yhteensopivuustestauksessa yritys voi antaa ulkoistetulle testauspalvelun tuottajalle priorisoidun listan selainversioista, haittaohjelmien torjuntaohjelmista, käyttöjärjestelmistä, yhteystyypeistä ja muista kokoonpanovaihtoehdoista jotka he haluavat arvioitavan sovellustaan vasten. Testauspalvelun tuottaja voi sitten käyttää eri tekniikoita kuten syötteiden pareittainen testaus (korkean prioriteetin vaihtoehdoille) ja ekvivalenssiositus (alemman prioriteetin vaihtoehdoille) testien luomiseen.
- Regressiota välttävät testausstrategiat, kuten laaja-alainen automaatio, jossa testaustiimi käyttää taantumisriskin hallitsemiseen useita tekniikoita, erityisesti toiminnallisen ja/tai ei-toiminnallisen regressiotestauksen automatisointia yhdellä tai useammalla tasolla. Esimerkiksi jos regressiotestataan web-pohjaista sovellusta, testaajat voivat käyttää käyttöliittymäpohjaista automaatiotyökalua sovelluksen tyypillisten ja poikkeustilanteiden käyttötapausten automatisointiin. Nämä testit suoritetaan sitten aina, kun sovellusta muokataan.

Eri strategioita voidaan yhdistää. Valittujen strategioiden pitäisi sopia organisaation tarpeisiin ja tapoihin, ja organisaatiot voivat räätälöidä strategioita sopimaan tiettyihin tilanteisiin ja projekteihin.

Testausstrategia voi kuvata käytettävät testaustasot. Tällaisissa tilanteissa sen pitäisi ohjata jokaisen tason aloitus- ja lopetuskriteereitä sekä eri tasojen välisiä suhteita (esim. testikattavuuden tavoitteiden jakautuminen).

Testausstrategia voi kuvata myös seuraavat seikat:

- integrointitoimenpiteet
- testien määrittelytekniikat
- testauksen riippumattomuus (joka voi vaihdella testaustasojen mukaan)
- pakolliset ja valinnaiset standardit
- testiympäristö
- testiautomaatio
- testaustyökalut
- ohjelmistotuotteiden ja testauksen tuotosten uudelleenkäytettävyys
- varmistustestaus (uudelleentestaus) ja regressiotestaus
- testauksen valvonta ja raportointi
- testauksen mittaaminen ja metriikat
- havaintojenhallinta
- kokoonpanonhallinnan suhde testimateriaaliin
- roolit ja vastuut.

Voi olla tarpeen laatia erikseen lyhyen ja pitkän tähtäimen testausstrategiat. Eri testausstrategiat sopivat eri organisaatioihin ja projekteihin. Kun kyse on esimerkiksi tietoturva- tai turvallisuuskriittisistä sovelluksista, perusteellisempi strategia voi olla soveliaampi kuin muissa tilanteissa. Testausstrategia vaihtelee myös eri kehitysmallien mukaan.

### 2.4.3 Kokonaistestaussuunnitelma

Kokonaistestaussuunnitelma kattaa kaiken tietyssä projektissa tehtävän testaustyön, mukaan luettuna suoritettavat eri testaustasot ja suhteet niiden välillä sekä testaustasojen ja vastaavien kehitystehtävien väliset suhteet. Kokonaistestaussuunnitelman pitää kuvata, kuinka testaajat toteuttavat testausstrategian (eli testauksen lähestymistavat) kyseisessä projektissa. Kokonaistestaussuunnitelman pitää olla yhdenmukainen testauspolitiikan ja -strategian kanssa, ja niissä määrätyissä kohdissa, joissa se ei sitä ole, on selitettävä nämä poikkeamat ja poikkeukset, mukaan luettuna kaikki mahdolliset poikkeamien aiheuttamat vaikutukset. Jos esimerkiksi organisaation testausstrategian mukaan suoritetaan muuttumattomalle järjestelmälle yksi kokonainen regressiotestauskierros ennen julkaisua, mutta kyseisessä projektissa ei tehdä regressiotestausta, testaussuunnitelmassa pitää selittää, miksi näin on suunniteltu ja mitä aiotaan tehdä tämän tavallisesta strategiasta tehdyn poikkeaman aiheuttamien mahdollisten riskien pienentämiseksi. Testaussuunnitelman pitää myös sisältää selitykset kaikille muille vaikutuksille, joita tämän poikkeaman odotetaan aiheuttavan. Esimerkiksi regressiotestauksen väliin jättäminen voi vaatia ylläpitojulkaisun toteuttamista kuukausi ensimmäisen projektin ohjelmistojulkaisun jälkeen. Kokonaistestaussuunnitelman pitäisi täydentää projektisuunnitelmaa tai toimintaohjeita siten, että sen pitäisi kuvata testauspanos, joka on osa isompaa projektia tai tehtäväkokonaisuutta.

Vaikka kokonaistestaussuunnitelman yksityiskohtainen sisältö ja rakenne vaihtelevatkin organisaation, sen dokumentointistandardien ja projektin muodollisuuden mukaan, tyypillisiin kokonaistestaussuunnitelman aiheisiin kuuluvat:

- Testattavat kohteet sekä kohteet, joita ei testata
- Testattavat laatuominaisuudet sekä laatuominaisuudet, joita ei testata
- Testausaikataulu ja budjetti (joka pitää sovittaa yhteen projektin tai toiminnallisen budjetin kanssa)
- Testien suorituskierrokset ja niiden suhde ohjelmiston julkaisusuunnitelmaan
- Testauksen ja muiden henkilöiden tai osastojen väliset suhteet ja tuotokset
- Sen määrittely, mitkä testattavat kohteet kuuluvat kunkin kuvatun testaustason piiriin ja mitkä ovat sen ulkopuolella
- Jokaisen testaustason aloituskriteerit, jatkamis- (keskeytys/jatkaminen) kriteerit ja päätöskriteerit sekä eri tasojen väliset suhteet
- Testausprojektin riskit
- Testaustyön kokonaishallinta
- Kunkin testaustason suoritusvastuut
- Kunkin testaustason syötteet ja tulokset.

Pienemmissä projekteissa tai tehtäväkokonaisuuksia, joissa vain yksi testaustaso on muodollinen, kokonaistestaussuunnitelma ja kyseisen muodollisen testaustason suunnitelma yhdistetään usein yhdeksi dokumentiksi. Jos esimerkiksi järjestelmätestaus on ainoa muodollinen testaustaso, ja toteuttajat suorittavat yksikkötestauksen ja integraatiotestauksen sekä asiakkaat vapaamuotoisen hyväksymistestauksen osana beta-testausprosessia, järjestelmätestaussuunnitelma voi sisältää tässä kappaleessa mainitut asiat.

Testaus on lisäksi tyypillisesti riippuvainen muista projektin tehtävistä. Jos näitä tehtäviä ei dokumentoida riittävästi erityisesti sen suhteen, miten ne vaikuttavat testaukseen ja mikä niiden suhde siihen on, näihin tehtäviin liittyvät asiat voidaan käsitellä kokonaistestaussuunnitelmassa (tai sopivassa tasokohtaisessa testaussuunnitelmassa). Esimerkiksi jos kokoonpanonhallinnan prosessia ei ole dokumentoitu, testaussuunnitelmassa pitäisi määritellä, kuinka testattavat kohteet toimitetaan testaustiimille.

### 2.4.4 Tasokohtainen testaussuunnitelma

Tasokohtaiset testaussuunnitelmat kuvaavat määrätyt tehtävät, jotka suoritetaan kullakin testaustasolla tai, joissain tapauksissa, tiettyä testaustyyppiä käytettäessä. Tasokohtaiset testaussuunnitelmat kuvaavat tarvittaessa yksityiskohtaisemmin kokonaistestaussuunnitelmassa kuvattua testaustasoa tai testaustyyppiä. Ne sisältävät aikataulun, tehtävien ja tarkistuspisteiden yksityiskohdat, joita ei välttämättä ole käsitelty kokonaistestaussuunnitelmassa. Tämän lisäksi, mikäli erilaiset standardit ja mallipohjat ohjaavat testien määrittelyä eri tasoilla, niiden yksityiskohdat käsitellään tasokohtaisissa testaussuunnitelmissa siltä osin kuin on tarpeen.

Vähemmän muodollisissa projekteissa tai tehtäväkokonaisuuksissa yksittäinen testaussuunnitelma on usein ainoa testauksen hallinnan dokumentti, joka kirjoitetaan. Tällaisissa tilanteissa jotkut aikaisemmin tässä kappaleessa mainituista tiedoista voidaan sisällyttää tähän testaussuunnitelmaan.

Ketterissä projekteissa voidaan tasokohtaisten testaussuunnitelmien sijaan käyttää sprintti- tai iteraatiokohtaisia testaussuunnitelmia.

### 2.4.5 Projektiriskien hallinta

Tärkeä osa kunnollista suunnittelua on projektiriskien käsittely. Projektiriskit voidaan tunnistaa käyttämällä samanlaisia prosesseja kuin on kuvattu kappaleessa 2.3. Kun projektiriskejä tunnistetaan, niistä pitäisi kertoa projektipäällikölle, jonka pitää ryhtyä toimeen niiden suhteen. Tällaisten riskien pienentäminen ei ole aina testausorganisaation valtuuksien rajoissa. Testauspäällikkö voi ja hänen pitäisikin kuitenkin pienentää joitakin projektiriskejä, kuten:

- Testiympäristön ja työkalujen valmius
- Testaushenkilöstön saatavuus ja osaaminen
- Testauksessa käytettävien standardien, sääntöjen ja tekniikoiden puute.

Projektiriskien hallinnan lähestymistapoihin kuuluvat testausmateriaalin aikainen laatiminen, testiympäristöjen esitestaus, tuotteen aikaisten versioiden esitestaus, tiukempien testaukseen hyväksymiskriteerien käyttäminen, vaatimusten testattavuuden edellyttäminen, varhaisten projektin tuotosten katselmointeihin osallistuminen, muutostenhallintaan osallistuminen sekä projektin edistymisen ja laadun seuranta.

Kun projektiriski on tunnistettu ja analysoitu, sen hallintaan on neljä päävaihtoehtoa:

1. Riskin pienentäminen ehkäiseviä toimenpiteitä käyttämällä sen todennäköisyyden ja/tai vaikutuksen vähentämiseksi.
2. Varasuunnitelmien laatiminen vaikutuksen vähentämiseksi riskin toteutumisen varalta.
3. Riskin siirtäminen jonkun muun osapuolen käsiteltäväksi.
4. Riski huomiotta jättäminen tai hyväksyminen.

Parhaan vaihtoehdon valitseminen riippuu vaihtoehdon tarjoamista hyödyistä ja tilaisuuksista sekä siihen liittyvistä kustannuksista ja mahdollisista lisäriskeistä. Kun projektiriskiä varten on laadittu varasuunnitelma, paras käytäntö on tunnistaa riskin laukaiseva tekijä (joka määrittää, milloin ja kuinka varasuunnitelma laitetaan käytäntöön) sekä omistaja (joka toteuttaa varasuunnitelman).

### 2.4.6 Muut testauksen tuotokset

Testauksen aikana laaditaan joukko muita tuotoksia, kuten vikaraportteja, testisuunnitelmia ja testilokeja. Suurimman osan näistä tuotoksista laativat Testausasiantuntijat ja Tekniset testausasiantuntijat. Jatkotason sertifikaattisisällön Testausasiantuntija-osiossa kuvataan seikat, jotka liittyvät näiden tuotosten laatimiseen ja dokumentointiin. Testauspäällikön pitää varmistaa näiden tuotosten yhdenmukaisuus ja laatu seuraavien tehtävien avulla:

- Tuotosten laatua, kuten hylättyjen vikaraporttien prosenttiosuutta kuvaavien metriikoiden laatiminen ja seuranta
- Yhteistyö Testausasiantuntijan ja Teknisen testausasiantuntijan kanssa sopivien mallipohjien valitsemiseksi ja muokkaamiseksi kyseisiä tuotoksia varten.
- Yhteistyö Testausasiantuntijan ja Teknisen testausasiantuntijan kanssa kyseisiä tuotoksia koskevien standardien, kuten testeissä, lokeissa ja raporteissa tarvittavan yksityiskohtaisuuden tason, määrittämiseksi.
- Testauksen tuotosten katselmoittaminen sopivia tekniikoita sekä soveltuvia osallistujia ja sidosryhmiä käyttämällä.

Testausdokumentaation laajuuteen, tyyppiin ja erityisominaisuuksiin voivat vaikuttaa muun muassa valittu ohjelmistokehityksen elinkaarimalli, sovellettavat standardit ja säädökset sekä kehitettävään järjestelmään liittyvät tuotteen laatu- ja projektiriskit.

Testauksen tuotoksille löytyy erilaisia mallipohjien lähteitä, kuten IEEE 829 [IEEE 829]. Testauspäällikön on tärkeä muistaa, että IEEE 829:n dokumentit on suunniteltu käytettäväksi millä teollisuudenalalla hyvänsä. Siksi mallipohjat sisältävät paljon yksityiskohtia jotka voivat sopia tai eivät sovellu tiettyyn organisaatioon. Paras käytäntö on räätälöidä IEEE 829:n asiakirjoja ja luoda niistä vakiomallipohjat käytettäväksi tietyssä organisaatiossa. Mallipohjien yhdenmukainen käyttäminen vähentää tarvittavaa koulutusta ja auttaa yhdenmukaistamaan prosesseja läpi organisaation.

Testaukseen kuuluu myös testitulosraporttien tuottaminen, jotka tyypillisesti laatii Testauspäällikkö ja jotka kuvataan edempänä tässä luvussa.

## 2.5 Testauksen työmääräarviointi

Arviointi johdon tehtävänä tarkoittaa likimääräisten kustannus- ja valmistumispäiväarvioiden laatimista tehtäville, jotka liittyvät määrättyyn toimintaan tai projektiin. Parhaat arviot

- edustavat kokeneiden ammattilaisten yhteistä tietämystä ja eri asianosaiset tukevat niitä
- tuottavat täsmällisiä, yksityiskohtaisia luetteloita asiaan liittyvistä kustannuksista, resursseista, tehtävistä ja ihmisistä
- esittävät jokaisen arvioidun tehtävän todennäköisimmät kustannukset, työmäärän ja keston.

Ohjelmisto- ja järjestelmätuotannon työmäärien arvioinnin on pitkään tiedetty olevan hyvin haasteellista sekä teknisesti että poliittisesti, vaikka projektijohtamisen arviointimenetelmien parhaat käytännöt ovatkin hyvin vakiintuneita. Testauksen arvioinnissa sovelletaan näitä parhaita käytäntöjä projektiin tai toimintaan liittyviin testaustehtäviin.

Testauksen arvioinnin pitäisi kattaa kaikki testausprosessiin liittyvät tehtävät, jotka on kuvattu luvussa 1. Johto on usein kiinnostunein kustannuksista, työmääristä ja erityisesti testauksen suorituksen kestosta, sillä testien suoritus on tyypillisesti projektin kriittisellä polulla. Testauksen suorituksen arviointi on kuitenkin usein vaikeaa ja epäluotettavaa, kun ohjelmiston kokonaislaatu on heikko tai sen tasoa ei tiedetä. Lisäksi järjestelmän tuttuus ja kokemus sen käytöstä vaikuttavat todennäköisesti arvioiden laatuun. Yleinen käytäntö on arvioida testauksen suorituksen aikana tarvittavien testitapausten määrä, mutta tämä toimii vain, jos voidaan olettaa, että testattavassa ohjelmistossa olevien vikojen määrä on pieni. Arvioinnin aikana tehdyt olettamukset pitää aina dokumentoida osana arviota.

Testauksen arvioinnissa pitää ottaa huomioon kaikki tekijät, jotka voivat vaikuttaa testaustehtävien kustannuksiin, työmäärään ja kestoon. Näihin tekijöihin kuuluvat seuraavat (lista ei ole tyhjentävä):

- järjestelmältä vaadittava laatutaso
- testattavan järjestelmän koko
- aikaisemmista testausprojekteista saatavilla oleva historiatieto, jota voidaan laajentaa samalta alalta saatavilla tai muiden organisaatioiden vertailukohtana käyttämillä tiedoilla
- prosessitekijät, mukaan luettuna testausstrategia, toteutuksen tai ylläpidon elinkaari ja prosessien kypsyys sekä projektiarvioiden tarkkuus
- aineelliset tekijät, mukaan luettuna testausautomaatio ja työvälineet, testiympäristö(t), testiaineisto, kehitysympäristö(t), projektidokumentaatio (esim. vaatimukset, suunnittelukuvaukset jne.) ja uudelleen käytettävät testauksen tuotokset
- ihmistekijät, mukaan lukien johtajat ja tekniset päälliköt, operatiivisen ja ylemmän tason johdon sitoutuminen ja odotukset, projektitiimin taidot, kokemus ja asenne sekä vakaus, projektitiimin suhteet, testaus- ja virheenjäljitysympäristön tuki, osaavien alihankkijoiden ja konsulttien saatavuus sekä liiketoiminnan tuntemus
- prosessin monimutkaisuus, teknologia, organisaatio, testauksen sidosryhmien määrä, alitiimien kokoonpano ja sijainti
- merkittävät toiminnan kasvattamis-, koulutus- ja perehtymistarpeet
- uusiin työkaluihin, teknologioihin, prosesseihin, tekniikoihin, räätälöityihin laitteisiin tai suureen testimateriaalimäärään sopeutuminen tai niiden kehittäminen
- vaatimukset testauksen määrittelydokumenttien korkeammasta yksityiskohtaisuuden tasosta, erityisesti kun pyritään toimimaan vieraan dokumentointistandardin mukaisesti

- monimutkainen komponenttien erityisesti integraatiotestaukseen ja testien kehittämiseen saapumisen ajoitus
- "herkkä" testiaineisto (esim. aineisto, joka on aikariippuvaista).

Testaukseen toimitetun ohjelmiston laatu on myös merkittävä tekijä, joka Testauspäälliköiden pitää ottaa huomioon arvioissaan. Jos esimerkiksi toteuttajat ovat omaksuneet parhaita käytäntöjä kuten esimerkiksi automatisoidun yksikkötestauksen ja jatkuvan integroinnin, jopa 50 % vioista tulee poistetuiksi ennen kuin koodi toimitetaan testaustiimille (lisää tietoja näiden käytäntöjen vikojenpoistotehokkuudesta, ks. [Jones11]). Jotkut ovat raportoineet, että ketterät menetelmät, kuten testiohjattu kehitys, johtavat korkeamman laadun toimittamiseen testaukseen.

Arviointi voidaan tehdä joko alhaalta ylös tai ylhäältä alas. Tekniikoihin, joita voidaan käyttää testauksen arvioinnissa joko yksinään tai yhdistelminä, kuuluvat seuraavat:

- intuitio, arvaukset ja aiemmat kokemukset
- Work Breakdown Structures (WBS)
- Testauksen arviointi-istunnot (esim. Wide Band Delphi –menetelmä)
- yrityksen standardit ja säännöt
- prosenttiosuus koko projektin työmäärästä tai henkilöstömääristä (esim. testaaja-toteuttaja-suhdeluvut)
- organisaation historia ja mittaritiedot, mukaan luettuna mittarit, jotka johdetaan malleista, jotka arvioivat vikojen, testikierrosten tai testitapausten määrää, jokaisen testaajan keskimääräistä työmäärää ja asiaan liittyvien regressiokierrosten määrää
- teollisuudenalan keskiarvot ja ennakoivat mallit kuten toimintopisteet, koodirivien määrä, arvioitu toteutuksen työmäärä tai muut projektin muuttujat (esim. ks. [Jones07]).

Useimmissa tapauksissa kun arvio on saatu valmiiksi, se täytyy toimittaa yrityksen johdolle yhdessä perusteluiden kanssa (ks. kappale 2.7). Tyypillisesti seuraa usein neuvotteluja, jotka johtavat usein arvioiden uudelleentyöstämiseen. Ihannetilanteessa lopullinen testiarviointi edustaa laadun, aikataulun, budjetin ja ominaisuuksien osalta organisaation ja projektin tavoitteiden välistä parasta mahdollista tasapainoa.

On tärkeää muistaa, että kaikki arviot perustuvat tietoon, joka on ollut saatavilla arvion tekohetkellä. Projektin aikaisessa vaiheessa tiedon määrä voi olla hyvin rajallinen. Tämän lisäksi tieto, joka on käytettävissä projektin alussa, voi muuttua ajan myötä. Jotta arviot pysyvät ajan tasalla, niitä pitää päivittää vastaamaan uutta ja muuttunutta tietoa.

## 2.6 Testauksen mittaritietojen määrittäminen ja käyttö

Johtamiseen liittyvä klisee sanoo, että se, mitä mitataan, tulee tehdyksi. Tämän lisäksi se, mitä ei mitata, jää tekemättä, koska on helppo jättää huomiotta se, mitä ei mitata. Siksi on tärkeää luoda sopiva joukko mittareita kaikkeen, mitä pyritään tekemään, testaus mukaan luettuna.

Testauksen mittarit voidaan luokitella kuuluviksi yhteen tai useampaan seuraavista ryhmistä:

- Projektimittarit, jotka mittaavat edistymistä suhteessa määriteltyihin projektin lopetuskriteereihin, kuten suoritettujen, läpäistyjen ja hylättyjen testitapausten prosenttiosuus
- Tuotemittarit, jotka mittaavat tuotteen jotain ominaisuutta, kuten esimerkiksi kuinka laajasti se on testattu tai tuotteen vikatiheys
- Prosessimittarit, jotka mittaavat testaus- tai kehitysprosessin kyvykkyyttä, kuten esimerkiksi testauksen aikana löydettyjen vikojen määrä
- Ihmismittarit, jotka mittaavat yksilöiden tai ryhmien kyvykkyyttä, kuten esimerkiksi testitapausten toteutus annetun aikataulun puitteissa.

Mikä tahansa mittari voi kuulua kahteen, kolmeen tai jopa neljään luokkaan. Esimerkiksi päivittäistä vikojen saapumisnopeutta kuvaava trendikaavio voidaan yhdistää lopetuskriteereihin (ei yhtään uusia vikoja viikon aikana), tuotteen laatuun (testaus ei pysty löytämään enää lisää vikoja tuotteessa) ja testausprosessin kyvykkyyteen (testaus löytää paljon vikoja aikaisin testauksen suorituksen alussa).

Ihmismittarit ovat erityisen arkaluontoisia. Joskus esimiehet pitävät pääasiassa prosessimittareiksi tarkoitettuja mittareita virheellisesti ihmismittareina, mikä johtaa tuhoisiin seurauksiin, kun ihmiset yrittävät vääristelemällä saada mittarit kääntymään heille suotuisaan suuntaan. Testaushenkilöiden oikeista motivointi- ja arviointitavoista keskustellaan tämän sertifikaattisisällön luvussa 7 ja Asiantuntijatason sertifikaattisisällön Testauksen hallinta osassa [ISTQB ETL SYL].

Jatkotasolla meitä kiinnostavat eniten testauksen edistymisen mittaamiseen käytettävät mittarit, eli projektimittarit. Jotkut testauksen edistymisen seurantaan käytetyistä projektimittareista liittyvät myös tuotteeseen ja prosessiin. Enemmän tietoa siitä, kuinka johto voi käyttää tuote- ja prosessimittareita, löytyy Asiantuntijatason sertifikaattisisällön Testauksen hallinta osasta. Enemmän tietoa prosessimittareiden käytöstä löytyy Asiantuntijatason sertifikaattisisällön Testauksen prosessin kehittäminen osasta [ISTQB ITP SYL].

Mittareiden käyttö suo testaajille mahdollisuuden raportoida tulokset yhdenmukaisella tavalla ja mahdollistaa johdonmukaisen edistymisen seurannan ajan myötä. Testauspäälliköitä pyydetään jatkuvasti esittämään mittareita eri kokouksissa, joihin voi osallistua sidosryhmien edustajia monelta eri tasolta teknisestä henkilöstöstä toiminnalliseen johtoon. Koska mittareita joskus käytetään projektin kokonaismenestyksen määrittämiseen, on kiinnitettävä suurta huomiota siihen, mitä seurataan, kuinka usein siitä raportoidaan ja mitä tapoja käytetään tiedon esittämiseen. Testauspäällikön on erityisesti kiinnitettävä huomiota seuraaviin seikkoihin:

- Mittareiden määrittäminen. Mittareita tulisi määrittää rajallinen joukko. Ne pitäisi määritellä projektin, prosessin ja/tai tuotteen määrättyjen tavoitteiden perusteella. Mittarit pitäisi määrittää tasapainottamaan toisiaan, sillä yksittäinen mittari voi antaa harhaanjohtavan kuvan tilasta tai suuntauksista. Kun mittarit on määritelty, niiden tulkintatavoista täytyy sopia sidosryhmien kesken, jotta vältetään sekaannukset, kun mittareista keskustellaan. On tyypillistä, että usein määritetään liian monia mittareita sen sijaan, että keskityttäisiin kaikkein oleellisimpiin.
- Mittareiden seuranta. Mittaritietojen yhdistämisen ja raportoinnin pitäisi olla niin automatisoitua kuin mahdollista, jotta vähennetään mittaritietojen keruuseen ja käsittelyyn kuluvaa aikaa. Tietyn mittarin lukemissa ajan myötä havaittavat muutokset voivat heijastaa muutakin tietoa kuin mitä määrittelyvaiheessa kuvattu tulkintatapa antaa ymmärtää. Testauspäällikön on oltava valmis analysoimaan huolellisesti mittaritietojen mahdollista poikkeamaa odotetuista lukemista ja syytä kyseiseen eroon.
- Mittaritietojen raportointi. Tavoitteena on tuottaa välittömästi ymmärrettävää tietoa johdon tarpeisiin. Esitykset voivat kuvata tietyn mittarin tilanteen määrätyllä hetkellä tai esittää mittarin kehityksen ajan myötä niin, että on mahdollista arvioida sen suuntauksia.
- Mittareiden kelpoisuus. Testauspäällikön on myös todennettava raportoitavan tiedon oikeellisuus. Mittaria varten kerätyt tiedot eivät ehkä kuvaa projektin todellista tilannetta tai voivat välittää ylipositiivisen tai negatiivisen suuntauksen. Ennen kuin mitään tietoa esitetään, Testauspäällikön on katselmoitava tiedot sekä oikeellisuuden varmistamiseksi sekä sen suhteen, minkälaisen viestin tiedot tulevat todennäköisesti välittämään.

Testauksen edistymistä seurataan pääasiassa viidellä eri alueella:

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

Tuoteriskejä, vikoja, testejä ja kattavuutta voidaan mitata ja niistä voidaan raportoida määrätyllä tavalla projektin tai tehtävän aikana. Jos nämä mittaritiedot liittyvät testaussuunnitelmassa määriteltyihin lopetuskriteereihin, ne voivat tarjota objektiivisen vertailutason, jonka perusteella voidaan arvioida testauksen valmistumista. Luottamusta voidaan mitata tutkimuksilla tai käyttämällä kattavuutta sijaismittarina; luottamuksesta raportoidaan kuitenkin usein myös subjektiivisesti.

Tuoteriskeihin liittyviin mittareihin kuuluvat:

- kuinka paljon riskeistä on täysin katettu läpäistyillä testeillä (prosenttiosuus)

- monenko riskin kohdalla osaa tai yhtään testeistä ei läpäisty (prosenttiosuus)
- kuinka paljon riskeistä on vielä osittain testaamatta (prosenttiosuus)
- katettujen riskien prosenttiosuus lajiteltuna riskiluokittain
- alkuperäisen laaturiskianalyysin perusteella tunnistettujen riskien prosenttiosuus.
-

Vikoihin liittyviin mittareihin kuuluvat:

- raportoitujen (löydettyjen) kumulatiivinen määrä vs. ratkaistujen (korjattujen) kumulatiivinen määrä
- häiriöiden välinen keskimääräinen aika tai häiriöiden saapumisnopeus
- vikamäärien tai prosenttiosuuksien jakaminen osiin seuraavien kriteerien perusteella:
  - erityiset testauskohteet tai komponentit
  - alkuperäissyyt
  - vian lähde (esim. vaatimusmäärittely, uusi piirre, regressio jne.)
  - testijulkaisut
  - vaihe, jossa vika on syntynyt, löydetty ja poistettu
  - kiireellisyys/vakavuus
  - hylätyt raportit tai kaksoiskappaleet
- suuntaukset vian löytämisestä sen ratkaisuun kuluvan ajan kehityksessä
- uusia vikoja paljastaneiden vikakorjausten määrä (kutsutaan joskus tytärvioiksi).

Testaukseen liittyviin mittareihin kuuluvat:

- suunniteltujen, määriteltyjen (toteutettujen), ajettujen, läpäistyjen, hylättyjen, muiden vikojen estämien ja väliin jätettyjen testien kokonaismäärä
- regressio- ja uudelleentestauksen tila, mukaan luettuna trendit sekä hylättyjen regressio- ja uudelleentestien kokonaismäärät
- testaukseen päivässä käytettäväksi suunniteltu aika vs. todellinen käytetty määrä
- testausympäristön käytettävissä olo (kuinka monta prosenttia suunnitellusta ajasta testiympäristö oli testaustiimin käytettävissä).

Testikattavuuteen liittyviin mittareihin kuuluvat:

- vaatimusten ja suunnitteluosien kattavuus
- riskikattavuus
- ympäristö-/kokoonpanokattavuus
- koodikattavuus.

Testauspäällikön on tärkeä ymmärtää, kuinka tulkita ja käyttää kattavuusmittareita, jotta hän ymmärtää testauksen tilan ja pystyy raportoimaan siitä. Ylemmillä testaustasoilla, kuten järjestelmätestauksessa, järjestelmä-integraatiotestauksessa ja hyväksymistestauksessa, pääasiallisena testauksen perustana ovat yleensä esimerkiksi vaatimusmäärittelyt, suunnittelukuvaukset, käyttötapaukset, käyttäjätarinat, tuoteriskit sekä tuetut ympäristöt ja kokoonpanot. Rakenteellisen koodikattavuuden mittarit soveltuvat paremmin testauksen alemmille tasoille kuten yksikkötestaukseen (esim. lause- ja haarakattavuus) sekä komponentti-integraatiotestaukseen (esim. rajapintakattavuus). Vaikka Testauspäällikkö voikin käyttää koodikattavuusmittareita mittaamaan, missä määrin testit käyvät läpi testattavan järjestelmän rakenteen, ylemmän tason testitulosten raportoinnissa ei tyypillisesti pitäisi olla mukana koodikattavuuslukuja. Testauspäälliköiden pitäisi ymmärtää, että vaikka yksikkö- ja komponentti-integraatiotestaus saavuttaisikin 100 % niille asetetuista rakenteellisen kattavuuden tavoitteista, viat ja laaturiskit jäävät käsiteltäviksi ylemmillä testaustasoilla.

Mittareita voidaan yhdistää myös testauksen perusprosessin tehtäviin (jotka on kuvattu Perustason sertifikaattisisällössä ja tässä sertifikaattisisällössä). Näin tekemällä mittareita voidaan käyttää läpi koko testausprosessin itse testausprosessin seuraamisessa samoin kuin projektin etenemisen seuraamisessa kohti sen tavoitteita.

Testauksen suunnittelun ja hallinnan tehtävien seurannassa käytettäviin mittareihin kuuluvat mm.

- riski-, vaatimus- ja muun testauksen pohjamateriaalin osien kattavuus
- vikojen löytyminen
- testimateriaalin kehittämiseen ja testien suoritukseen suunniteltu vs. toteutunut aika.

Testien analysointitehtävien seurannassa käytettäviin mittareihin kuuluvat mm.

- tunnistettujen testattavien tilanteiden määrä
- testianalyysin aikana löydettyjen vikojen määrä (esim. tunnistamalla riskejä tai muita testattavia tilanteita testauksen pohjamateriaalin perusteella).

Testisuunnittelutehtävien seurannassa käytettäviin mittareihin kuuluvat mm.

- kuinka paljon testattavista tilanteista on katettu testitapauksilla (prosenttiosuus)
- testisuunnittelun aikana löydettyjen vikojen määrä (esim. määrittämällä testejä testauksen pohjamateriaalin perusteella).

Testien toteutustehtävien seurannassa käytettäviin mittareihin kuuluvat mm.

- kuinka paljon testiympäristöistä on konfiguroitu (prosenttiosuus)
- kuinka paljon testiaineiston tietueista on ladattu tietokantaan (prosenttiosuus)
- kuinka paljon testitapauksista on automatisoitu (prosenttiosuus).

Testien suoritustehtävien seurannassa käytettäviin mittareihin kuuluvat mm.

- suoritettujen, läpäistyjen ja hylättyjen testien prosenttiosuus suunniteltuista testeistä
- kuinka paljon testattavista tilanteista on katettu suoritetuilla (ja/tai läpäistyillä) testitapauksilla (prosenttiosuus)
- raportoitujen/ratkaistujen vikojen suunniteltu vs. todellinen määrä
- suunniteltu vs. todellinen saavutettu kattavuus.

Testauksen edistymiseen ja loppuun viemiseen kuuluviin tehtävien seurantaan liittyviin mittareihin kuuluu niiden yhdistäminen tarkastuspisteisiin sekä (testauksen suunnittelun yhteydessä määriteltyihin ja hyväksyttyihin) aloitus- ja lopetuskriteereihin, joihin voivat kuulua mm. seuraavat:

- suunniteltujen testattavien tilanteiden, testitapausten tai testisuunnitelmien määrä ja toteutettujen erittely läpäistyihin ja hylättyihin
- vikojen kokonaismäärä, usein eriteltynä vakavuuden, kiireellisyyden, tilan, vikaan liittyvän alijärjestelmän tai muun luokittelun mukaan (ks. luku 4)
- vaadittujen, hyväksyttyjen, toteutettujen ja testattujen muutosten määrä
- suunnitellut vs. toteutuneet kustannukset
- suunniteltu vs. toteutunut kesto
- suunnitellut vs. toteutuneet testauksen tarkistuspisteiden ajankohdat
- testaukseen liittyvien projektien suunnitellut vs. toteutuneet tarkistuspisteiden ajankohdat (esim. koodin jäädyttäminen)
- tuote-(laatu-)riskien tila, usein eriteltynä pienennettyihin vs. pienentämättömiin, suurimpiin riskialueisiin, testianalyysin jälkeen löydettyihin uusiin riskeihin jne.
- testausta estävien seikkojen tai suunniteltujen muutosten vuoksi menetetyn testaustyön, rahan tai ajan prosenttiosuus
- uudelleen- ja regressiotestauksen tila.

Testauksen päätöstehtävien seurannassa käytettäviin mittareihin kuuluvat mm.

- suoritettujen, läpäistyjen, hylättyjen, estettyjen tai suorituksen aikana väliin jätettyjen testitapausten prosenttiosuus
- uudelleenkäytettävien testitapausten tietovarastoon vietyjen testitapausten prosenttiosuus
- automatisoitujen testitapausten tai automatisoitavaksi suunniteltujen vs. automatisoitujen testitapausten prosenttiosuus
- regressiotesteihin integroitujen testitapausten prosenttiosuus
- ratkaistujen/ratkaisemattomien vikaraporttien prosenttiosuus
- nimettyjen ja arkistoitujen testauksen tuotosten prosenttiosuus.
-

Tämän lisäksi testausprosessin seurantaan käytetään usein projektinhallinnallisia vakiotekniikoita kuten tehtävien pienempiin osiin jakamista. Ketterissä tiimeissä testaus on osa käyttäjätarinan etenemistä edistymiskäyrällä. Kun käytössä ovat Lean management –tekniikat, tarinakohtaista testauksen edistymistä seurataan usein siirtämällä käyttäjätarinakorttia sarakkeesta toiseen Kanban-taululla.

Kun käsitellään määrättyä metriikoiden joukkoa, mittaritiedot voidaan raportoida suullisesti sanallisessa muodossa, numeerisesti taulukoiden avulla tai kuvallisesti kaavioita käyttäen. Mittaritietoja voidaan käyttää moniin eri tarkoituksiin, kuten:

- Analyysi, mahdollisten trendien ja syiden löytämiseksi testitulosten kautta
- Raportointi, testitulosten välittämiseksi niistä kiinnostuneille projektin osapuolille ja sidosryhmille
- Hallinta, testauksen tai koko projektin suunnan muuttamiseksi sekä muutoksen aikaansaamien tulosten seuraamiseksi.

Sopivat tavat näiden testauksen mittaritietojen keräämiseksi, analysoimiseksi ja raportoimiseksi riippuvat tietoja käyttävien henkilöiden erityisistä tietotarpeista, tavoitteista ja osaamisesta. Tämän lisäksi testiraportin tarkan sisällön pitäisi vaihdella kohdeyleisön mukaan.

Testauksen hallinnan kannalta on oleellista, että testauksen suunnittelun jälkeen testausprosessin aikana kerättävät metriikat tuottavat Testauspäällikölle tarpeellista tietoa, jotta hän voi ohjata testauksen kohti testausmission, strategioiden ja tavoitteiden onnistunutta saavuttamista. Siksi suunnittelussa on otettava huomioon nämä tietotarpeet ja seurannan täytyy sisältää metriikoiden kerääminen kaikista tarpeellisista tuotoksista. Tarvittavan tiedon määrä ja tiedon keruuseen käytettävä työmäärä riippuvat monista projektiin liittyvistä tekijöistä, kuten projektin koosta, monimutkaisuudesta ja riskeistä.

Testauksen hallinnan täytyy vastata testauksen tuottamaan tietoon samoin kuin projektin tai tehtävän muuttuviin olosuhteisiin. Jos esimerkiksi dynaaminen testaus paljastaa vikaryppäitä alueilla, joilla useiden vikojen esiintymistä pidettiin epätodennäköisenä, tai jos testauksen suoritusaikaa lyhennetään siksi, että testauksen aloitus oli myöhässä, riskianalyysi ja suunnitelma on tarkastettava. Tämä voi johtaa testien uudelleenpriorisointiin ja jäljellä olevan testauksen suorituksen työmäärän uudelleensuunnitteluun.

Jos testauksen edistymisraportti paljastaa poikkeamia testaussuunnitelmasta, on ryhdyttävä testauksen hallintatoimenpiteisiin. Testauksen hallinta kohdistuu projektin ja/tai testauksen ohjaamiseen uudelleen parempaan suuntaan. Kun testituloksia käytetään vaikuttamaan projektin hallinnollisiin toimenpiteisiin tai mittaamaan niitä, on harkittava seuraavia vaihtoehtoja:

- Laaturiskianalyysin, testien prioriteettien ja/tai testisuunnitelmien tarkastaminen
- Resurssien lisääminen tai projektin tai testauksen työn lisääminen muulla keinoin
- Julkaisupäivän lykkääminen
- Testauksen päätöskriteerien löyhentäminen tai kiristäminen
- Projektin rajausten (toiminnalliset ja/tai ei-toiminnalliset) muuttaminen.

Näiden vaihtoehtojen toimeenpano vaatii tyypillisesti projektin tai tehtäväkokonaisuuden sidosryhmien yksimielisyyttä ja projektin tai tehtäväkokonaisuuden johtajien suostumusta.

Testausraportissa kerrotun tiedon pitäisi pohjautua suuressa määrin kohdeyleisön, esim. projektin johdon tai liiketoimintajohdon, tarpeisiin. Projektipäällikkö on todennäköisesti kiinnostunut vikojen yksityiskohtaisista tiedoista; liiketoimintajohtajalle tuoteriskien tila voi olla raportin avainasia.

## 2.7 Testauksen liiketoiminnallinen arvo

Testauspäällikön täytyy pyrkiä optimoimaan testaus niin, että se tuottaa liiketoiminnalle arvoa. Liiallinen testaus ei tuota hyvää liiketoiminnallista arvoa, koska testaus aiheuttaa kohtuuttomia viivästyksiä ja maksaa enemmän kuin se tuottaa. Liian vähäinen testaus ei tuota hyvää liiketoiminnallista arvoa, koska liian monia vikoja toimitetaan käyttäjille. Optimaalinen kohta on näiden kahden ääripään välissä. Testauspäällikön täytyy auttaa testauksen sidosryhmiä ymmärtämään tämä optimaalisuus ja testauksen tuoma arvo.

Vaikka monissa organisaatioissa testausta pidetäänkin jossain mielessä arvokkaana, harvat johtajat, Testauspäälliköt mukaan luettuna, pystyvät määrällistämään, kuvaamaan tai ilmaisemaan muuten selkeästi tuon arvon. Tämän lisäksi Testauspäälliköt, Testauksen johtajat ja testaajat keskittyvät testauk-

sen taktisiin yksityiskohtiin (tietyn tehtävän tai testaustason erityisseikkoihin) ja samalla jättävät huomiotta testaukseen liittyvät laajemmat strategiset (ylemmän tason) ongelmat, joista muut projektin osapuolet, erityisesti johtajat, välittävät.

Testaus tuottaa organisaatiolle, projektille ja/tai tehtäväkokonaisuudelle arvoa sekä määrällisesti että laadullisesti:

- Määrälliseen arvoon kuuluu vikojen löytäminen ja niiden estäminen tai korjaaminen ennen julkaisua, ennen julkaisua tunnettujen vikojen löytäminen (vikoja ei korjata mutta ne dokumentoidaan ehkä mahdollisen kiertotien kanssa), riskin pienentäminen testejä suorittamalla sekä tiedon tuottaminen projektista, prosessista ja tuotteen tilasta
- Laadulliseen arvoon kuuluu kasvanut laadun maine, sulavammat ja ennakoitavammat julkaisut, kasvanut luottamus, suoja oikeudellisilta korvausvaatimuksilta ja kokonaisen toimeksiannon tai jopa ihmishenkien menettämisen riskin pienentäminen.

Testauspäällikön on ymmärrettävä, mitkä näistä arvoista koskevat hänen organisaatiotaan, projektiaan ja/tai tehtäväkokonaisuuttaan, ja hänen on pystyttävä keskustelemaan testauksesta näiden arvojen näkökulmasta.

Hyväksi havaittua menetelmää testauksen määrällisen arvon ja tehokkuuden mittaamiseksi kutsutaan laadun hinnaksi (tai, joskus, huonon laadun hinnaksi). Laadun hintaan kuuluu projektin ja toiminnallisten kustannusten luokittelu neljään ryhmään, jotka liittyvät tuotteen vikojen kustannuksiin:

- Estämisen kustannukset, esim. koulutetaan kehittäjät kirjoittamaan ylläpidettävämpää tai turvallisempaa koodia
- Löytämisen kustannukset, esim. testitapausten kirjoittaminen, testiympäristön konfigurointi ja vaatimusten katselmointi
- Sisäisen häiriön hinta, esim. testauksen tai katselmointien aikana löydetyn vian korjaaminen ennen toimitusta
- Ulkoisen häiriön hinta, esim. asiakkaalle toimitettuun vialliseen ohjelmistoon liittyvät tukikustannukset.

Osan testauksen budjetista muodostavat löytämisen kustannukset (eli raha, joka käytettäisiin, vaikka testaajat eivät löytäisi vikoja, kuten testien kehittämiseen käytetty raha), kun loppu muodostuu sisäisten häiriöiden kustannuksista (eli todellisista löydettyihin vikoihin liittyvistä kustannuksista). Löytämiseen ja sisäisiin häiriöihin liittyvät kokonaiskustannukset ovat tyypillisesti huomattavasti alle ulkoisten häiriöiden kustannusten, mikä tekee testauksesta erittäin arvokasta. Määrittämällä näiden neljän luokan kustannukset Testauspäällikkö voi luoda testauksesta vakuuttavan liiketoimintakuvauksen.

Lisätietoa testauksen liiketoiminnallisesta arvosta ja laadun hinnasta, ks. [Black03].

## 2.8 Hajautettu, ulkoistettu ja paikallinen ulkoistettu testaus

Monissa tapauksissa osan tai ehkä jopa koko testaustyön tekevät ihmiset, jotka ovat eri paikoissa, eri yritysten palkkaamia tai erossa projektitiimistä. Jos testausta tehdään useissa paikoissa, testaustyö on hajautettua. Jos testausta tehdään yhdessä tai useammassa paikassa ja sitä ovat tekemässä henkilöt, jotka eivät ole yrityksen työntekijöitä eivätkä he työskentele samassa paikassa projektitiimin kanssa, testaustyö on ulkoistettua. Jos testausta tekevät henkilöt, jotka työskentelevät samassa paikassa kuin muu projektitiimi mutta joka eivät ole muiden tiimiläisten työtovereita, testaustyö on paikalliseksi ulkoistettua.

Yhteistä kaikille näille testaustyötavoille on, että ne tarvitsevat selkeät viestintäkanavat ja odotukset niiden mission, tehtävien ja tuotosten suhteen on määriteltävä hyvin. Projektitiimin täytyy luottaa vähemmän epämuodollisiin viestintäkanaviin kuten käytäväkeskusteluihin ja työtovereiden väliseen sosiaaliseen kanssakäymiseen. On tärkeää määritellä selkeästi, miten viestinnän pitää tapahtua, ja ottaa kantaa esimerkiksi siihen, miten hoidetaan ongelmien organisaatiossa ylöspäin vieminen, minkä tyyppistä tietoa viestitään, ja mitä viestintätapoja käytetään. Kaikilla, olipa heidän suhteensa tiimiin mikä hyvänsä, on oltava selkeä käsitys roolistaan ja vastuistaan samoin kuin muiden osapuolien rooleista ja vastuista,

jotta vältetään väärinkäsitykset ja epärealistiset odotukset. Erot sijainnissa, aikavyöhykkeissä, kulttuurissa ja kielessä saavat viestintään ja odotuksiin liittyvien ongelmien esiintymisen todennäköisyyden kasvamaan.

Myös yhteistä kaikille näille testaustavoille on menetelmien yhtenäistämisen tarve. Vaikka missä tahansa projektissa voi esiintyä menetelmien epäyhtenäisyyttä, sitä esiintyy todennäköisemmin tilanteissa, joissa työ on hajautettua ja/tai sitä tekevät ulkopuoliset tekijät. Jos kaksi testausryhmää käyttää eri menetelmiä tai testausryhmä käyttää eri menetelmiä kuin kehitys tai projektin hallinto, se voi johtaa merkittäviin ongelmiin erityisesti testien suorituksen aikana. Jos esimerkiksi asiakas käyttää Ketterää kehitystä, kun testauspalveluiden tuottaja noudattaa suunnitelmavetoista testausmenetelmää, joka pohjautuu peräkkäismallia käyttävään elinkaareen, testattavien kohteiden toimituksen ajoitus ja luonne muodostuvat riidan kohteeksi.

Hajautetussa testauksessa testaustyön jako useiden eri paikkojen välillä täytyy olla yksiselitteinen ja järkevästi päätetty. Ilman tällaista ohjausta kaikkein osaavin ryhmä ei ehkä tee testaustyötä, johon he olisivat kykeneviä. Tämän lisäksi, jos kukin tiimi ei tiedä omia vastuualueitaan, ne eivät ehkä tee sitä mitä niiden pitäisi. Odotukset joka tiimin suhteen on tuotava selvästi esille. Ilman huolellista hallintaa testaustyöhän kokonaisuutena voi jäädä aukkoja (jotka kasvattavat jäljellä olevia laaturiskejä, kun järjestelmä toimitetaan) ja siinä voi syntyä päällekkäisyyksiä (mikä heikentää tehokkuutta).

Lopuksi, kaikkien tällaisten testaustöiden osalta on kriittistä, että koko projektitiimille kehittyy luottamus siihen, että testaustiimi suorittaa tehtävänsä kunnolla organisatorisista, kulttuurillisista, kielellisistä ja maantieteellisistä rajoista huolimatta, ja että tämä luottamus säilyy. Luottamuksen puute johtaa varmistustehtävien tehottomuuteen ja viivästyksiin, ongelmien pallotteluun ja organisaatiopolitiikan peluuseen.

## 2.9 Teollisuusstandardien soveltamisen hallinta

Sekä Perustason että Jatkotason sertifikaattisisällössä viitataan useisiin standardeihin. Standardit, joihin on viitattu, kattavat ohjelmistotestauksen elinkaaren, ohjelmistotestauksen, ohjelmiston laatuominaisuudet, katselmoinnit ja vianhallinnan. Testauspäällikön pitäisi olla tietoinen näistä standardeista, organisaationsa politiikasta standardien käytön suhteen sekä siitä, vaaditaanko standardeja vai ei tai onko niiden käyttö tarpeellista tai hyödyllistä.

Standardit voivat olla peräisin monesta eri lähteestä, kuten:

- Kansainväliset lähteet tai sellaiset, joilla on kansainvälisiä tavoitteita
- Kansalliset lähteet, kuten kansainvälisten standardien kansalliset sovellukset
- Toimialakohtaiset lähteet, esimerkiksi silloin, kun kansainvälistä tai kansallista standardia sovelletaan tiettyyn toimialaan, tai tiettyä toimialaa varten kehitetyt standardit.

Kansainvälisiin standardointielimiin kuuluvat ISO ja IEEE. ISO on International Standards Organization, ja sitä kutsutaan myös nimellä IOS, International Organization for Standardization. Se koostuu jäsenistä, jotka edustavat oman maansa kansallista elintä, joka on keskeisin standardoitavan alueen kannalta. Tämä kansainvälinen elin on luonut useita ohjelmistotestaajille hyödyllisiä standardeja, kuten ISO 9126 (joka korvautuu ISO 25000:lla), ISO 12207 [ISO 12207] ja ISO 15504 [ISO 15504].

IEEE on Institute of Electrical and Electronics Engineers, Yhdysvalloissa sijaitseva ammatillinen organisaatio, jossa on kansallisia edustajia yli sadasta maasta. Tämä organisaatio on luonut useita ohjelmistotestaajille hyödyllisiä standardeja, kuten IEEE 829 [IEEE 829] ja IEEE 1028 [IEEE 1028].

Monissa maissa on omat kansalliset standardinsa. Jotkut näistä standardeista ovat hyödyllisiä ohjelmistotestauksessa. Yksi esimerkki on Brittiläinen standardi BS 7925-2 [BS7925-2], joka tarjoaa tietoa monista Jatkotason Sertifikaattisisällön Testausasiantuntija- ja Tekninen Testausasiantuntija –osioissa kuvatuista testisuunnittelutekniikoista.

Jotkut standardit ovat toimialakohtaisia, ja jotkut näistä standardeista vaikuttavat ohjelmistotestaukseen, ohjelmiston laatuun ja ohjelmistokehitykseen. Esimerkiksi ilmailualalla Yhdysvaltain Ilmailuhallitus

FAA:n (Federal Aviation Administration) standardia DO-178B (ja sen vastinetta EU:ssa, ED 12B) sovelletaan siviili-ilma-aluksissa käytettäviin ohjelmistoihin. Tämä standardi kuvaa tietyt rakenteellisen kattavuuden kriteerien tasot, jotka perustuvat testattavan ohjelmiston kriittisyyteen.

Toinen esimerkki toimialakohtaisesta standardista koskee lääkintäalan järjestelmiä ja löytyy Yhdysvaltain Elintarvike- ja lääkintävirastosta, Code of Federal Regulations (CFR), kappale 21, osa 820 [FDA21]. Tämä standardi suosittelee tiettyjä rakenteellisia ja toiminnallisia testaustekniikoita. Standardi suosittelee myös testausstrategioita ja periaatteita, jotka ovat yhteneväisiä ISTQB:n sertifikaattisisältöjen kanssa.

Joissain tapauksissa testaukseen vaikuttaa standardeja tai laajalle levinneitä toimintatapoja, jotka eivät ensisijaisesti koske testausta, mutta jotka vaikuttavat voimakkaasti ohjelmistoprosessiin, jossa testaus tapahtuu. Eräs esimerkki on CMMI® ohjelmistoprosessin kehitysmalli. Se muodostuu kahdesta avainprosessialueesta, todentamisesta ja kelpuutuksesta, joiden usein tulkitaan viittaavan testaustasoihin (kuten todentamisen järjestelmätestaukseen ja kelpuutuksen hyväksymistestaukseen). Sillä on myös merkitystä testausstrategian kannalta, sillä sen tulkitaan usein edellyttävän, että analyyttinen vaatimuspohjainen testaus on sisällytettävä osaksi testausstrategiaa.

Kolme muuta tärkeää esimerkkiä ovat PMI'n PMBOK, PRINCE2® ja ITIL®. PMI ja PRINCE2 ovat yleisesti käytettyjä projektinhallintamenetelmiä, edellinen Pohjois-Amerikassa ja jälkimmäinen Euroopassa. ITIL tarjoaa puitteet sen varmistamiseksi, että IT-ryhmä tuottaa arvokasta palvelua sille organisaatiolle, jossa se sijaitsee. Näissä malleissa määritellyt terminologia ja tehtävät eroavat merkittävästi ISTQB:n sertifikaattisisällöistä ja sanastosta. Kun Testauspäällikkö työskentelee organisaatiossa, joka käyttää PMI'n PMBOKia, PRINCE2:ta ja/tai ITILiä, hänen on ymmärrettävä valitut mallit, niiden käyttötavat ja terminologiat riittävän hyvin pystyäkseen työskentelemään tehokkaasti kyseisessä tilanteessa.

Mitä tahansa standardeja tai menetelmiä käytetäänkin, on tärkeä muistaa, että ne ovat ammattilaisryhmien muodostamia. Standardit heijastavat lähderyhmän yhteistä kokemusta ja viisautta, mutta myös sen heikkouksia. Testauspäällikön pitää olla tietoinen ympäristöönsä ja tilanteeseen liittyvistä standardeista, olivatpa ne muodollisia (kansainvälisiä, kansallisia tai toimialakohtaisia) standardeja tai yrityksen sisäisiä standardeja ja suositeltuja käytäntöjä.

Kun harkitaan useiden standardien käyttöä, on pidettävä mielessä, että jotkut standardit ovat epäyhteneväisiä toisten standardien kanssa, tai ne tarjoavat jopa ristiriitaisia määritelmiä. Testauspäällikön on päätettävä eri standardien hyödyllisyys siinä tilanteessa, jossa testaus tapahtuu. Standardissa esitetty tieto voi olla projektin kannalta hyödyllistä tai se voi olla sen esteenä. Standardit voivat kuitenkin tarjota viittauksia hyväksi todettuihin käytäntöihin ja antaa pohjan testausprosessin organisoinnille.

Joissain tapauksissa standardien noudattaminen on pakollista ja sillä on vaikutuksia testaukseen. Testauspäällikön pitää olla tietoinen kaikista tällaisista standardinmukaisuuden vaatimuksista ja varmistaa, että riittävä yhdenmukaisuus niiden kanssa säilyy.

# 3. Katselmoinnit – 180 min.

## Avainsanat

auditointi, epämuodollinen katselmointi, johdon katselmointi, katselmoija, katselmointi, katselmointi-suunnitelma, läpikäynti, puheenjohtaja, tarkastus, tekninen katselmointi,

## Oppimistavoitteet: Katselmoinnit

### 3.2 Johdon katselmoinnit ja auditoinnit

TM-3.2.1 (K2) Ymmärtää johdon katselmointien ja auditointien avainpiirteet.

### 3.3 Katselmointien hallinta

TM-3.3.1 (K4) Analysoida projektia sopivan katselmointityypin valitsemiseksi ja katselmointien läpi-vientisuunnitelman määrittämiseksi oikeanlaisen suorituksen, seurannan ja vastuunjaon varmistamiseksi.

TM-3.3.2 (K2) Ymmärtää katselmointeihin osallistumiseen vaadittavat seikat, taidot ja aika.

### 3.4 Katselmointien metriikat

TM-3.4.1 (K3) Määritellä katselmoinneissa käytettävät prosessi- ja tuotemittarit.

### 3.5 Muodollisten katselmointien hallinta

TM-3.5.1 (K2) Selittää esimerkkien avulla muodollisen katselmoinnin ominaispiirteet.

## 3.1 Esittely

Katselmoinnit esiteltiin ISTQB Perustason sertifikaattisisällössä tuotteisiin kohdistuvina staattisen testauksen tehtävinä. Auditoinnit ja johdon katselmoinnit keskittyvät enemmän ohjelmistoprosessiin kuin ohjelmistoprosessin tuotoksiin.

Koska katselmoinnit ovat staattisen testauksen muoto, Testauspäälliköt voivat olla vastuussa niiden yleisestä onnistumisesta, erityisesti kun kyse on testaukseen liittyvästä materiaalista. Tarkasteltaessa asiaa laajemmin ohjelmistoprojektin näkökulmasta kyseessä pitäisi kuitenkin olla organisaation politiikkaan kuuluva vastuu. Kun otetaan huomioon, kuinka laajasti muodollisia katselmointeja voidaan käyttää eri alueilla sekä ennen ohjelmistoprojekteja että niiden aikana, vastuullinen osapuoli voi olla Testauspäällikkö, Laatupäällikkö tai koulutettu Katselmointikoordinaattori. Tässä sertifikaattisisällössä vastuuhenkilöstä (olipa hän kuka hyvänsä) käytetään nimitystä katselmointipäällikkö.

Katselmointipäällikön tehtävänä on varmistaa, että olemassa on ympäristö, joka edistää Perustason sertifikaattisisällössä kuvattujen katselmoinnin menestystekijöiden toteutumista. Tämän lisäksi katselmointipäällikön pitää laatia suunnitelma keinoista, joilla voidaan mitata, että katselmoinnit tuottavat tehokkaasti hyötyä.

Koska testaajilla on vahva ymmärrys ohjelmistojärjestelmän toiminnallisesta käyttäytymisestä ja siltä vaadituista ominaisuuksista, testaajien osallistuminen katselmointiprosessiin on tärkeää.

Katselmointeihin osallistujien pitäisi saada koulutus katselmointeihin, jotta he ymmärtävät paremmin kulloisenkin roolinsa katselmointiprosessissa. Kaikkien katselmointiin osallistuvien täytyy olla sitoutuneita hyvin toteutetusta katselmoinnista saataviin hyötyihin.

Mikäli katselmoinnit tehdään kunnolla, ne ovat suurin ja kustannustehokkain yksittäinen tuotettuun laatuun myötävaikuttava tekijä. Tämän vuoksi on äärimmäisen tärkeää, että katselmointipäälliköt pystyvät viemään projekteissaan läpi tuottavia katselmointeja ja osoittamaan näiden katselmointien hyödyt.

Projektissa mahdollisesti tehtäviin katselmointeihin kuuluvat mm.

- sopimuksiin liittyvät katselmoinnit, jotka pidetään projektin alussa ja kun saavutetaan tärkeitä tarkistuspisteitä
- vaatimuskatselmoinnit, jotka tehdään, kun vaatimukset ovat saatavilla katselmoitaviksi. Ihannetilanteessa ne kattavat sekä toiminnalliset että ei-toiminnalliset vaatimukset
- ylemmän tason suunnittelukatselmoinnit, jotka tehdään, kun arkkitehtuurin kokonaissuunnitelma on saatavilla katselmoitavaksi
- yksityiskohtaisten suunnitelmien katselmoinnit, jotka tehdään, kun yksityiskohtaiset suunnitelmat ovat saatavilla katselmoitaviksi
- koodikatselmoinnit, jotka tehdään, kun yksittäiset ohjelmistomoduulit toteutetaan. Ne voivat sisältää yksikkötestit ja niiden tulokset sekä itse ohjelmakoodin
- testauksen tuotosten katselmoinnit, jotka voivat kattaa testaussuunnitelmat, testattavat tilanteet, laaturiskianalyysin tulokset, testitapaukset, testiaineiston, testiympäristön ja testitulokset
- testauksen aloitus- (testivalmius)katselmoinnit ja testauksen päätöskatselmoinnit jokaisella testaustasolla. Niissä tarkastetaan testauksen aloituskriteerit ennen testien suorituksen aloittamista ja testien päätöskriteerit ennen testauksen lopettamista
- hyväksymiskatselmoinnit, joita käyttävät asiakkaat tai sidosryhmät järjestelmän hyväksymiseen.

Sen lisäksi, että katselmoitavaan kohteeseen käytetään useita katselmointityyppejä, katselmointipäällikön on tärkeä muistaa, että vaikka katselmoinnit voivat paljastaa vikoja staattisessa dokumentissa, katselmointeja pitäisi laajentaa muilla staattisen testauksen muodoilla (kuten staattinen analyysi) ja koodin dynaamisella testauksella. Näiden tekniikoiden yhdistelmien käyttäminen parantaa testikattavuutta ja paljastaa lisää vikoja.

Eri tekniikat keskittyvät eri asioihin. Esimerkiksi katselmointi voi poistaa ongelman vaatimustasolla ennen kuin sitä on ehditty toteuttaa koodiin. Staattinen analyysi voi auttaa vahvistamaan ohjelmointistandardien käyttöä ja tarkistamaan ongelmia, jotka tiimin olisi liian työlästä löytää tutkimalla katselmoinnin

kohdetta. Tarkastukset eivät aina johda pelkästään vikojen paljastumiseen ja poistamiseen, vaan ne voivat myös opettaa materiaalin laatijaa siinä, kuinka he välttävät vikojen luomista tuotoksiinsa.

ISTQB Perustason sertifikaattisisältö esitteli seuraavat katselmointityypit:
- Epämuodollinen katselmointi
- Läpikäynti
- Tekninen katselmointi
- Tarkastus.

Näiden lisäksi Testauspäällikkö saattaa olla mukana myös
- johdon katselmoinneissa
- auditoinneissa.

## 3.2 Johdon katselmoinnit ja auditoinnit

Johdon katselmointeja käytetään edistymisen seurantaan, tilanteen arviointiin ja tuleviin toimenpiteisiin liittyvien päätösten tekemiseen. Nämä katselmoinnit tukevat projektin tulevaisuuteen liittyviä päätöksiä, kuten esimerkiksi resurssien määrän muuttaminen, korjaavien toimenpiteiden käynnistäminen tai projektin laajuuden muuttaminen.

Seuraavat ovat johdon katselmointien avainpiirteitä:
- Niitä vetävät johtajat, jotka ovat suoraan vastuussa projektista tai järjestelmästä, tai ne pidetään heitä varten
- Niitä pitävät sidosryhmät tai päätöstentekijät, esim. ylemmän tason johtajat tai esimiehet, tai ne pidetään heitä varten
- Niissä tarkistetaan yhdenmukaisuus suunnitelmien kanssa ja mahdolliset poikkeamat niistä
- Niissä tarkistetaan hallinnollisten toimenpiteiden riittävyys
- Niissä arvioidaan projektiriskejä
- Niissä arvioidaan toimenpiteiden vaikutuksia ja tapoja näiden vaikutusten mittaamiseksi
- Niissä laaditaan listat toimenpiteitä vaativista asioista, ratkaistavista ongelmista ja tehtävistä päätöksistä.

Prosesseihin kohdistuvat johdon katselmoinnit, kuten projektin jälkipalaverit (eli saatujen kokemusten läpikäynnit) ovat keskeinen osa prosessien kehitystoimenpiteitä.

Testauspäälliköiden pitäisi osallistua testauksen edistymistä koskeviin johdon katselmointeihin ja he voivat käynnistää niitä.

Auditoinnit suoritetaan yleensä todentamaan yhdenmukaisuutta määriteltyjen kriteerien kanssa, joita yleisimmin ovat tilanteeseen soveltuvat standardit, säädöksien aiheuttamat rajoitteet tai sopimukselliset velvoitteet. Sellaisena auditointien tarkoitus on tarjota prosessien, säädösten, standardien ym. noudattamisen riippumaton arviointi. Seuraavat ovat auditointien avainpiirteitä:
- Pääauditoijan vetämiä ja hallinnoimia
- Yhdenmukaisuudesta kerätään todisteita haastattelujen ja havainnointien avulla sekä asiakirjoja tutkimalla
- Dokumentoituihin tuloksiin kuuluvat havainnot, suositukset, korjaavat toimenpiteet sekä auditoinnin hyväksytty/hylätty-arviointi.

## 3.3 Katselmointien hallinta

Katselmoinnit pitäisi suunnitella tapahtuviksi ohjelmistoprojektin luonnollisissa pysähdyspaikoissa tai tarkastuspisteissä. Tyypillisesti katselmointeja pitäisi pitää vaatimus- ja suunnittelumäärittelyjen jälkeen niin, että ne käynnistetään liiketoiminnan tavoitteista ja siitä edetään kohti alinta suunnittelutasoa. Johdon katselmointien pitäisi tapahtua merkittävissä projektin tarkistuspisteissä, usein osana todentamista testien suorituksen tai muiden projektin merkittävien vaiheiden aikana sekä ennen niitä ja niiden jälkeen. Katselmointistrategia täytyy koordinoida testauspolitiikan ja yleisen testausstrategian kanssa.

Ennen projektitasoisen yleisen katselmointisuunnitelman laatimista katselmointipäällikön (joka voi olla Testauspäällikkö) on otettava huomioon

- mitä pitää katselmoida (tuote ja prosessit)
- kenen pitäisi osallistua tiettyihin katselmointeihin
- mitä oleellisia riskitekijöitä halutaan kattaa.

Katselmointipäällikön pitää jo aikaisin projektin suunnitteluvaiheessa tunnistaa katselmoitavat kohteet ja valita soveltuvat katselmointityypit (epämuodollinen katselmointi, läpikäynti, tekninen katselmointi tai tarkastus, tai kahden tai kolmen tyypin sekoitus) sekä muodollisuuden taso. Tämä on kohta, jossa voidaan suositella lisäkoulutusta katselmointeihin. Tämän jälkeen voidaan määritellä katselmointiprosessin budjetti (aika ja resurssit). Budjetin määrittelyssä pitäisi käyttää riskiarviointia ja laskelmia sijoitukselle saatavasta tuotosta.

Katselmoinneissa sijoitukselle saatava tuotto muodostuu erosta niiden kustannusten välillä, joita syntyy katselmoinnin läpiviennistä ja niiden, joita syntyy, jos katselmointeja ei tehdä ja samoja vikoja joudutaan selvittämään myöhemmässä vaiheessa (tai ne jäävät kokonaan huomaamatta). Kappaleessa 2.7 esiteltyä laatukustannusten laskentaa voidaan käyttää apuna tämän luvun laskemisessa.

Ihanteellisen katselmointien suoritusajan määrittäminen riippuu seuraavista:

- katselmoitavan materiaalin saatavuus riittävän viimeistellyssä muodossa
- oikeiden henkilöiden saatavuus osallistumaan katselmointiin
- ajankohta, jolloin materiaalin lopullisen version pitäisi olla saatavilla
- aika, joka tarvitaan kyseisen materiaalin katselmointiprosessiin.

Katselmointipäällikön pitää testauksen suunnittelun yhteydessä määritellä riittävät mittarit katselmointien arviointia varten. Jos käytetään tarkastuksia, materiaalin laatijan pyynnöstä tulisi pitää lyhyitä tarkastuksia sitä mukaa, kun asiakirjan osat (esim. yksittäiset vaatimukset tai luvut) valmistuvat.

Katselmointiprosessin tavoitteet pitää määritellä testaussuunnittelun aikana. Näihin kuuluvat tehokkaiden ja tuottavien katselmointien läpivienti ja yhteisymmärrykseen pääseminen koskien katselmoinnista saatua palautetta.

Projektikatselmointeja pidetään säännöllisesti koko järjestelmälle ja ne voivat olla myös tarpeen alijärjestelmille ja jopa yksittäisille ohjelmiston osille. Katselmointien määrä, tyyppi, organisaatio ja niihin osallistuvat ihmiset riippuvat kaikki projektin koosta ja monimutkaisuudesta sekä tuoteriskeistä.

Ollakseen tuottavia, katselmointeihin osallistujilla täytyy olla riittävästi sekä teknistä että menettelytapoihin liittyvää osaamista. Perusteellisuus ja yksityiskohtien huomiointi ovat muita katselmoijilta vaadittavia taitoja, jotta katselmoinnit olisivat tehokkaita. Selkeys ja oikea priorisointi ovat ominaisuuksia, jotka löytyvät hyvistä katselmointikommenteista. Menettelytapojen osaamisen tarve voi merkitä sitä, että tarvitaan jonkinlaista koulutusta sen varmistamiseksi, että katselmoijat ymmärtävät roolinsa ja vastuunsa katselmointiprosessissa.

Katselmoinnin suunnittelussa pitäisi ottaa huomioon katselmointien suorituksessa teknisiin ja organisatorisiin tekijöihin liittyvät riskit sekä ihmisiin liittyvät tekijät. Onnistuneen katselmoinnin kriittinen tekijä on, että saatavilla on katselmoijia, joilla on riittävä tekninen osaaminen. Kaikkien projektin tiimien pitäisi olla osallisina katselmointien suunnittelussa, minkä pitäisi varmistaa, että jokainen tiimi on sitoutunut katselmointiprosessin onnistumiseen. Suunnittelussa täytyy varmistaa, että jokainen organisaatio antaa tarvittaville katselmoijille riittävästi aikaa, jotta nämä voivat valmistautua ja osallistua katselmointeihin projektiaikataulun sovituissa kohdissa. Aikaa pitää varata myös katselmoijien mahdollisesti tarvitsemaa teknistä tai prosessikoulutusta varten. Varakatselmoijat on nimettävä siltä varalta, että avainkatselmoijat eivät henkilökohtaisten tai liiketoiminnan suunnitelmien muutosten vuoksi olekaan käytettävissä.

Muodollisen katselmoinnin varsinaisen suorituksen aikana katselmoinnin vetäjän on varmistettava, että

- osallistujat tuottavat riittäviä mittaritietoja katselmoinnin tehokkuuden arvioimiseksi
- tarkistuslistoja luodaan ja ylläpidetään tulevien katselmointien parantamiseksi

- määritellään vian vakavuuden ja kiireellisyyden arviointi, jota voidaan käyttää katselmoinnissa löydettyjen ongelmien kohdalla vianhallinnassa (ks. luku 4).

Jokaisen katselmoinnin jälkeen katselmoinnin vetäjän pitää

- kerätä katselmoinnin mittaritiedot ja varmistaa, että esiin tulleet ongelmat ratkaistaan niin, että katselmoinnille testauksen näkökulmasta asetut tavoitteet saavutetaan
- käyttää katselmoinnin mittaritietoja syötteenä, kun määritetään katselmoinneista sijoitukselle saatu tuotto (ROI)
- tuottaa palautetta asiaankuuluville sidosryhmille
- tuottaa palautetta katselmointiin osallistuneille.

Katselmointien tehokkuuden arvioimiseksi Testauspäällikkö voi verrata katselmointien jälkeen tehdyn testauksen todellisia tuloksia katselmointiraportteihin kirjattuihin tuloksiin. Tilanteissa, joissa tuote on katselmoitu ja hyväksytty katselmoinnin perusteella, mutta jälkeenpäin todettu vialliseksi, katselmoinnin vetäjän on mietittävä tapoja, jotka ovat saattaneet mahdollistaa vikojen karkaamisen katselmointiprosessin läpi. Mahdollisiin syihin kuuluvat ongelmat katselmointiprosessissa (esim. huonot aloitus-/päätöskriteerit), katselmointitiimin vääränlainen kokoonpano, riittämättömät katselmointityökalut (tarkistuslistat jne.), riittämätön katselmoijien koulutus ja kokemus sekä liian lyhyt valmistautumis- ja katselmointipalaveriaika.

Jos vikojen (erityisesti merkittävien) huomataan karkaavan toistuvasti useissa projekteissa, se kertoo, että katselmointien läpiviennissä on merkittäviä ongelmia. Tällaisessa tilanteessa katselmoinnin vetäjän pitää katselmoida katselmointiprosessi ja ryhtyä sopiviin toimenpiteisiin. On myös mahdollista, että erilaisista syistä johtuen katselmoinnit menettävät ajan myötä tehokkuutensa. Tällainen tilanne paljastuu projektien jälkipalavereissa katselmointien laskeneena vianlöytötehokkuutena. Tällöin katselmoinnin vetäjän on jälleen tutkittava tilanne ja korjattava syyt. Missään tapauksessa katselmointien mittaritietoja ei saa käyttää yksittäisten katselmoijien tai materiaalin laatijoiden rankaisemiseen tai palkitsemiseen, vaan niiden pitää kohdistua itse katselmointiprosessiin.

## 3.4 Katselmointien metriikat

Katselmointipäälliköiden (jotka voivat olla Testauspäälliköitä, kuten edellisissä kappaleissa on mainittu) täytyy varmistaa, että saatavilla on mittaritietoa

- katselmoinnin kohteen laadun arvioimiseksi
- katselmoinnin läpiviennin kustannusten arvioimiseksi
- katselmoinnin läpiviennistä saatujen jatkohyötyjen arvioimiseksi.

Katselmointien vetäjät voivat käyttää mittaritietoja sijoitukselle saatavan tuoton sekä katselmointien tehokkuuden määrittämiseen. Näitä metriikoita voidaan käyttää myös raportointiin ja prosessin kehitystehtäviin.

Jokaisen katselmoidun tuotoksen osalta voidaan mitata seuraavat metriikat ja raportoida ne tuotteen arviointia varten:

- Tuotoksen koko (sivuja, koodirivejä jne.)
- Valmisteluaika (ennen katselmointia)
- Katselmointiin kulunut aika
- Vikojen korjaamiseen kulunut aika
- Katselmointiprosessin kesto
- Löydettyjen vikojen määrä ja vakavuus
- Tunnistetut vikaryppäät (eli alueet, joiden vikatiheys on suurempi)
- Katselmoinnin tyyppi (epämuodollinen katselmointi, läpikäynti, tekninen katselmointi tai tarkastus)
- Keskimääräinen vikatiheys (esim. vikoja sivua tai tuhatta koodiriviä kohti)
- Arvioidut jäljelle jääneet viat (tai jäljellä oleva vikatiheys).

Jokaisen katselmoinnin osalta voidaan mitata seuraavat metriikat ja raportoida ne prosessin arviointia varten:

- Vikojen löytötehokkuus (ottaen huomioon myöhemmin tuotteen elinkaaren aikana löydetyt viat)
- Katselmointiprosessin työmäärän ja ajoituksen parantaminen
- Suunniteltujen tuotosten prosentuaalinen kattavuus
- Löydettyjen vikojen tyyppi ja vakavuus
- Osallistujakyselyt koskien katselmointiprosessin sisäistä ja kokonaistehokkuutta
- Katselmoinnissa löytyneiden vikojen laadun mittauskustannukset suhteessa dynaamisen testauksen ja tuotannon vikoihin
- Katselmoinnin vaikutuksen korrelaatio (katselmointityyppi vs. vikojen löytötehokkuus)
- Katselmoijien määrä
- Löydettyjen vikojen määrä suhteessa käytettyyn työtuntiin
- Arvioitu projektin säästämä aika
- Keskimääräinen vikakohtainen työmäärä (löytämisen ja korjaamisen kokonaisaika jaettuna vikojen määrällä).

Tämän lisäksi yllä mainitut tuotteen arviointiin liittyvät metriikat ovat myös hyödyllisiä prosessin arvioinnissa.

## 3.5 Muodollisten katselmointien hallinta

ISTQB Perustason sertifikaattisisällössä kuvataan muodollisen katselmoinnin eri vaiheet: suunnittelu, käynnistys, yksilöllinen valmistautuminen, katselmointipalaveri, uusintatyö ja seuranta. Jotta muodollinen katselmointi viedään oikein läpi, katselmointipäällikön on varmistettava, että katselmointiprosessin kaikkia vaiheita noudatetaan.

Muodollisiin katselmointeihin liittyy joukko ominaisuuksia, kuten:

- määritellyt aloitus- ja lopetuskriteerit
- tarkistuslistat, joita katselmoijat käyttävät
- tuotokset, kuten raportit, arviointilomakkeet tai muut katselmoinnin yhteenvetolomakkeet
- mittaritiedot katselmoinnin sisäisestä ja kokonaistehokkuudesta sekä edistymisestä raportointia varten.

Ennen muodollisen katselmoinnin käynnistämistä katselmointipäällikön on varmistettava, että katselmoinnin esiehdot (jotka on määritelty katselmointiproseduureissa tai aloituskriteerit sisältävässä listassa) on täytetty.

Jos muodollisen katselmoinnin esiehdot eivät täyty, katselmointipäällikkö voi esittää yhtä seuraavista vaihtoehdoista katselmoinneista vastaavalle taholle lopullista päätöstä varten:

- katselmoinnin ja sen tavoitteiden uudelleenmäärittely
- tarpeelliset korjaavat toimenpiteet, jotta katselmoinnissa voidaan edetä
- katselmoinnin lykkääminen.

Muodollisten katselmointien hallinnan osana näitä katselmointeja seurataan koko (ylemmän tason) katselmointisuunnitelman näkökulmasta ja ne liittyvät projektin laadunvarmistuksen tehtäviin. Muodollisten katselmointien hallintaan kuuluu tuote- ja prosessimittareiden avulla kerättävä palaute.

# 4. Vianhallinta – 150 min.

## Avainsanat

alkuperäissyy, häiriö, kiireellisyys, poikkeama, vaiheen vikarajaus, vakavuus, vianluokitteluraati, vika, väärä negatiivinen tulos, väärä positiivinen tulos

## Oppimistavoitteet: Vianhallinta

### 4.2 Vikojen elinkaari ja ohjelmistokehityksen elinkaari

TM-4.2.1 (K3) Kehittää testausorganisaatiolle vianhallintaprosessi, mukaan luettuna havaintoraportin elinkaari, jota voidaan käyttää projektin vikojen seurantaan ja hallintaan läpi testauksen elinkaaren.

TM-4.2.2 (K2) Selittää tehokkaaseen vianhallintaan tarvittava prosessi ja osallistujat.

### 4.3 Vikaraportin tiedot

TM-4.3.1 (K3) Määritellä vianhallintaprosessin aikana kerättävä tietosisältö ja luokittelutiedot.

### 4.4 Prosessin kyvykkyyden arviointi vikaraportin tietojen avulla

TM-4.4.1 (K2) Selittää, kuinka vikaraportin tietoja voidaan käyttää testausprosessin ja ohjelmistokehitysprosessin kyvykkyyden arvioinnissa.

## 4.1 Esittely

Organisaation vianhallintaprosessi ja hallintatyössä käytettävä työkalu ovat tärkeydeltään kriittisiä ei pelkästään testaustiimille vaan kaikille tiimeille, jotka ovat tekemisissä ohjelmistokehityksen kanssa. Tehokkaan vianhallinnan kautta kerätty tieto antaa Testauspäällikölle ja muille projektin sidosryhmille mahdollisuuden luoda näkemys projektin tilaan läpi koko kehityselinkaaren, ja keräämällä ja analysoimalla tietoja pidemmältä ajalta voidaan paikallistaa mahdollisia testaus- ja kehitysprosessin parannuskohteita.

Sen lisäksi, että Testauspäällikön täytyy ymmärtää yleinen vikojen elinkaari ja kuinka sitä käytetään testaus- ja ohjelmistokehitysprosessien seurantaan ja hallintaan, hänen on myös tiedettävä, minkä tiedon tallentaminen on kriittistä, ja toimittava sekä prosessin että valitun vianhallintatyökalun oikean käytön puolestapuhujana.

## 4.2 Vikojen elinkaari ja ohjelmistokehityksen elinkaari

Kuten Perustason sertifikaattisisällössä on selitetty, vikoja syntyy, kun ihminen tekee virheen tuotosta laatiessaan. Tämä tuotos voi olla vaatimusmäärittely, käyttäjätarina, tekninen dokumentti, testitapaus, ohjelmakoodi tai mikä tahansa muu ohjelmistokehityksen tai ylläpitoprosessin aikana syntyvä tuote.

Vikoja voi syntyä missä tahansa ohjelmistokehityksen elinkaaren vaiheessa ja mihin tahansa ohjelmistoon liittyvään tuotokseen. Siksi jokaisen ohjelmistokehityksen elinkaaren vaiheen tulisi sisältää vikojen löytämiseen ja poistamiseen kuuluvat tehtävät. Esimerkiksi staattisia testaustekniikoita (katselmointeja ja staattista analyysiä) voidaan käyttää suunnittelukuvauksiin, vaatimusmäärittelyihin ja koodiin, ennen kuin nämä tuotokset toimitetaan käytettäväksi seuraavissa tehtävissä. Mitä aikaisemmin jokainen vika löydetään ja poistetaan, sitä pienemmät ovat järjestelmän laadun kokonaiskustannukset; tietyn vaiheen vikojen laatukustannukset minimoidaan poistamalla jokainen vika samassa vaiheessa kuin missä se syntyi (eli ohjelmistoprosessi saavuttaa täydellisen vaiheen vikarajauksen). Tämän lisäksi, kuten Perustason sertifikaattisisällössä on selitetty, staattinen testaus löytää suoraan vikoja sen sijaan, että se löytäisi häiriöitä, ja näin vian poistamiskustannukset ovat pienemmät, koska vian löytämiseen ei tarvita virheenjäljitystoimenpiteitä.

Dynaamisen testauksen tehtävissä esimerkiksi yksikkötestauksessa, integraatiotestauksessa ja järjestelmätestauksessa vian läsnäolo paljastuu, kun se aiheuttaa häiriön, josta seuraa ero testin todellisten ja odotettujen tulosten välillä (eli poikkeama). Joissain tapauksissa syntyy väärä negatiivinen lopputulos, kun testaaja ei huomaa poikkeamaa. Jos testaaja huomaa poikkeaman, on syntynyt tilanne, joka vaatii lisätutkimusta. Tämä tutkiminen alkaa vikaraportin laatimisella.

Testiohjatussa kehityksessä automatisoituja yksikkötestejä käytetään eräänlaisena suoritettavana suunnittelukuvauksena. Kun koodi on toteutettu, se suoritetaan välittömästi näitä testejä käyttämällä. Siihen asti, kunnes osan kehitys on kokonaan valmis, joku tai kaikki testit hylkääntyvät. Siksi tällaisen testin aiheuttama häiriö ei merkitse vikaa ja sitä ei tyypillisesti jäljitetä.

### 4.2.1 Vian elinkaari ja tilat

Useimmat testausorganisaatiot käyttävät työkalua vikaraporttien hallintaan vian elinkaaren aikana. Tyypillisesti vikaraportti kehittyy läpi elinkaaren ja kulkee läpi tilojen sarjan, kun se etenee läpi vian elinkaaren. Useimmissa näistä tiloista yksittäinen vian elinkaaren osapuoli omistaa vikaraportin ja hänen vastuullaan on suorittaa tehtävä, jonka valmistumisesta seuraa vikaraportin siirtyminen seuraavaan tilaan (ja raportin osoittaminen seuraavalle vastuulliselle osapuolelle). Päätöstiloissa, kuten esimerkiksi kun vikaraportti suljetaan (tarkoittaa yleensä, että taustalla oleva vika on korjattu ja korjaus on todennettu uudelleentestauksella), peruutetaan (tarkoittaa yleensä, että vikaraportti on epäkelpo), ei ole toistettavissa (tarkoittaa yleensä, että poikkeamaa ei voida enää havaita), tai raportti lykätään (tarkoittaa yleensä, että poikkeama liittyy todelliseen vikaan, mutta vikaa ei korjata projektin aikana), raportilla ei ole omistajaa, koska jatkotoimenpiteitä ei tarvita.

Testaajien testauksen aikana löytämiin vikoihin liittyy erityisesti kolme tilaa, joihin liittyvä toiminta kuuluu testaustiimille:

- Alkutila

- Tässä tilassa yksi tai useampia testaajia kerää tarvittavat tiedot vian selvittämisestä vastuussa olevalle henkilölle poikkeaman toistamista varten (vikaraporttiin sisällytettävät tiedot: lisätietoa ks. kappale 4.3).
- Tähän tilaan voidaan myös viitata nimellä “avoin” tai “uusi”.

- Palautustila
  - Tässä tilassa raportin vastaanottaja on hylännyt raportin tai pyytää testaajalta lisätietoja. Tämä tila voi ilmaista puutetta alkuperäisessä tiedonkeruuprosessissa tai itse testauksessa, ja Testauspäälliköiden pitäisi seurata liiallisia palautuslukuja. Testaajan pitää toimittaa lisätietoja tai vahvistaa, että raportti todella pitäisi hylätä.
  - Tähän tilaan voidaan myös viitata nimellä “hylätty” tai ”selvitettävänä”.
- Uusintatestaustila
  - Tässä tilassa testaaja suorittaa uusintatestin (joka usein seuraa vikaraportin sisältämiä askeleita häiriön toistamiseksi) sen varmistamiseksi, onko korjaus todella ratkaissut ongelman. Jos uusintatesti osoittaa, että vika on korjattu, testaajan pitää sulkea raportti. Jos uusintatesti osoittaa, että vika ei ole korjautunut, testaajan pitää avata raportti uudelleen, jolloin se osoitetaan uudelleen edelliselle omistajalle, joka voi täydentää vian korjaamiseen tarvittavat työt.
  - Tähän tilaan voidaan myös viitata nimellä “ratkaistu” tai ”todennettavana”.

### 4.2.2 Epäkelpojen vikaraporttien ja kaksoiskappaleiden hallinta

Joissakin tapauksissa poikkeama ei ole seuraus viasta vaan ennemminkin ongelmasta testiympäristössä, testiaineistossa tai jossain muussa testimateriaalin osassa, tai se johtuu testaajan omasta väärinymmärryksestä. Jos testaaja luo vikaraportin, josta myöhemmin todetaan, että se ei liity vikaan testattavassa tuotteessa, kyseessä on väärä positiivinen tulos. Tällaiset raportit tyypillisesti peruutetaan tai suljetaan epäkelpoina vikaraportteina. Lisäksi joissain tapauksissa vika voi ilmentyä erilaisina oireina, jotka testaajasta voivat näyttää täysin toisiinsa kuulumattomilta. Jos on laadittu kaksi tai useampia vikaraportteja, joiden myöhemmin huomataan liittyvän samaan perussyyhyn, tyypillisesti yksi raporteista säilytetään ja muut suljetaan vikaraportin kaksoiskappaleina.

Vaikka epäkelvot raportit ja kaksoiskappaleet aiheuttavatkin osittain tehottomuutta, pieni määrä tällaisia raportteja syntyy väistämättä ja Testauspäällikön pitäisi hyväksyä asia sellaisena. Kun päälliköt pyrkivät estämään kaikki epäkelvot vikaraportit ja kaksoiskappaleet, väärien negatiivisten määrä tyypillisesti kasvaa, sillä testaajia kannustetaan olemaan raportoimatta vikoja. Tämä vähentää testausorganisaation vianlöytötehokkuutta, joka useimmissa tapauksissa liittyy testausorganisaation avaintavoitteisiin.

### 4.2.3 Monialainen vianhallinta

Vaikka testausorganisaatio ja Testauspäällikkö tyypillisesti omistavat koko vianhallintaprosessin ja vianhallintatyökalun, monialainen tiimi on yleensä vastuussa tietyssä projektissa raportoitujen vikojen hallinnasta. Testauspäällikön lisäksi vianhallintaraatiin (tai vianluokitteluraatiin) kuuluu tyypillisesti edustajia kehityksestä, projektinhallinnasta, tuotehallinnasta ja muista sidosryhmistä, jotka ovat kiinnostuneita kehitettävästä ohjelmistosta.

Kun poikkeamia löydetään ja syötetään vianhallintatyökaluun, vianhallintaraadin pitäisi tavata päättämään, edustaako jokainen vikaraportti kelvollista vikaa, ja pitääkö vika korjata vai lykätä. Tämän päätöksen tekemiseksi vianhallintaraadin on pohdittava vian korjaukseen tai korjaamatta jättämiseen liittyviä hyötyjä, riskejä ja kustannuksia. Jos vika korjataan, tiimin pitää määritellä vian korjauksen kiireellisyys suhteessa muihin projektin tehtäviin. Testauspäälliköltä ja testaustiimiltä voidaan kysyä näkemystä vian suhteellisesta tärkeydestä ja heidän pitää toimittaa saatavilla oleva objektiivinen tieto.

Vianhallintatyökalua ei pidä käyttää hyvän viestinnän korvikkeena eikä vianhallintaraadin kokouksia pidä käyttää korvaamaan hyvän vianhallintatyökalun tehokasta käyttöä. Viestintä, riittävä työkalutuki, hyvin määritelty vian elinkaari ja aktiivinen vianhallintaraati ovat kaikki tarpeellisia tehokkaaseen ja tuottavaan vianhallintaan.

## 4.3 Vikaraportin tiedot

Kun löydetään vika (osana staattista testausta) tai todetaan häiriö (osana dynaamista testausta), asianomaisten henkilöiden pitää kerätä siitä tiedot ja liittää ne vikaraporttiin. Tämän tiedon pitää palvella kolmea tarkoitusta:

- Raportin hallinta läpi vian elinkaaren
- Projektin tilan arviointi erityisesti suhteessa tuotteen laatuun ja testauksen edistymisen arviointi
- Prosessin kyvykkyyden arviointi (kuten kuvattu kappaleessa 4.4 alla).

Vikaraportin hallintaan ja projektin tilaan liittyvät tiedot voivat vaihdella riippuen siitä, missä elinkaaren vaiheessa vika löydetään, ja aikaisessa vaiheessa (esim. vaatimuskatselmoinnit ja yksikkötestaus) tarvitaan tyypillisesti vähemmän tietoa. Kerättävien perustietojen pitää kuitenkin olla yhdenmukaisia läpi elinkaaren ja ihannetilanteessa läpi projektien, jotta voidaan tehdä merkityksellistä prosessin vikatietojen vertailua sekä projektin sisällä että läpi kaikkien projektien.

Vikatietojen keruu voi auttaa testauksen edistymisen seurannassa, hallinnassa ja päätöskriteerien arvioinnissa. Vikatietojen pitäisi esimerkiksi tukea vikatiheysanalyysiä, löydettyjen ja ratkaistujen vikojen trendianalyysiä, keskimääräisen ajan selvittämistä vian löytämisestä sen selvittämiseen, ja kun selvitetään häiriötiheyttä (esim. MTBF-analyysi , häiriöiden välinen keskimääräinen aika).

Kerättävät vikatiedot voivat sisältää seuraavia asioita:

- Vian löytäneen henkilön nimi
- Henkilön rooli (esim. loppukäyttäjä, liiketoiminta-asiantuntija, kehittäjä, teknisen tuen edustaja)
- Testauksen tyyppi (esim. käytettävyystestaus, suorituskykytestaus, regressiotestaus)
- Ongelman yhteenveto
- Ongelman yksityiskohtainen kuvaus
- Häiriön (tai vian) toistamiseen tarvittavat askeleet, sekä todelliset ja odotetut tulokset (poikkeama korostettuna) sekä näytönkuvat, tietokantaotokset ja lokit, mikäli käytettävissä
- Vian syntymisen, löytämisen ja poistamisen elinkaarivaihe, mukaan luettuna testausvaihe, jos sopii tilanteeseen
- Tuotos, jossa vika syntyi
- Vaikutuksen vakavuus järjestelmän ja/tai tuotteen sidosryhmien kannalta (määritellään yleensä järjestelmän teknisen käyttäytymisen perusteella)
- Ongelman korjauksen kiireellisyys (määritellään yleensä häiriön liiketoiminnalle aiheuttaman vaikutuksen perusteella)
- Alijärjestelmä tai komponentti, jossa vika sijaitsee (vikaryppäiden analyysia varten)
- Projektin tehtävä, jonka aikana ongelma havaittiin
- Tunnistamismenetelmä, joka paljasti ongelman (esim. katselmointi, staattinen analyysi, dynaaminen testaus, tuotantokäyttö)
- Vian tyyppi (vastaa yleensä vikaluokittelua, jos käytössä)
- Laatuominaisuudet, joihin vika vaikuttaa
- Testiympäristö, jossa vika havaittiin (koskee dynaamista testausta)
- Projekti ja tuote, joissa vika on
- Tämänhetkinen omistaja; eli henkilö, joka tällä hetkellä työskentelee ongelman parissa, edellyttäen että raportti ei ole päätöstilassa
- Raportin tämänhetkinen tila (yleensä hallitaan vianjäljitystyökalulla osana elinkaarta)
- Yksittäiset tuotokset (esim. testattavat nimikkeet ja niiden versionumerot), joissa ongelma havaittiin, sekä yksittäiset tuotokset, joissa ongelma lopulta ratkaistiin.
- Vaikutus projektin ja tuotteen sidosryhmien hyötyihin
- Johtopäätökset, suositukset ja ongelman ratkaisemiseksi tehtyjen tai tekemättä jätettyjen toimenpiteiden hyväksynnät
- Vian korjaamiseen tai korjaamatta jättämiseen liittyvät riskit, kustannukset, mahdollisuudet ja hyödyt
- Päivämäärät, jolloin vian elinkaaren eri siirtymät tapahtuivat, jokaista siirtymää seurannut raportin omistaja, ja toimenpiteet, jotka projektitiimin jäsenet suorittivat vian löytämiseksi ja korjaamiseksi sekä korjauksen todentamiseksi.

- Kuvaus siitä, kuinka vika lopuksi ratkaistiin, ja suositukset korjauksen testaamiseksi (jos vika ratkaistiin ohjelmistoa muuttamalla)
- Muut viitteet, kuten esim. testitapaus, joka paljasti vian ja riskin, vaatimus tai muu vikaan liittyvä testauksen pohjamateriaalin osa (koskee dynaamista testausta).

On olemassa monia standardeja ja dokumentteja, kuten ISO 9126 [ISO 9126] (korvautuu ISO 25000:lla), IEEE 829 [IEEE 829], IEEE 1044 [IEEE 1044] ja Ortogonaalinen vikaluokittelu, jotka auttavat Testauspäällikköä päättämään, mitä tietoa vikojen raportoinnissa kerätään.

Mitä hyvänsä tietoa sitten katsotaankin vikaraporteissa tarpeelliseksi, on kriittistä että testaajat raportoivat tiedot täydellisesti, tiiviisti, täsmällisesti, objektiivisesti, tarpeeseen sopivasti ja oikea-aikaisesti. Vaikka henkilökohtainen kanssakäyminen ja kasvokkain tapahtuva viestintä voivat auttaa yksittäisen vian selvittämiseen liittyvissä vikaraporttia koskevissa ongelmissa, nämä ongelmat voivat aiheuttaa ylitsepääsemättömiä esteitä kunnolliselle projektin tilan, testauksen edistymisen ja prosessin kyvykkyyden arvioinnille.

## 4.4 Prosessin kyvykkyyden arviointi vikaraportin tietojen avulla

Kuten luvussa 2 on kerrottu, vikaraportit voivat olla hyödyllisiä projektin tilan seurannassa ja raportoinnissa. Metriikoiden merkitystä prosessien kannalta käsitellään pääasiassa Asiantuntijatason sertifikaattisisällön Testauksenhallinta-osiossa [ISTQB ETM SYL], mutta Jatkotasolla Testauspäällikön pitäisi tietää, mikä on vikaraporttien merkitys, kun arvioidaan testaus- ja ohjelmistokehitysprosessien kyvykkyyttä.

Luvussa 2 ja kappaleessa 4.3 mainittujen testauksen edistymisen seurannan tietojen lisäksi tarvitaan vikatietoja tukemaan prosessikehitysaloitteita. Esimerkkeihin kuuluvat:

- Käytetään vaihekohtaisesti vikojen synty-, löytämis- ja poistamisvaiheiden tietoja, kun arvioidaan vaiheen vikarajaustehokkuutta ja kun ehdotetaan tapoja vianlöytämistehokkuuden parantamiseksi kussakin vaiheessa.
- Käytetään vikojen syntyvaiheen tietoja Pareto-analyysin tekemiseksi niiden vaiheiden osalta, joissa suurin osa vioista syntyy, jotta voidaan tehdä kohdistettuja parannustoimenpiteitä vikojen kokonaismäärän vähentämiseksi.
- Käytetään vikojen alkuperäissyyanalyysin tietoja vikojen syntymisen taustalla olevien syiden määrittämisessä, jotta voidaan tehdä prosessinkehitystoimenpiteitä vikojen kokonaismäärän vähentämiseksi.
- Käytetään synty-, löytämis- ja poistamisvaiheiden tietoja laatukustannusanalyysin tekemisessä vikoihin liittyvien kustannusten minimoimiseksi.
- Käytetään vikojen komponenttitietoja vikojen kasaantumisanalyysin tekemiseksi, jotta ymmärretään tekniset riskit paremmin (riskipohjaisessa testauksessa), sekä ongelmallisten komponenttien uudelleensuunnittelemiseksi.

Metriikoiden käyttöä testausprosessin sisäisen ja kokonaistehokkuuden arvioinnissa käsitellään Asiantuntijatason sertifikaattisisällön Testauspäällikkö-osiossa [ISTQB ETM SYL].

Joissain tapauksissa tiimi voi päättää olla seuraamatta löydettyjä vikoja jonkin tai kaikkien ohjelmistokehityksen elinkaaren vaiheiden aikana. Vaikka näin tehdään usein tehokkuuden nimissä ja prosessin lisätyön vähentämiseksi, todellisuudessa se vähentää suuresti näkyvyyttä testaus- ja ohjelmistokehitysprosessien kyvykkyyteen. Tämä tekee edellä mainittujen parannusten toteuttamisen hankalaksi luotettavan tiedon puutteen vuoksi.

# 5. Testausprosessin kehittäminen – 135 min.

## Avainsanat

Capability Maturity Model Integration (CMMI), Critical Testing Processes (CTP), Systematic Test and Evaluation Process (STEP), Test Maturity Model integration (TMMi), TPI Next

## Oppimistavoitteet: Testausprosessin kehittäminen

### 5.2 Testauksen kehittämisprosessi

TM-5.2.1 (K2) Selittää esimerkkejä käyttämällä miksi testausprosessin kehittäminen on tärkeää

### 5.3 Testausprosessin kehittäminen

TM-5.3.1 (K3) Määritellä testausprosessin kehittämissuunnitelma IDEAL-mallia käyttämällä

### 5.4 Testausprosessin kehittäminen TMMi:tä käyttämällä

TM-5.4.1 (K2) Esitellä TMMi-testausprosessin kehittämismallin tausta, puitteet ja tavoitteet

### 5.5 Testausprosessin kehittäminen TPI Next:iä käyttämällä

TM-5.5.1 (K2) Esitellä TPI Next -testausprosessin kehittämismallin tausta, puitteet ja tavoitteet

### 5.6 Testausprosessin kehittäminen CTP:tä käyttämällä

TM-5.6.1 (K2) Esitellä CTP-testausprosessin kehittämismallin tausta, puitteet ja tavoitteet

### 5.7 Testausprosessin kehittäminen STEP:iä käyttämällä

TM-5.7.1 (K2) Esitellä STEP-testausprosessin kehittämismallin tausta, puitteet ja tavoitteet

## 5.1 Esittely

Organisaation yleisen testausprosessin vakiinnuttua sen pitäisi olla jatkuvan kehityksen alaisena. Tässä luvussa käydään läpi yleisiä kehittämiseen liittyviä asioita ja sen jälkeen esitellään joitakin malleja, joita voidaan käyttää testausprosessin kehittämiseen. Testauspäälliköiden on syytä lähteä ajatuksesta, että he ovat testausprosessin muutoksia ja kehitystä eteenpäin vievä voima, ja siksi heidän pitäisi tuntea alalla hyväksytyt tekniikat, joista tässä luvussa keskustellaan. Testausprosessin kehittämistä käsitellään tarkemmin Asiantuntijatason Testausprosessin kehittäminen –sertifikaattisisällössä.

## 5.2 Testauksen kehittämisprosessi

Aivan kuten organisaatiot käyttävät testausta parantamaan ohjelmistoa, ne voivat valita prosessien kehittämistekniikoita ja käyttää niitä parantamaan ohjelmistokehityksen prosesseja ja tuloksena syntyviä ohjelmistotuotteita. Prosessikehitystä voidaan soveltaa myös testausprosesseihin. Ohjelmistojen ja ohjelmistoja sisältävien järjestelmien testauksen kehittämiseen on olemassa erilaisia tapoja ja menetelmiä. Nämä menetelmät tähtäävät prosessin ja sen myötä syntyvien tuotosten parantamiseen tarjoamalla ohjeita ja tuomalla esiin kehitysalueita.

Testaus muodostaa usein merkittävän osan projektin kokonaiskustannuksista. Kuitenkin testausprosessin kehittäminen on otettu vain rajoitetusti huomioon eri ohjelmistokehityksen kehitysmalleissa, kuten CMMI®:ssä (ks. tarkemmin alla).

Testauksen kehitysmallit kuten Test Maturity Model Integration (TMMi®), Systematic Test and Evaluation Process (STEP), Critical Testing Processes (CTP) ja TPI Next® kehitettiin paikkaamaan useimmissa ohjelmistokehitysmalleissa esiintyvää puutteellista testauksen huomioonottamista. Oikein käytettynä nämä mallit voivat tarjota joukon läpi organisaation ulottuvia mittaritietoja, joita voidaan käyttää lähtötilanteen vertailukohtina.

Tässä sertifikaattisisällössä esitettyjä malleja ei ole tarkoitettu käyttösuosituksiksi, vaan niiden avulla pyritään tarjoamaan kattava näkymä siihen, miten mallit toimivat ja mitä ne sisältävät.

### 5.2.1 Prosessikehityksen taustaa

Prosessien kehittäminen on tärkeää niin ohjelmistokehitysprosesseissa kuin testausprosesseissa. Omista virheistä oppimalla on mahdollista parantaa prosessia, jota organisaatiot käyttävät ohjelmiston kehittämiseen ja testaamiseen. Demingin kehityssykliä “Suunnittele, Tee, Tarkista, Korjaa” (PDCA, ”Plan, Do, Check, Act”) on käytetty vuosikymmeniä, ja se on yhä käyttökelpoinen, kun testaajien täytyy parantaa tänä päivänä käytössä olevaa prosessia.

Yksi peruste prosessikehitykselle on usko siihen, että ohjelmiston kehityksessä käytetyn prosessin laatu vaikuttaa suuresti järjestelmän laatuun. Ohjelmistotuotannon parantunut laatu vähentää ohjelmiston ylläpidossa tarvittavia resursseja ja suo näin jatkossa enemmän aikaa kehittää lisää parempia ratkaisuja. Prosessimallit tarjoavat paikan, josta kehityksen voi aloittaa, mittaamalla organisaation prosessien kyvykkyyden mallia vastaan. Mallit tarjoavat myös rungon, jonka avulla organisaation prosesseja voi arvion pohjalta saatujen tulosten perusteella lähteä kehittämään.

Prosessiarviointi johtaa prosessin kyvykkyyden määrittelyyn, mikä kannustaa prosessien kehittämiseen. Tämä voi myös johtaa prosessien jatkoarviointeihin parannusten vaikutusten mittaamiseksi.

### 5.2.2 Prosessikehityksen tyypit

Arviointimallien käyttäminen on tyypillinen menetelmä, jolla varmistetaan testattuja ja luotettavia käytäntöjä noudattamalla standardoitu lähestymistapa testausprosessien kehittämiseen.

Prosessien kehitysmallit jaetaan kahteen tyyppiin:
1. Arviointimalli, joka tuottaa osana arviointia kypsyysmittauksen, jolla voidaan arvioida organisaation kyvykkyyttä malliin verrattuna sekä organisaatiota yleensä mallin sisällä ja joka tarjoaa suunnan prosessin parantamiselle.

2. Tilannemalli, joka tuottaa organisaation kehittymismahdollisuuksista liiketoimintavetoisen arvion, johon kuuluu joissakin tapauksissa organisaation arvioiminen teollisuuden keskiarvoja vasten objektiivisia mittauskeinoja käyttämällä. Tätä arviointia voidaan käyttää prosessien kehittämissuunnitelman luomiseen.

Testausprosessin kehittäminen voidaan saada myös aikaan ilman malleja käyttämällä esimerkiksi analyyttisiä lähestymistapoja ja jälkipalavereita.

## 5.3 Testausprosessin kehittäminen

IT-ala voi käyttää testausprosessin kehittämismalleja korkeamman kypsyystason ja ammattimaisuuden saavuttamiseksi. Teollisuudenalan standardimallit auttavat kehittämään läpi organisaation käytettäviä metriikoita ja mittareita, joita voidaan käyttää vertailussa. Testausalalta nousseen prosessien kehitystarpeen pohjalta on syntynyt useita suositeltuja prosesseja. Näihin kuuluvat STEP, TMMi, TPI Next ja CTP. Vaiheittaiset mallit, kuten TMMi ja CMMI, tarjoavat standardit käytettäväksi eri yritysten ja organisaatioiden välisessä vertailussa. Jatkuvat mallit, kuten CTP, STEP ja TPI Next, luovat organisaatiolle mahdollisuuden puuttua sen korkeimman prioriteetin ongelmiin ja antavat enemmän vapautta toteutusjärjestykseen. Näitä jokaista käsitellään edempänä tässä kappaleessa.

Kaikki nämä mallit suovat organisaatiolle mahdollisuuden määritellä, mikä sen tilanne on senhetkisen testausprosessin osalta. Kun arviointi on tehty, TMMi ja TPI Next esittävät ehdotuksen testausprosessin kehittämispoluksi. Vaihtoehtoisesti STEP ja CTP antavat organisaatiolle keinot määritellä, mistä se saa suurimman tuoton sijoitukselleen prosessikehitykseen ja jättää sopivasta etenemispolusta päättämisen organisaation tehtäväksi.

Kun on päätetty, että testausprosessi pitää katselmoida ja sitä pitää parantaa, tässä tehtävässä käytettävät prosessikehityksen toteutusaskeleet voivat noudattaa esimerkiksi IDEAL$^{SM}$-mallissa [IDEAL96] määriteltyjä vaiheita:

- Kehitysprosessin käynnistäminen (Initiating)
- Tämänhetkisen tilanteen arviointi (Diagnosing)
- Testausprosessin kehittämissuunnitelman laatiminen (Establishing)
- Kehitystoimenpiteiden toteuttaminen (Acting)
- Kehitysohjelmasta oppiminen (Learning).

**Kehitysprosessin käynnistäminen (Initiating)**
Ennen kuin prosessikehityksen tehtävät alkavat, sidosryhmien on sovittava prosessikehityksen tavoitteista, tähtäimestä, laajuudesta ja kattavuudesta. Myös prosessikehitysmallin valinta on tehtävä tässä vaiheessa. Malli voidaan valita joko julkisesti saatavilla olevista vaihtoehdoista (kuten CTP, STEP, TMMi ja TPI Next) tai se voidaan kehittää sisäisesti. Lisäksi on määriteltävä onnistumiskriteerit ja menetelmä, jolla niitä mitataan läpi kehitystehtävien.

**Tämänhetkisen tilanteen arviointi (Diagnosing)**
Sovittua arviointitapaa käytetään ja sen avulla luodaan testauksen arviointiraportti, joka sisältää arvion nykyisistä testauskäytännöistä ja listan mahdollisista prosessin parannuksista.

**Testausprosessin kehittämissuunnitelman laatiminen (Establishing)**
Mahdollisten prosessin parannusten lista priorisoidaan. Priorisointi voi perustua sijoitukselle saatavaan tuottoon, riskeihin, organisaation strategioihin soveltuvuuteen ja/tai mitattaviin määrällisiin tai laadullisiin hyötyihin. Kun järjestys on sovittu, laaditaan suunnitelma parannusten toteuttamiseksi.

**Kehitystoimenpiteiden toteuttaminen (Acting)**
Testausprosessin kehitystoimenpiteiden toteutussuunnitelma laitetaan käytäntöön. Tähän voi kuulua tarvittavan koulutuksen tai mentoroinnin järjestäminen, prosessien pilotointi ja lopulta niiden täysimittainen käyttöönotto.

**Kehitysohjelmasta oppiminen (Learning).**

Kun prosessin parannukset on kokonaan tehty, on oleellisen tärkeää todentaa, mitä hyötyjä (sekä niitä, jotka määriteltiin aikaisemmin, että mahdollisesti odottamattomia hyötyjä) saatiin aikaan. On myös tärkeä tarkistaa, mitkä prosessikehityksen tehtävien onnistumiskriteereistä on täytetty.

Käytetystä prosessimallista riippuen tämä on prosessin vaihe, jossa seuraavan kypsyystason seuranta alkaa ja tehdään päätös, aloitetaanko kehitysprosessi uudelleen vai lopetetaanko tehtävät tässä vaiheessa.

## 5.4 Testausprosessin kehittäminen TMMi:tä käyttämällä

Testing Maturity Model Integration (TMMi) muodostuu viidestä kypsyystasosta ja se on tarkoitettu täydentämään CMMI:tä. Jokainen kypsyystaso sisältää määritellyt prosessialueet, joiden erityis- ja yleistavoitteiden täytyy täyttyä 85-prosenttisesti, ennen kuin organisaatio voi edetä seuraavalle tasolle.

TMMi:n kypsyystasot ovat:

- Taso I: Lähtötaso
  Lähtötaso edustaa tilaa, jossa ei ole muodollisesti dokumentoitua tai jäsennettyä testausprosessia. Testit kehitetään tyypillisesti ad hoc –tyylillä toteutuksen jälkeen, ja testauksen katsotaan tarkoittavan samaa kuin debuggaus. Testauksen tavoitteena pidetään ohjelmiston toimivuuden todistamista.
- Taso 2: Hallittu
  Toinen taso saavutetaan, kun testausprosessit on selkeästi erotettu virheiden jäljityksestä (debuggauksesta). Tämä voidaan saavuttaa määrittämällä testauspolitiikka ja tavoitteet, esittelemällä perustestausprosessin sisältämät vaiheet (esim. testauksen suunnittelu) ja ottamalla käyttöön perustestaustekniikat ja –menetelmät.
- Taso 3: Määritelty
  Kolmas taso saavutetaan, kun testausprosessit integroidaan ohjelmistokehityksen elinkaareen ja dokumentoidaan muodollisina standardeina, toimintatapoina ja menetelminä. Katselmointeja käytetään ja ohjelmistotestauksen pitäisi olla selkeä toiminto, jota voidaan valvoa ja seurata.
- Taso 4: Mitattu
  Taso neljä saavutetaan, kun testausprosessia voidaan mitata ja hallita tehokkaasti organisaatiotasolla määrättyjen projektien tukemiseksi.
- Taso 5: Optimoitu
  Viimeinen taso edustaa testausprosessin kypsyyden tilaa, jossa testausprosessista saatuja tietoja voidaan käyttää vikojen ehkäisemiseen ja painopiste on vakiintuneiden prosessien optimoinnissa.

Lisätietoja TMMi:stä: Ks. [vanVeenendaal11] ja [www.tmmi.org].

## 5.5 Testausprosessin kehittäminen TPI Next:iä käyttämällä

TPI Next –malli määrittelee 16 avainaluetta, joista jokainen kattaa testausprosessin määrätyn alueen, kuten testausstrategian, mittarit, testaustyökalut ja testiympäristön.

Mallissa on määritelty neljä kypsyystasoa:

- Lähtötaso
- Hallittu
- Tuottava
- Optimoitu.

Jokaisen avainalueen arviointiin jokaisella kypsyystasolla on määritelty tarkastuspisteet. Löydökset vedetään yhteen ja ne kuvataan visuaalisesti kypsyysmatriisin avulla, joka kattaa kaikki avainalueet. Kehitystavoitteiden määrittely ja toteutus voidaan räätälöidä testausorganisaation tarpeiden ja kyvykkyyden mukaan.

Geneerinen lähestymistapa tekee TPI Nextistä mistään ohjelmistoprosessien kehitysmalleista riippumattoman. Se kattaa sekä testauksen teknisen näkökulman että hallinnollisen päätöksenteon tuen [deVries09].

Lisätietoja TPI Nextistä: Ks. [www.tpinext.com].

## 5.6 Testausprosessin kehittäminen CTP:tä käyttämällä

Critical Testing Processes (CTP) –arviointimallin perusajatus on, että tietyt testausprosessit ovat kriittisiä. Nämä kriittiset prosessit tukevat menestyksekkäitä testaustiimejä, jos ne toteutetaan hyvin. Vastaavasti, mikäli nämä tehtävät suoritetaan huonosti, jopa lahjakkaiden yksittäisten testaajien ja Testauspäälliköiden menestyminen on epätodennäköistä. Malli määrittelee 12 kriittistä testausprosessia. CTP on pääasiassa tilannemalli.

CTP-malli on tilanneriippuvainen lähestymistapa, joka sallii mallin räätälöimisen mukaan lukien

- yksittäisten haasteiden tunnistaminen
- hyvien prosessin ominaisuuksien tunnistaminen
- prosessin parannustoimenpiteiden järjestyksen ja toteutuksen tärkeyden valinta.

CTP-malli mukautuu kaikkiin ohjelmistokehityksen elinkaarimalleihin.

Osallistujahaastatteluiden lisäksi CTP-mallissa käytetään mittareita, joiden avulla määritellään organisaatioiden taso verrattuna teollisuudenalan keskiarvoihin ja parhaisiin käytäntöihin.

Lisätietoja CTP:stä: Ks. [Black03].

## 5.7 Testausprosessin kehittäminen STEP:iä käyttämällä

Samoin kuin CTP mutta toisin kuin TMMi ja TPI Next, STEP (Systematic Test and Evaluation Process) ei edellytä, että parannustoimenpiteet tapahtuvat määrätyssä järjestyksessä.

STEP on ensisijassa tilannemalli, joka perustuu ajatukseen, että testaus on ohjelmiston elinkaaren tehtävä, joka alkaa vaatimusten määrittelyn aikana ja jatkuu järjestelmän käytöstä poistumiseen saakka. STEP-menetelmä korostaa “ensin testi, sitten koodi” –lähestymistapaa käyttämällä vaatimuspohjaista testausstrategiaa sen varmistamiseksi, että aikainen testitapausten luominen kelpuuttaa vaatimusmäärittelyt ennen suunnittelua ja toteutusta.

Menetelmän perusajatuksiin kuuluvat seuraavat:

- Vaatimuspohjainen testausstrategia
- Testaus alkaa elinkaaren alussa
- Testejä käytetään vaatimuksina ja käyttömalleina
- Testimateriaalin suunnittelu ohjaa ohjelmistosuunnittelua
- Viat löydetään aikaisemmin tai ne ehkäistään kokonaan
- Viat analysoidaan järjestelmällisesti
- Testaajat ja toteuttajat työskentelevät yhdessä.

Joissakin tapauksissa STEP-arviointimalli yhdistetään TPI Next-kypsyysmalliin.

Lisätietoja STEP:stä: Ks. [Craig02].

# 6. Testaustyökalut ja automaatio – 135 min.

**Avainsanat**
avoimen lähdekoodin työkalu, räätälöity työkalu

**Oppimistavoitteet: Testaustyökalut ja automaatio**

## 6.2 Työkalun valinta

TM-6.2.1 (K2) Kuvata avoimen lähdekoodin työkalun valintaan liittyviä hallinnollisia ongelmia.
TM-6.2.2 (K2) Kuvata räätälöidystä työkalusta päätettäessä esiin nousevia hallinnollisia ongelmia.
TM-6.2.3 (K4) Arvioida annettu tilanne ja laatia sen perusteella suunnitelma työkalun valitsemiseksi, mukaan luettuna riskit, kustannukset ja hyödyt.

## 6.3 Työkalun elinkaari

TM-6.3.1 (K2) Selittää työkalun elinkaaren eri vaiheet.

## 6.4 Työkalumetriikat

TM-6.4.1 (K2) Kuvata, kuinka työkalut voivat parantaa metriikoiden keräämistä ja arviointia.

## 6.1 Esittely

Tässä luvussa laajennetaan Perustason sertifikaattisisältöä käsittelemällä useita yleisiä seikkoja, joita Testauspäällikön on otettava huomioon työkalujen ja automaation suhteen.

## 6.2 Työkalun valinta

Valitessaan testaustyökalua Testauspäällikön on harkittava monia erilaisia asioita.

Perinteisesti tyypillisin valinta on ostaa työkalu kaupalliselta työkalutoimittajalta. Joissain tapauksissa tämä saattaa olla ainoa mahdollinen vaihtoehto. On kuitenkin olemassa muitakin vaihtoehtoja, kuten avoimen lähdekoodin työkalut ja räätälöidyt työkalut, jotka ovat myös harkittavissa olevia vaihtoehtoja.

Riippumatta työkalun tyypistä Testauspäällikön täytyy kustannus-tuotto-analyysi tekemällä selvittää huolellisesti työkalun omistamisen kokonaiskustannukset työkalun odotettuna käyttöaikana. Tätä asiaa käsitellään alempana kohdassa “Sijoitukselle saatava tuotto (ROI)”.

### 6.2.1 Avoimen lähdekoodin työkalut

Avoimen lähdekoodin työkaluja on saatavilla miltei kaikkiin testausprosessin osa-alueisiin, testitapausten hallinnasta vikojen seurantaan ja testitapausten automatisointiin, vain muutamia mainitaksemme. Tärkeä piirre avoimen lähdekoodin työkaluissa on, että vaikka työkaluun ei yleensä liity suuria hankkimiskustannuksia, työkalulle ei välttämättä ole saatavilla muodollista tukea. Monilla avoimen lähdekoodin työkaluilla on kuitenkin uskollinen seuraajakunta, joka on halukas tukemaan käyttäjiä ei-perinteisellä tai epämuodollisella tavalla.

Tämän lisäksi monet avoimen lähdekoodin työkalut suunniteltiin alun perin ratkaisemaan jokin tietty ongelma tai ne kohdistuivat yksittäiseen asiaan; siksi työkalu ei välttämättä sisällä kaikkia toiminnallisuuksia, jotka ovat saatavilla vastaavassa työkalutoimittajan työkalussa. Siksi testausryhmän todellisten tarpeiden huolellinen analysointi on tehtävä ennen kuin päädytään valitsemaan avoimen lähdekoodin työkalu.

Eräs avoimen lähdekoodin työkalujen hyöty on, että käyttäjät pystyvät usein muokkaamaan tai laajentamaan niitä. Jos testausorganisaatiolta löytyy ydinosaamista, työkalu voidaan muokata toimimaan muiden työkalujen kanssa tai sitä voidaan muuttaa vastamaan testaustiimin tarpeita. Useita työkaluja voidaan yhdistää ratkaisemaan ongelmia, joihin työkalutoimittajien työkaluja ei voi käyttää. Tietenkin, mitä enemmän työkaluja käytetään ja mitä enemmän muutoksia tehdään, sitä enemmän monimutkaisuutta ja lisäkustannuksia syntyy. Testauspäällikön pitää varmistaa, että tiimi ei ala käyttää avoimen lähdekoodin työkaluja vain niiden käytön vuoksi; kuten muidenkin työkalujen kohdalla, työpanos täytyy aina kohdistaa positiivisen tuoton aikaansaamiseen.

Testauspäällikön täytyy ymmärtää valittuun työkaluun liittyvät lisensiointiohjelmat. Moniin avoimen lähdekoodin työkaluihin liittyy GNU General Public Licencen jokin versio, joka määrittelee, että ohjelmiston jakelu täytyy aina tapahtua samoilla ehdoilla kuin millä ohjelma on hankittu. Jos testaustiimi tekee työkaluun muutoksia, jotta työkalu tukee paremmin heidän tekemäänsä testausta, nämä muutokset on lisenssisopimuksen perusteella ehkä tarjottava kaikkien työkalun ulkopuolisten käyttäjien saataville. Testauspäällikön on tarkistettava mahdolliset oikeudelliset seuraamukset, jotka liittyvät ohjelmiston jakeluun heidän organisaatiossaan.

Turvallisuuskriittisiä tai tehtäväkriittisiä ohjelmistoja kehittävillä organisaatioilla tai organisaatioilla, joiden toimintaa säätelevät erilaiset säädökset, voi olla ongelmia avoimen lähdekoodin työkalujen käytössä. Vaikka monet avoimen lähdekoodin työkalut ovat erittäin korkealaatuisia, minkään avoimen lähdekoodin työkalun oikeaa toimintaa ei todennäköisesti ole sertifioitu. Työkalutoimittajien työkalut on usein sertifioitu niiden oikean toimivuuden ja tiettyyn tehtävään soveltuvuuden suhteen (esim. DO-178B). Vaikka avoimen lähdekoodin työkalu voi olla aivan yhtä hyvä, sertifiointi voi olla työkalua käyttävän ryhmän vastuulla ja aiheuttaa sille lisäkustannuksia.

### 6.2.2 Räätälöidyt työkalut

Testausorganisaatio voi todeta, että heillä on erityistarve, johon ei ole saatavissa työkalutoimittajien toimittamaa tai avoimen lähdekoodin työkalua. Syynä voi olla yksinoikeudella tehty laitteistoalusta, räätälöity ympäristö tai prosessi, jota on muokattu poikkeuksellisella tavalla. Tällaisissa tapauksissa Testauspäällikkö voi harkita räätälöidyn työkalun kehittämistä, mikäli tiimistä löytyy siihen ydinosaamista.

Räätälöidyn työkalun kehittämisen etuihin kuuluu, että työkalu vastaa täsmälleen tiimin tarpeita ja voi toimia tuottavasti testaustiimin vaatimassa tilanteessa. Työkalu voidaan laatia niin, että sillä on liittymät toisiin käytössä oleviin työkaluihin, ja että se luo aineistoa täsmälleen tiimin tarvitsemassa muodossa. Tämän lisäksi työkalulla voi olla soveltamismahdollisuuksia organisaatiossa alkuperäisen projektin ulkopuolella. Ennen kuin työkalu julkaistaan muiden projektien käyttöön on kuitenkin tärkeää ensin katselmoida sen käyttötarkoitus, tavoite, hyödyt ja mahdolliset haitat.

Räätälöidyn työkalun laatimista harkitsevan Testauspäällikön on myös harkittava mahdollisia negatiivisia puolia. Räätälöidyt työkalut ovat usein riippuvaisia työkalun laatineesta henkilöstä. Siksi räätälöidyt työkalut on dokumentoitava riittävän hyvin, jotta myös muut voivat ylläpitää niitä. Jos tätä ei voi tehdä, ne voivat jäädä “orvoiksi” ja niiden käyttö voi lakata, kun työkalun laatija lähtee projektista. Ajan myötä räätälöityjen työkalujen käyttötarkoitusta voidaan laajentaa niiden alkuperäisen tarkoituksen ulkopuolelle, mikä voi aiheuttaa työkalun laatuongelmia, jotka johtavat vääriin positiivisiin havaintoraportteihin tai vääränlaisen aineiston luomiseen. Testauspäällikön on pidettävä mielessä, että räätälöity työkalu on sekin vain ohjelmistotuote, ja sellaisena siihen kohdistuu samoja toteutusongelmia kuin mihin tahansa muihinkin ohjelmistotuotteisiin. Räätälöidyt työkalut pitää suunnitella ja testata sen varmistamiseksi, että ne toimivat odotetulla tavalla.

### 6.2.3 Sijoitetun pääoman tuotto (Return on Investment - ROI)

Testauspäällikön vastuulla on varmistaa, että kaikki testausorganisaatiossa käyttöön otetut työkalut tuovat lisäarvoa tiimin työhön ja voivat osoittaa positiivista sijoituksen tuottoa (ROI) organisaatiolle. Ennen työkalun hankkimista tai laatimista pitäisi suorittaa panos-tuotos-analyysi sen varmistamiseksi, että työkalulla saavutetaan todellisia ja pitkäaikaisia hyötyjä. Analyysissä ROI:ssa pitäisi ottaa huomioon toistuvat ja kertaluontoiset kustannukset, joista osa on rahallisia ja osa liittyy resursseihin tai aikaan, sekä riskit, jotka voivat vähentää työkalun arvoa.

Kertaluontoisiin kustannuksiin kuuluvat seuraavat:

- Työkaluvaatimusten määrittäminen tavoitteiden ja päämäärien saavuttamiseksi
- Oikean työkalun ja työkalutoimittajan arviointi ja valinta
- Työkalun hankinta, mukauttaminen tai kehittäminen
- Työkaluun liittyvän peruskoulutuksen toteuttaminen
- Työkalun integroiminen muihin työkaluihin
- Työkalun tukemiseksi tarvittavan laitteiston/ohjelmiston hankkiminen.

Toistuviin kustannuksiin kuuluvat seuraavat:

- Työkalun omistaminen
    - Lisenssi- ja tukikustannukset
    - Itse työkaluun liittyvät ylläpitokustannukset
    - Työkalun tuottaman materiaalin ylläpitokustannukset
    - Jatkuvan koulutuksen ja mentoroinnin kustannukset
- Työkalun siirtäminen eri ympäristöihin
- Työkalun sopeuttaminen tuleviin tarpeisiin
- Laadun ja prosessien parantaminen valittujen työkalujen optimaalisen käytön varmistamiseksi.

Testauspäällikön pitää myös pohtia kaikkiin työkaluihin liittyvien vaihtoehtojen kustannuksia. Työkalun hankintaan, hallinnointiin, koulutukseen ja käyttöön liittyvä aika olisi voitu käyttää varsinaisiin testaustehtäviin; siksi alkuvaiheessa voidaan tarvita enemmän testausresursseja, kunnes työkalua aletaan käyttää.

Työkalun käyttöön liittyy monia riskejä; kaikki työkalut eivät itse asiassa tuota hyötyjä, jotka korvaisivat niihin liittyvät riskit. Työkaluihin liittyviä riskejä käsiteltiin Perustason sertifikaattisisällössä. Testauspäällikön pitäisi lisäksi harkita seuraavia riskejä, kun hän miettii ROI:ta:

- Organisaation epäkypsyys (se ei ole valmis käyttämään työkalua)
- Työkalun tuottaman materiaalin ylläpito voi olla vaikeaa ja edellyttää useita muutoksia, kun testattavaa ohjelmistoa muutetaan.
- Kun Testausasiantuntija osallistuu vähemmän testaustehtäviin, se voi heikentää testauksen arvoa (esim. vikojen löytötehokkuus voi pienentyä, kun suoritetaan pelkästään automatisoituja skriptejä).

Lopuksi Testauspäällikön on tarkasteltava työkalun käytöstä mahdollisesti saatavia hyötyjä. Työkalun käyttöönottoon ja käyttöön liittyviin hyötyihin kuuluvat mm. seuraavat:

- Toistuvan työn väheneminen
- Testauskierrokseen kuluvan ajan väheneminen (esim. käyttämällä automatisoituja regressiotestejä)
- Testauksen suorituskustannusten väheneminen
- Määrätyn tyyppisen testauksen lisääntyminen (esim. regressiotestaus)
- Inhimillisten virheiden väheneminen testauksen eri vaiheissa. Esimerkkeihin kuuluvat:
- Testiaineistosta voi tulla kelvollisempaa testiaineiston generointityökaluilla
- Testitulosten vertailu voi olla tarkempaa vertailutyökaluja käyttämällä
- Testiaineiston syöttäminen voi tapahtua oikeellisemmin skriptaustyökaluja käyttämällä.
- Testeihin liittyvän tiedon saamiseen tarvittava työmäärä vähenee. Esimerkkeihin kuuluvat:
- Työkaluilla tuotetut raportit ja metriikat
- Testimateriaalin uudelleenkäyttö (esim. testitapaukset, testiskriptit ja testiaineisto)
- Sellaisen testauksen lisääntyminen, joka ei olisi mahdollista ilman työkaluja (esim. suorituskykytestit, kuormitustestit)
- Automaatiota toteuttavien testaajien ja koko testausorganisaation arvostuksen nousu, kun he osoittavat ymmärtävänsä erikoistyökaluja ja näyttävät, kuinka niitä käytetään.

Yleensä testausryhmä harvoin käyttää vain yksittäistä työkalua; testaustiimin saavuttama koko ROI muodostuu yleensä kaikkien käytössä olevien työkalujen kokonaisuudesta. Työkalujen täytyy pystyä jakamaan tietoa ja toimimaan yhteistyössä toistensa kanssa. Kattavan, pitkän tähtäimen testaustyökalustrategian laatiminen on suositeltavaa.

### 6.2.4 Valintaprosessi

Testaustyökalut ovat pitkän tähtäimen sijoitus, joka todennäköisesti ulottuu yli yksittäisen projektin monien iteraatioiden ja/tai koskee monia projekteja. Testauspäällikön on harkittava mahdollista työkalua useista näkökulmista.

- Liiketoiminnan kannalta vaaditaan positiivista ROI:ta. Jotta organisaatio saa sijoitukselleen korkean tuoton, sen pitää varmistaa, että työkalut, joilta vaaditaan keskinäistä yhteistoimintaa – mukaan lukien sekä testaustyökalut että ei-testaustyökalut – toimivat yhdessä. Joissain tapauksissa tämän yhteentoimivuuden saavuttamiseksi on kehitettävä prosessit ja yhteenliittymät, ja niiden toteuttaminen voi viedä aikaa.
- Projektin näkökulmasta työkalun on oltava tehokas (esim. vältetään manuaalitestauksen aikaiset virheet, kuten esim. tietojen syöttämisessä tapahtuvat kirjoitusvirheet). Työkalu voi vaatia huomattavasti aikaa ennen kuin se alkaa tuottaa positiivista ROI:ta. Monissa tapauksissa sijoitus voi alkaa tuottaa vasta toisen julkaisukierroksen tai ylläpidon aikana ennemmin kuin heti ensimmäisessä projektissa, jossa automaatio otettiin käyttöön. Testauspäällikön on otettava huomioon sovelluksen koko elinkaari.
- Työkalua käyttävän henkilön näkökulmasta työkalun pitää tukea projektin jäseniä tekemällä heidän tehtävänsä tehokkaasti ja tuottavasti. Jotta käyttäjät oppivat työkalun käytön nopeasti ja mahdollisimman helposti, on pohdittava oppimiskäyrää. Kun testaustyökalu otetaan käyttöön, sen käyttäjät tarvitsevat koulutusta ja mentorointia.

Sen varmistamiseksi, että kaikki näkökulmat on otettu huomioon, on tärkeää laatia askelittainen suunnitelma testaustyökalun käyttöönottoa varten.

Testaustyökalun valintaprosessi esiteltiin jo Perustason sertifikaattisisällössä seuraavasti:

- Organisaation kypsyyden arviointi
- Työkaluvaatimusten tunnistaminen
- Työkalujen arviointi
- Työkalutoimittajan tai tukipalveluiden (avoimen lähdekoodin työkalut, räätälöidyt työkalut) arviointi
- Työkalun käytön tuomien sisäiset valmennus- ja mentorointivaatimusten tunnistaminen
- Koulutustarpeiden arviointi ottaen huomioon nykyisen testaustiimin automatisointitaidot
- Kustannusten ja tuottojen arviointi (kuten esitetty kappaleessa 6.2.3 ROI).

Riippumatta testausvaiheesta, jossa työkalua tullaan käyttämään, Testauspäällikön on jokaisen työkalutyypin osalta arvioitava alla lueteltuja seikkoja:

- Analysointi
  - Pystyykö työkalu “ymmärtämään” sille annetun syötteen?
  - Sopiiko työkalu sille suunniteltuun käyttöön?
- Suunnittelu
  - Auttaako työkalu testimateriaalin laatimisessa olemassa olevan tiedon perusteella (esim. testisuunnittelutyökalut, jotka laativat testitapauksia vaatimuksista)?
  - Voidaanko suunnitelmat luoda automaattisesti?
  - Voidaanko varsinainen testimateriaalin koodi luoda kokonaan tai osittain ylläpidettävässä ja käytettävässä muodossa?
  - Voidaanko tarvittava testiaineisto luoda automaattisesti (esim. koodin analyysin pohjalta luotava aineisto)?
- Aineisto ja testien valinta
  - Miten työkalu valitsee tarvitsemansa aineiston (esim. mitkä testitapaukset suoritetaan milläkin aineistojoukolla)?
  - Hyväksyykö työkalu valintakriteerit, jotka annetaan joko manuaalisesti tai automaattisesti?
  - Pystyykö työkalu määrittelemään, kuinka tuotantoaineisto “putsataan” määrättyjen syötteiden perusteella?
  - Pystyykö työkalu päättelemään, mitä testejä tarvitaan kattavuuskriteerien perusteella (esim. jos annetaan joukko vaatimuksia, pystyykö työkalu seuraamaan jäljitettävyyttä ja päättelemään, mitkä testit on suoritettava)?
- Suoritus
  - Toimiiko työkalu automaattisesti vai tarvitaanko manuaalisia välitoimenpiteitä?
  - Miten työkalu pysäytetään ja käynnistetään uudelleen?
  - Pitääkö työkalun pystyä “kuuntelemaan” keskeisiä tapahtumia (esim. pitääkö testauksen hallintatyökalun automaattisesti päivittää testitapauksen tila, jos kyseiseen testitapaukseen liittyvä vikaraportti suljetaan)?
- Arviointi
  - Miten työkalu päättelee, onko lopputulos kunnollinen (esim. työkalu käyttää testioraakkelia vastauksen oikeellisuuden päättelemiseen)?
  - Minkälaiset virhetilanteesta toipumisominaisuudet työkalussa on?
  - Tarjoaako työkalu riittävät loki- ja raportointiominaisuudet?

## 6.3 Työkalun elinkaari

Työkalun hyödyllisessä elinkaaressa on neljä seuraavassa kuvattua vaihetta, joita Testauspäällikön täytyy hallinnoida.

1. Hankinta. Työkalu täytyy hankkia yllä kuvatun mukaisesti (ks. luku 6.2 Työkalun valinta). Sen jälkeen, kun päätös työkalun hankkimisesta on tehty, Testauspäällikön pitää nimittää joku (usein Testausasiantuntija tai Tekninen testausasiantuntija) toimimaan työkalun järjestelmävalvojana. Tämän henkilön tulee tehdä päätökset siitä, kuinka ja milloin työkalua käytetään, mihin syntyvät tuotokset varastoidaan, mitä nimeämiskäytäntöjä käytetään jne. Tekemällä nämä päätökset etukäteen sen sijaan, että annettaisiin niiden syntyä tilanteen sitä vaatiessa

voi saada aikaan merkittävän eron työkalun lopullisessa tuotossa sijoitukseen nähden. Työkalun käyttäjille on todennäköisesti järjestettävä koulutusta.

2. Tuki ja ylläpito. Työkalun tukeen ja ylläpitoon tarvitaan jatkuvasti seurattavat prosessit. Työkalun ylläpitovastuu saattaa kuulua työkalun järjestelmävalvojalle tai se voidaan antaa erikoistuneelle työkaluryhmälle. Jos työkalun tulee toimia yhdessä muiden työkalujen kanssa, on otettava huomioon tiedon siirrettävyys ja yhteistyöprosessit. Työkalun sekä sen tuottamien tulosten varmuuskopiointi ja tietojen palautus on myös otettava huomioon.
3. Kehitys. Yksi suunniteltavista asioista on konversio. Ajan kuluessa ympäristö, liiketoiminnan tarpeet tai työkalutoimittajaan liittyvät seikat voivat aiheuttaa tarpeen suuriin muutoksiin työkalussa tai sen käytössä. Esimerkiksi työkalutoimittaja voi vaatia työkaluun päivitystä, joka aiheuttaa ongelmia muiden yhdessä toimivien työkalujen kanssa. Liiketoimintasyistä ympäristöön tarvittava muutos voi aiheuttaa ongelmia työkalun kanssa. Mitä monimutkaisempi työkalun käyttöympäristö on, sitä enemmän jokin muutostoimenpide voi aiheuttaa häiriöitä työkalun käyttöön. Riippuen siitä, kuinka suuri merkitys työkalulla on testauksessa, Testauspäällikkö voi tässä vaiheessa joutua varmistamaan, että organisaatiolla on keinot taata palveluiden jatkuvuus.
4. Käytöstä poistuminen. Jossain vaiheessa tulee hetki, jolloin työkalun käyttöikä on kulunut loppuun. Tässä vaiheessa työkalu täytyy poistaa hallitusti käytöstä. Työkalun tarjoama toiminnallisuus täytyy korvata ja sen sisältämä aineisto täytyy tallentaa ja arkistoida. Näin voi tapahtua siksi, että työkalu on tullut elinkaarensa päähän, tai pelkästään siksi, että on tultu siihen pisteeseen, että uuteen työkaluun vaihtamisen hyödyt ja sen tarjoamat mahdollisuudet ylittävät kustannukset ja riskit.

Testauspäällikön vastuulla on koko työkalun elinkaaren ajan varmistaa työkalun testaustiimille tuottamien palveluiden ongelmaton toiminta ja hallittu jatkuminen.

## 6.4 Työkalumetriikat

Testauspäällikkö voi suunnitella ja kerätä objektiivisia mittaritietoja Teknisen testausasiantuntijan ja Testausasiantuntijan käyttämistä työkaluista. Erilaiset työkalut voivat tallentaa arvokasta reaaliaikaista tietoa ja vähentää tiedon keruussa tarvittavaa työmäärää. Testauspäällikkö voi käyttää näitä tietoja testauksen kokonaishallintaan.

Erilaiset työkalut keskittyvät eri tyyppisten tietojen keräämiseen. Esimerkkeinä voidaan mainita seuraavat:

- Testauksen hallintatyökalut voivat tuottaa joukon erilaisia mittaritietoja. Jäljitettävyys vaatimuksista testitapauksiin ja automatisoituihin skripteihin mahdollistaa kattavuustietojen saamisen. Milloin tahansa voidaan tuottaa tilannekatsaus sillä hetkellä saatavilla oleviin testeihin, suunniteltuihin testeihin ja senhetkiseen suoritustilanteeseen (läpäisty, hylätty, ohitettu, estetty, jonossa).
- Havaintojenhallintatyökalut voivat tuottaa suuren määrän tietoa vioista, kuten sen hetkinen tilanne, vakavuus ja kiireellisyys, jakautuminen järjestelmän läpi jne. Muut lisätiedot, kuten vikojen aiheuttamis- ja löytövaihe sekä läpipääsymäärä auttavat Testauspäällikköä ohjaamaan prosessin kehitystä.
- Staattisen analyysin työkalut auttavat löytämään ylläpidettävyysongelmia ja raportoimaan niistä.
- Suorituskykytyökalut voivat tuottaa arvokasta tietoa järjestelmän skaalautuvuudesta.
- Kattavuustyökalut voivat auttaa Testauspäällikköä ymmärtämään, kuinka suuri osuus järjestelmästä on todella käyty läpi testauksessa.

Työkalujen raportointivaatimukset pitää määritellä työkalujen valintaprosessin aikana. Nämä vaatimukset täytyy toteuttaa kunnolla työkalun konfiguroinnin aikana sen varmistamiseksi, että työkalujen tuottamia tietoja voidaan raportoida tavalla, jota sidosryhmät ymmärtävät.

# 7. Vuorovaikutustaidot – Tiimin kokoaminen – 210 min.

## Avainsanat

Testauksen riippumattomuus

## Oppimistavoitteet: Vuorovaikutustaidot – Tiimin kokoaminen

### 7.2 Yksilölliset taidot

TM-7.2.1 (K4) Osaamisen arviointitaulukkoa käyttämällä analysoida tiimin jäsenten ohjelmistojärjestelmän käyttöön, toimialan ja liiketoiminnantuntemukseen, järjestelmäkehityksen alueisiin, ohjelmistotestaukseen ja vuorovaikutustaitoihin liittyviä vahvuuksia ja heikkouksia.

TM-7.2.2 (K4) Analysoida määrätyn tiimin osaamisarviota koulutus- ja osaamisenkehittämissuunnitelman laatimiseksi.

### 7.3 Testaustiimin dynamiikka

TM-7.3.1 (K2) Keskustella tapauskohtaisesti testaustiimin johtamisessa tarvittavista kovista ja pehmeistä taidoista.

### 7.4 Testauksen sovittaminen organisaatioon

TM-7.4.1 (K2) Selittää riippumattoman testauksen vaihtoehdot.

### 7.5 Motivaatio

TM-7.5.1 (K2) Antaa esimerkkejä testaajia motivoivista ja ei-motivoivista tekijöistä.

### 7.6 Viestintä

TM-7.6.1 (K2) Selittää tekijät, jotka vaikuttavat testaustiimin sisäisen sekä testaustiimin ja sen sidosryhmien välisen viestinnän tehokkuuteen.

## 7.1 Esittely

Menestyksekäs Testauspäällikkö värvää, palkkaa ja ylläpitää tiimejä, joissa on sopiva sekoitus osaamista. Taitovaatimukset voivat muuttua ajan myötä, joten sen lisäksi, että palkataan alun perin oikeat ihmiset, on tärkeää tarjota riittävästi koulutusta ja kasvumahdollisuuksia testaustiimin koossa pitämiseksi ja sen suorituskyvyn säilyttämiseksi huipussaan. Testaustiimin osaamisen lisäksi Testauspäälliköllä on myös oltava osaamista, joka mahdollistaa tehokkaan toiminnan paineen alla nopeatahtisessa työympäristössä.

Tässä luvussa tarkastellaan, kuinka osaamista voidaan arvioida, kuinka täytetään aukot niin, että saadaan luotua synerginen, sisäisesti yhtenäinen ja organisaatioltaan tehokas tiimi, kuinka tiimiä voidaan motivoida ja miten viestitään tehokkaasti.

## 7.2 Yksilölliset taidot

Yksilö voi oppia ohjelmistotestausta sekä kokemuksen kautta että kouluttautumalla ja harjoittelemalla. Jokainen seuraavista tekijöistä voi edesauttaa testaajan osaamista:

- ohjelmistojärjestelmien käyttö
- liiketoiminta-alueen tai liiketoiminnan tuntemus
- osallistuminen ohjelmistokehitysprosessin tehtäviin sen eri vaiheissa, mukaan luettuna analysointi, toteutus ja tekninen tuki
- osallistuminen ohjelmistotestauksen tehtäviin.

Ohjelmistojärjestelmien loppukäyttäjillä on hyvä käsitys siitä, kuinka järjestelmä toimii, missä häiriöillä olisi suurin vaikutus ja kuinka järjestelmän pitäisi toimia eri tilanteissa. Liiketoiminta-alueen asiatuntemusta omaavat käyttäjät tietävät, mitkä alueet ovat liiketoiminnan kannalta tärkeimpiä ja kuinka ne vaikuttavat siihen, miten liiketoiminta saavuttaa sille asetetut tavoitteet. Tätä tietämystä voidaan käyttää apuna, kun priorisoidaan testaustehtäviä, luodaan realistista testiaineistoa ja testitapauksia ja kun todennetaan tai luodaan käyttötapauksia.

Ohjelmistokehitysprosessin (vaatimusanalyysi, arkkitehtuuri, suunnittelu ja toteutus) tuntemus tuo näkemystä siihen, kuinka virheet johtavat vikojen syntymiseen, mistä vikoja voi löytyä ja kuinka vikojen syntyminen voidaan jo alusta alkaen estää. Kokemus teknisestä tuesta tuo tietoa käyttäjien kokemuksista, odotuksista ja käytettävyysvaatimuksista. Kokemus ohjelmistokehityksestä on tärkeää, kun käytetään työkaluja, jotka vaativat ohjelmointi- ja suunnittelukokemusta ja kun osallistutaan staattiseen koodin analyysiin, koodikatselmointeihin, yksikkötestaukseen ja teknisesti painottuneeseen integrointitestaukseen.

Erityisiin ohjelmistotestauksen taitoihin kuuluu osaaminen, josta on keskusteltu Perustason sertifikaattisisällössä sekä Jatkotason Testausasiantuntijan ja Teknisen testausasiantuntijan sertifikaattisisällöissä, kuten esimerkiksi kyvyt analysoida määrityksiä, osallistua riskianalyysiin ja suunnitella testitapauksia sekä huolellisuus, jota tarvitaan testitapausten suorituksessa ja tulosten kirjaamisessa.

Erityisesti Testauspäällikön osalta on tärkeää, että hänellä on tietoa, taitoa ja kokemusta projektinhallinnasta, sillä testauksenhallintaan kuuluu monia projektinhallinnan tehtäviä, esim. suunnitelman laatiminen, edistymisen seuranta ja raportointi sidosryhmille. Projektipäällikön poissa ollessa Testauspäällikkö saattaa ottaa sekä Testauspäällikön että projektipäällikön roolin erityisesti projektin myöhemmissä vaiheissa. Nämä taidot ovat lisäys niihin, joista on keskusteltu Perustason sertifikaattisisällössä ja tässä sertifikaattisisällössä.

Teknisten taitojen lisäksi ihmissuhdetaidot, kuten rakentavan palautteen antaminen ja saaminen, vaikuttamiskyky ja neuvottelutaidot, ovat kaikki tärkeitä testauksen yhteydessä. Teknisesti pätevät testaajat todennäköisesti epäonnistuvat ellei heillä ole myös tarvittavia pehmeitä taitoja eivätkä he osaa käyttää niitä. Sen lisäksi, että menestyksellisen testausammattilaisen täytyy työskennellä tehokkaasti muiden kanssa, hänen täytyy myös olla järjestelmällinen, osata kiinnittää huomiota yksityiskohtiin ja hänellä on oltava hyvät kirjallisen ja suullisen viestinnän taidot.

Ihanteellisessa testaustiimissä on sekoitus osaamista ja kokemusta, ja tiimin jäsenet ovat halukkaita ja pystyvät opettamaan työtovereitaan ja oppimaan heiltä. Joissain ympäristöissä jotkut taidot ovat tärkeämpiä tai niitä arvostetaan enemmän kuin muita. Esimerkiksi teknisessä testausympäristössä, jossa vaaditaan API-testausta ja ohjelmointitaitoja, teknisiä taitoja voidaan arvostaa enemmän kuin liiketoiminta-alueen tuntemusta. Mustalaatikkotestausympäristössä liiketoiminta-alueen asiantuntemus voi olla kaikkein arvostetuinta. On tärkeää muistaa, että ympäristöt ja projektit muuttuvat.

Luodessaan osaamisen arviointilomaketta Testauspäällikön pitää listata kaikki taidot, jotka ovat työn kannalta tärkeitä sekä myös tehtävän kannalta sopivia. Kun taidot on listattu, jokainen tiimin jäsen voidaan arvioida käyttämällä pisteytysjärjestelmää (esim. arvio yhdestä viiteen, missä viisi on korkein kyseisellä alueella odotettu osaamisen taso). Yksilöitä voidaan arvioida heidän vahvojen ja heikkojen alueidensa määrittelemiseksi ja saadun tiedon perusteella voidaan laatia yksilö- tai ryhmäkohtaiset koulutussuunnitelmat. Testauspäällikkö voi asettaa yksilöille suoritustavoitteet heidän taitojensa parantamiseksi tietyllä alueella ja hänen tulee määritellä kriteerit, joita käytetään yksilön taitojen arvioimiseksi.

Ihmiset pitäisi palkata pitkällä tähtäimellä, ei siksi, mitä he voivat tuoda yhteen yksittäisen projektiin. Kun Testauspäällikkö investoi testaajiin ja luo jatkuvan oppimisen ympäristön, tiimin jäsenet motivoituvat kasvattamaan omia taitojaan ja osaamistaan niin, että seuraavan tilaisuuden tullen he ovat valmiita siihen.

Testauspäällikkö pystyy harvoin palkkaamaan täydellisiä tiimin jäseniä. Vaikka tiimin jäsenet olisivatkin täydellisiä senhetkiseen projektiin, he eivät välttämättä ole täydellinen yhdistelmä seuraavaan projektiin. Testauspäällikön pitää palkata henkilöitä, jotka ovat älykkäitä, uteliaita, sopeutuvaisia, halukkaita työhön, kykeneviä työskentelemään tehokkaasti osana tiimiä sekä halukkaita ja kykeneviä oppimaan. Vaikka täydellistä yksilöiden joukkoa ei todennäköisesti ole saatavilla, on mahdollista rakentaa vahva tiimi tasapainottamalla yksilöiden heikkoudet ja vahvuudet.

Osaamisen arviointilomaketta käyttämällä Testauspäällikkö voi tunnistaa, missä tiimi on vahva ja missä se on heikko. Tämä tieto luo pohjan koulutus- ja osaamisenkehittämissuunnitelmalle. Testauspäällikön pitäisi lähteä liikkeelle heikkouksista, joilla on suurin vaikutus tiimin yleiseen ja toiminnalliseen tehokkuuteen, ja päättää, kuinka nämä alueet käsitellään. Yksi lähestymistapa on koulutus, esim. lähetetään ihmisiä kursseille, järjestetään yrityksen sisäisiä koulutuksia, kehitetään räätälöityjä koulutuksia, käytetään verkko-oppimiskursseja. Toinen lähestymistapa on itseopiskelu, esim. kirjat, webinaarit, Internet-lähteet. Lisäksi eräs lähestymistapa on pariharjoittelu, esim. laitetaan jonkin tietyn taidon oppimisesta kiinnostunut henkilö tekemään kyseistä taitoa vaativaa tehtävää yhdessä sellaisen henkilön kanssa, jolla kyseinen taito jo on, tai pyydetään omia asiantuntijoita pitämään lyhyitä esityksiä omalta asiantuntemusalueeltaan, jne. (Mentorointi on samanlainen lähestymistapa, jossa uuteen rooliin siirtyvä henkilö laitetaan työpariksi kokeneemman, samaa roolia edustavan henkilön kanssa, ja tämä henkilö toimii jatkuvana tuki- ja ohjausresurssina.) Heikkouksien käsittelyn lisäksi Testauspäällikön pitää muistaa hyödyntää osaamisen arvioinnissa tunnistettuja vahvuuksia osana koulutus- ja osaamisenkehittämissuunnitelmaa. Lisää tietoja testaustiimin kehittämissuunnitelmasta, ks. [McKay07].

## 7.3 Testaustiimin dynamiikka

Parhaan mahdollisen tiimin rakentamisessa henkilöstön valinta on yksi tärkeimmistä esimiesrooliin kuuluvista tehtävistä organisaatiossa. Tehtävässä tarvittavien erityisten yksilöllisten taitojen lisäksi on monia muita asioita, joita on harkittava. Kun tiimiin ollaan valitsemassa uutta henkilöä, on mietittävä tiimin dynamiikkaa. Täydentääkö kyseinen henkilö testaustiimissä jo olevia taitoja ja persoonallisuustyyppejä? On tärkeää miettiä hyötyjä, joita saadaan, kun testaustiimissä on erilaisia persoonallisuustyyppejä samoin kuin sekoitus teknisiä taitoja. Vahva testaustiimi pystyy hoitamaan useita monimutkaisuudeltaan eri tasoisia projekteja samalla kun se hoitaa onnistuneesti vuorovaikutuksen muiden projektin tiimijäsenten kanssa.

Testaus on usein kovan paineen alainen tehtävä. Ohjelmistokehityksen aikataulut ovat usein tiiviitä, jopa epärealistisia. Sidosryhmillä on usein suuret odotukset testaustiimin suhteen, joskus jopa epärealistisen korkeat. Testauspäällikön täytyy palkata henkilöitä, jotka pystyvät selviämään hyvin painetilanteista ja jotka pystyvät työntämään turhautumisen sivuun ja keskittymään työhön, vaikka aikataulut vaikuttavat mahdottomilta. Testauspäällikön tehtävänä on huolehtia aikatauluun ja odotuksiin liittyvistä seikoista,

mutta hänen täytyy myös ymmärtää, että myös testaustiimin jäsenet tuntevat nämä paineet. Kun Testauspäällikkö hankkii henkilöitä tiimiin, on tärkeää harkita työskentely-ympäristöä ja valita persoonallisuustyyppejä, jotka sopivat kyseiseen ympäristöön. Ympäristössä, jossa on enemmän työtä kuin aikaa, Testauspäällikön pitäisi etsiä ihmisiä, jotka tekevät työnsä loppuun ja kysyvät, mitä he voivat tehdä seuraavaksi.

Yksilöt työskentelevät lujemmin ja kiinnittävät enemmän huomiota tekemiseensä, jos he tuntevat olevansa arvostettuja ja tarpeellisia. Yksilöiden täytyy ymmärtää, että he ovat tärkeä tiimin jäseniä ja että heidän panoksensa on elintärkeä tiimin menestymiselle kokonaisuutena. Kun tällainen dynamiikka saadaan aikaan, ristiinoppiminen tapahtuu vapaamuotoisesti, tiimin jäsenet itse tasapainottavat työkuorman ja Testauspäällikölle jää enemmän aikaa keskittyä ulkoisiin ongelmiin.

Uudet tiimin jäsenet täytyy sopeuttaa nopeasti tiimiin ja heille on tarjottava riittävästi ohjausta ja tukea. Jokaiselle henkilölle pitäisi antaa määrätty rooli tiimissä. Tämä voi perustua yksilölliseen arviointiprosessiin. Tavoitteena on tehdä jokaisesta henkilöstä menestyvä yksilönä samalla, kun hän myötävaikuttaa koko tiimin menestykseen. Tämä tehdään yleensä sovittamalla persoonallisuustyypit tiimin roolien mukaan ja kasvattamalla yksilön luontaisia taitoja samalla kun kasvatetaan hänen osaamissalkkuaan.

Kun tehdään päätöstä siitä, kenet palkataan tai lisätään testaustiimiin, osaamisen objektiivisesta arvioinnista voi olla apua. Tämä voidaan tehdä haastattelujen avulla, testaamalla ehdokkaita, katselmoimalla työnäytteitä ja todentamalla suositukset. Arvioitavaan osaamiseen kuuluvat mm.

Henkilö voi osoittaa tekniset taitonsa (kovat taidot)
- laatimalla testitapauksia vaatimuskuvauksesta
- katselmoimalla teknistä dokumentaatiota (vaatimukset, koodi jne.)
- kirjoittamalla katselmointikommentti selkeästi, ymmärrettävästi ja objektiivisesti
- käyttämällä eri testaustekniikoita annettuihin skenaarioihin soveltuvalla tavalla (esim. käyttämällä päätöstauluja, kun testataan joukkoa liiketoimintasääntöjä)
- arvioimalla häiriö ja dokumentoimalla se täsmällisesti
- osoittamalla, että hän ymmärtää vikaluokittelun sisältämät tiedot
- osoittamalla, että hän ymmärtää, mitä ovat vikojen perussyyt
- käyttämällä työkalua määrätyn APIn testaamiseen, mukaan luettuna positiiviset ja negatiiviset testit
- käyttämällä SQL-kieltä tietyn skenaarion testaamiseksi tarvittavan tietokanta-aineiston löytämiseen ja muuttamiseen
- suunnittelemalla testiautomaatiokehys, joka suorittaa useita testejä ja kerää testitulokset
- suorittamalla automatisoituja testejä ja selvittämällä häiriöiden syitä
- kirjoittamalla testaus- ja testisuunnitelmia
- kirjoittamalla testauksen yhteenvetoraportin, joka sisältää arvion testauksen tuloksista

Henkilö voi osoittaa vuorovaikutustaitonsa (pehmeät taidot)
- esittämällä tietoja, jotka koskevat aikataulusta myöhässä olevaa testausprojektia
- selittämällä vikaraportin toteuttajalle, jonka mielestä vikaa ei ole
- kouluttamalla työtovereita
- esittämällä johdolle ongelman, joka koskee tehotonta prosessia
- katselmoimalla työtoverin laatiman testitapauksen ja esittämällä kommentit testin laatijalle
- haastattelemalla työtovereita
- kehumalla toteuttajaa.

Tämä lista ei ole tyhjentävä ja toivotut yksittäiset taidot vaihtelevat ympäristön ja organisaation mukaan, mutta siinä luetellaan tyypillisesti tarvittavat taidot. Laatimalla tehokkaat haastattelukysymykset ja antamalla hakijalle mahdollisuuden osoittaa osaamisensa Testauspäällikkö voi arvioida hakijan taidot ja määritellä tämän vahvuudet ja heikkoudet. Kun hyvä joukko arviointimenetelmiä on laadittu, sitä pitäisi käyttää myös jo olemassa olevaan henkilöstöön kasvu- ja koulutuskohteiden tunnistamiseksi.

Yksilökohtaisten myötävaikutustaitojen lisäksi Testauspäälliköllä täytyy myös olla erinomaiset taidot viestinnässä ja diplomatiassa. Testauspäällikön täytyy pystyä selvittämään ristiriitatilanteet, hänen täytyy tietää, mikä missäkin tilanteessa on oikea viestintäkeino, ja hänen täytyy pystyä keskittymään rakentamaan ja ylläpitämään suhteita organisaation sisällä.

## 7.4 Testauksen sovittaminen organisaatioon

Organisaatioilla on monia tapoja sovittaa testaus organisaation rakenteeseen. Vaikka laatu onkin jokaisen ohjelmistokehityksen elinkaareen osallistuvan vastuulla, riippumaton testausryhmä voi myötävaikuttaa merkittävästi tuotteen laatuun. Testaustehtävien riippumattomuus vaihtelee käytännössä suuresti, kuten voidaan havaita seuraavasta listasta, joka on järjestyksessä vähiten riippumattomasta riippumattomimpaan.

- Ei riippumattomia testaajia
  - Tässä tapauksessa riippumattomuutta ei ole ja koodin testaa toteuttaja, joka kirjoitti sen.
  - Jos toteuttajalle annetaan aikaa testaamiseen, hän voi varmistaa, että koodi toimii kuten on tarkoitettu, mikä voi vastata alkuperäisiä vaatimuksia tai olla vastaamatta niitä.
  - Toteuttajat voivat korjata löydetyt viat nopeasti.
- Testauksen suorittaa joku toinen toteuttaja kuin se, joka kirjoitti koodin
  - Toteuttajan ja testaajan välillä on jonkin verran riippumattomuutta.
  - Toisen toteuttajan tekemää koodia testaava toteuttaja voi olla haluton raportoimaan vikoja.
  - Toteuttajan ajatusmalli testauksen suhteen keskittyy yleensä positiivisiin testitapauksiin.
- Testauksen tekee testaaja (tai testaustiimi), joka on osa toteutustiimiä
  - Testaaja (tai testaustiimi) voi raportoida projektijohdolle tai kehityspäällikölle.
  - Testaajan ajatusmalli on keskittynyt enemmän varmistamaan vaatimustenmukaisuutta.
  - Koska testaaja on osa kehitystiimiä, hänellä voi olla myös toteutusvastuuta testauksen lisäksi.
  - Testaajan esimies voi olla kiinnostuneempi aikataulussa pysymisestä kuin laatutavoitteiden saavuttamisesta.
  - Testaukselle voidaan antaa alempi asema tiimissä kuin toteutukselle.
  - Testaajalla ei ehkä ole valtuuksia vaikuttaa laatutavoitteisiin tai niiden noudattamiseen.
- Testauksen tekevät liiketoimintaorganisaatiosta, käyttäjäyhteisöstä tai muusta toteutukseen liittymättömästä teknisestä organisaatiosta tulevat testausasiantuntijat
  - Testitulokset raportoidaan objektiivisesti sidosryhmille.
  - Laatu on tälle tiimille keskeinen asia.
  - Osaamisen kehittäminen ja koulutus keskittyvät testaukseen.
  - Testaus nähdään urapolkuna.
  - Testausryhmälle on nimetty laatuun keskittynyt erillinen johto.
- Ulkopuoliset testausasiantuntijat suorittavat tiettyjen testaustyyppien testauksen.
  - Asiantuntemus suunnataan määrätyille alueille, joiden kohdalla yleinen testaus on todennäköisesti riittämätöntä.
  - Testityyppejä voivat olla käytettävyys, tietoturva, suorituskyky tai muut alueet, joissa erikoistuminen on välttämätöntä.
  - Laadun pitäisi olla näille henkilöille keskeinen asia, mutta heidän näkemyksensä rajoittuu heidän omaan erikoisalueeseensa. Tuote, jonka suorituskyky on hyvä, ei ehkä täytä sille asetettuja toiminnallisia vaatimuksia, mutta se voi jäädä suorituskykyasiantuntijoilta huomaamatta.
- Testauksen tekee yrityksen ulkopuolinen organisaatio
  - Tällä mallilla saavutetaan suurin riippumattomuus testaajan ja toteuttajan välillä.
  - Tiedonsiirto ei ehkä ole riittävää, jotta testaaja voisi testata riittävästi.
  - Tarvitaan selkeät vaatimukset ja hyvin määritelty viestintärakenne.
  - Ulkoisen organisaation laatu on tarkastettava säännöllisesti.

Tämä lista pohjautuu henkilöiden tyypillisiin painopistealueisiin, mutta ei ehkä päde määrätyssä organisaatiossa. Asema ja nimike eivät välttämättä määritä henkilön toiminnan painopistettä. Kehitysjohtajat voivat keskittyä laatuun ja samoin voivat hyvät Testauspäälliköt. Riippumattomat Testauspäälliköt voivat

raportoida aikataulua painottavalle johtoryhmälle ja näin ollen he voivat keskittyä enemmän aikatauluun kuin laatuun. Kun määritetään testaustiimille organisaation kannalta parasta sijaintia, Testauspäällikön pitää ymmärtää organisaation tavoitteet.

Toteutus- ja testausorganisaation välillä on erilaisia riippumattomuuden asteita. On tärkeää ymmärtää, että voidaan joutua tekemään vaihtokauppoja, kun korkeampi riippumattomuus johtaa suurempaan eristäytyneisyyteen ja vähempään tiedonvaihtoon. Matalan tason riippumattomuus voi kasvattaa osaamista, mutta se voi myös tuoda esiin ristiriitaisia tavoitteita. Riippumattomuuden tasoon vaikuttaa myös käytettävä ohjelmistokehityksen malli; esim. ketterässä kehityksessä testaajat ovat useimmiten osa kehitystiimiä.

Mikä tahansa yllä mainituista vaihtoehdoista saattaa esiintyä organisaatiossa. Testausta voivat tehdä kehitysorganisaation edustajat samoin kuin riippumattomat testaajat, ja käytössä voi olla ulkoisen organisaation tekemä lopullinen sertifiointi. On tärkeä ymmärtää jokaiseen testausvaiheeseen liittyvät vastuut ja odotukset ja asettaa vaatimukset niin, että lopullisen tuotteen laatu on paras mahdollinen, vaikka pysytään aikataulu- ja budjettirajojen sisällä.

## 7.5 Motivaatio

Testaustehtävässä toimivan henkilön motivoimiseen on monia tapoja. Näihin kuuluvat

- tehdyn työn arvostaminen
- johdon antama tunnustus
- kunnioitus projektitiimissä ja työtovereiden kesken
- kasvava vastuu ja itsenäisyys
- riittävät palkkiot tehdystä työstä (mukaan lukien palkka, kasvanut vastuu, arvostus).

Projektiin voi vaikuttaa tekijöitä, jotka tekevät näiden motivaatiota kasvattavien työkalujen käytön vaikeaksi. Testaaja voi esimerkiksi työskennellä ahkerasti projektissa, jolla on mahdoton aikataulu. Hän voi tehdä kaiken mahdollisen edistääkseen laatuun panostusta tiimissä, käyttää lisätunteja ja –panostusta, ja silti tuote voidaan toimittaa eteenpäin ulkopuolelta tulevien syiden vuoksi ennen kuin pitäisi. Tuloksena voi olla huonolaatuinen tuote kaikista testaajan ponnisteluista huolimatta. Tämä voidaan helposti kokea motivaatiota laskevana, jos testaajan panosta ei ymmärretä eikä mitata huolimatta siitä, onko lopputuote onnistunut.

Testaustiimin täytyy varmistaa, että tiimissä seurataan oikeita mittareita, joilla voidaan todistaa, että testauksessa, riskien pienentämisessä ja tulosten oikeellisessa kirjauksessa tehtiin kunnollista työtä. Jos tätä tietoa ei kerätä ja julkisteta, laskee testaustiimin jäsenten motivaatio helposti, kun he eivät saa mielestään hyvin tehdystä työstä heille kuuluvaa tunnustusta.

Tunnustus ei käsitä vain aineettomia asioita kuten kunnioitusta ja hyväksyntää, vaan se tulee esiin myös tilaisuuksina ylennyksiin, kasvavana vastuuna ja ansioitumiseen perustuvina korotuksina. Jos testausryhmää ei kunnioiteta, näitä mahdollisuuksia ei ehkä ole saatavilla.

Tunnustusta ja kunnioitusta saavutetaan, kun on selvää, että testaaja myötävaikuttaa projektin kasvavaan arvoon. Yksittäisessä projektissa tämä saadaan parhaiten aikaan ottamalla testaajat mukaan toimintaan heti projektin alussa ja jatkamalla heidän osallistumistaan läpi elinkaaren. Ajan myötä testaajat saavuttavat positiivisella myötävaikutuksellaan muiden projektin sidosryhmien kunnioituksen. Tälle vaikutukselle pitäisi myös saada numeeriset mittarit laatua ja riskien lieventämistä koskevien kustannusten vähenemiseen liittyen.

Testauspäälliköllä on tärkeä osa testaustiimin yksittäisten jäsenten motivoinnissa samoin kuin testaustiimin puolestapuhujana laajemmin organisaatiossa. Testauspäällikön pitää antaa tunnustusta yksittäisten testaajien saavutuksista ja hänen täytyy myös antaa oikeudenmukaista ja rehellistä palautetta virheistä. Reilu ja johdonmukainen Testauspäällikkö motivoi tiimin jäseniä omalla esimerkillään.

## 7.6 Viestintä

Testaustiimissä viestintä tapahtuu pääasiassa seuraavilla tavoilla:

- Testauksen tuotosten dokumentointi – testausstrategia, testaussuunnitelma, testitapaukset, testauksen yhteenvetoraportit, havaintoraportit jne.
- Palaute katselmoiduista dokumenteista – vaatimukset, toiminnalliset määritykset, käyttötapaukset, yksikkötestauksen dokumentaatio jne.
- Tietojen keruu ja käsittely – vuorovaikutus kehittäjien, muiden tiimin jäsenten, johdon jne. kanssa.
-

Testaajien ja muiden sidosryhmien välisen viestinnän pitää olla ammattimaista, objektiivista ja tehokasta, jotta testaustiimin kunnioitus kasvaa ja pysyy. Palautteen antamisessa toisten tuotoksista tarvitaan diplomaattisuutta ja objektiivisuutta, erityisesti kun annetaan rakentavaa palautetta. Tämän lisäksi viestinnän pitäisi keskittyä testauksen tavoitteiden saavuttamiseen ja parantamaan sekä tuotteen että ohjelmistojärjestelmien toteuttamisessa käytettävien prosessien laatua.

Testauspäälliköt kommunikoivat laajan kuulijajoukon kanssa, johon kuuluu mm. käyttäjiä, projektitiimin jäseniä, johtohenkilöitä, ulkoisia testausryhmiä ja asiakkaita. Viestinnän täytyy olla kohdeyleisön kannalta tehokasta. Esimerkiksi toteutustiimiä varten laadittu raportti, joka esittää löydettyjen vikojen määrän ja vakavuuden trendit ja suuntaukset, saattaa olla liian yksityiskohtainen ollakseen käyttökelpoinen toiminnallisen johdon tilannekatsauksessa. Kaikessa viestinnässä, mutta erityisesti esitysten aikana, on tärkeä ymmärtää viesti, jota esityksellä välitetään, tavat, joilla viesti voidaan vastaanottaa, ja selitykset, joita tarvitaan viestin hyväksymisen kannalta oikean ympäristön luomiseksi. Koska Testauspäällikön tehtäviin kuuluu usein projektin tilannetietojen esittäminen, on tärkeää, että nämä tiedot ovat sopivan yksityiskohtaisia (esim. johtajat haluavat yleensä nähdä vikatrendejä ennemmin kuin yksittäisiä vikoja) ja että ne esitetään selkeästi ja helposti ymmärrettävällä tavalla (esim. yksinkertaiset taulukot ja värilliset kaaviot). Tuottavasti viestiminen auttaa pitämään yleisön mielenkiinnon hereillä samalla kun se silti välittää oikean viestin. Testauspäällikön tulisi pitää jokaista esitystä mahdollisuutena edistää laatu- ja laatuprosessiasioita.

Testauspäällikkö ei kommunikoi vain osaston ulkopuolisten ihmisten kanssa (ulkoinen viestintä). Tärkeä osa Testauspäällikön tehtävää on kommunikoida tehokkaasti testausryhmän sisällä (sisäinen viestintä) ja tiedottaa uutisista, ohjeista, prioriteettien muutoksista ja muusta perustiedosta, jota välitetään tavanomaisen testauksen aikana. Testauspäällikkö voi myös viestiä määrätyille yksilöille organisaation johtoketjussa sekä ylöspäin (ylös suuntautuva viestintä) että alaspäin (alas suuntautuva viestintä). Viestinnän suunnasta huolimatta samat säännöt pätevät; viestinnän pitää olla kohdeyleisölle soveltuvaa, viesti pitää lähettää tehokkaasti ja sen ymmärtäminen pitää varmistaa.

Testauspäälliköiden täytyy olla eri kommunikaatiotapojen mestareita. Suuri osa tiedoista välitetään sähköpostilla, sanallisesti, muodollisissa tai epämuodollisissa kokouksissa, muodollisissa tai epämuodollisissa raporteissa ja jopa testauksen hallintatyökaluja, kuten havaintojenhallintatyökalua, käyttämällä. Kaiken viestinnän pitää olla ammattimaista ja objektiivista. Sekä laadullinen että sisältöön kohdistuva oikoluku on tarpeellinen vaihe kaikkein kiireisimmässäkin viestinnässä. Kirjallinen viestintä säilyy usein paljon varsinaista projektia pidempään. Testauspäällikön on tärkeää tuottaa laadullisesti ammattimaista dokumentaatiota, joka edustaa laatua arvostavaa organisaatiota.

# 8. Viitteet

## 8.1 Standardit

- [BS7925-2] BS 7925-2, Software Component Testing Standard.
  Luku 2
- [FDA21] FDA Title 21 CFR Part 820, Food and Drug Administration, US
  Luku 2
- [IEEE829] IEEE Standard for Software and System Test Documentation
  Luvut 2 ja 4
- [IEEE1028] IEEE Standard for Software Reviews and Audits
  Luku 2
- [IEEE1044] IEEE Standard Classification for Software Anomalies
  Luku 4
- [ISO9126] ISO/IEC 9126-1:2001, Software Engineering - Software Product Quality,
  Luvut 2 ja 4
- [ISO12207] ISO 12207, Systems and Software Engineering, Software Life Cycle Processes
  Luku 2
- [ISO15504] ISO/IEC 15504, Information Technology - Process Assessment
  Luku 2
- [ISO25000] ISO/IEC 25000:2005, Software Engineering - Software Product QualityRequirements and Evaluation (SQuaRE)
  Luvut ja 4
- [RTCA DO-178B/ED-12B]: Software Considerations in Airborne Systems and Equipment Certification, RTCA/EUROCAE ED12B.1992.
  Luku 2

## 8.2 ISTQB Documentit

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

## 8.3 Tavaramerkit

Seuraavia rekisteröityjä tavaramerkkejä ja palvelumerkkejä käytetään tässä dokumentissa:

- CMMI®, IDEALSM, ISTQB®, ITIL®, PRINCE2®, TMMi®, TPI Next®
- CMMI:n on rekisteröinyt Carnegie Mellon University Yhdysvaltain Patentti- ja tavaramerkkivirastossa.

- IDEAL on Software Engineering Institute (SEI), Carnegie Mellon Universityn rekisteröimä palvelumerkki.
- ISTQB on International Software Testing Boardin rekisteröity tavaramerkki.
- ITIL on rekisteröity tavaramerkki, Office of Government Commercen rekisteröimä yhteisöllinen tavaramerkki, ja se on rekisteröity Yhdysvaltain Patentti- ja tavaramerkkivirastossa.
- PRINCE2 on Cabinet Officen rekisteröimä tavaramerkki.
- TMMi on TMMi Foundationin rekisteröimä tavaramerkki.
- TPI Next on Sogeti Nederland B.V:n rekisteröimä tavaramerkki.

## 8.4 Kirjat

[Black03]: Rex Black, “Critical Testing Processes,” Addison-Wesley, 2003, ISBN 0-201-74868-1

[Black09]: Rex Black, “Managing the Testing Process, third edition,” John Wiley & Sons, 2009, ISBN 0-471-22398-0

[Black11]: Rex Black, Erik van Veenendaal, Dorothy Graham, “Foundations of Software Testing,” Thomson Learning, 2011, ISBN 978-1-84480-355-2

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

[Crispin09]: Lisa Crispin, Janet Gregory, “Agile Testing: A Practical Guide for Testers and Agile Teams”, Addison-Wesley, 2009, ISBN 0-321-53446-8

[de Vries09]: Gerrit de Vries, et al., “TPI Next”, UTN Publishers, 2009, ISBN 90-72194-97-7

[Goucher09]: Adam Goucher, Tim Riley (editors), “Beautiful Testing,” O’Reilly, 2009, ISBN 978-0596159818

[IDEAL96]: Bob McFeeley, “IDEAL: A User's Guide for Software Process Improvement,” Software Engineering Institute (SEI), 1996, CMU/SEI-96-HB-001

[Jones07]: Capers Jones, “Estimating Software Costs, second edition,” McGraw-Hill, 2007, ISBN 978-0071483001

[Jones11]: Capers Jones and Olivier Bonsignour, “Economics of Software Quality,” Pearson, 2011, ISBN 978-0132582209

[McKay07]: Judy McKay, “Managing the Test People,” Rocky Nook, 2007, ISBN 978-1933952123

[Musa04]: John Musa, “Software Reliability Engineering, second edition,” Author House, 2004, ISBN 978-1418493882

[Stamatis03]: D.H. Stamatis, “Failure Mode and Effect Analysis,” ASQC Quality Press, 2003, ISBN 0-873-89300

[vanVeenendaal11] Erik van Veenendaal, “The Little TMMi,” UTN Publishers, 2011, ISBN 9-490-986038

[vanVeenendaal12] Erik van Veenendaal, “Practical Risk-based Testing,” UTN Publishers, 2012, ISBN 978-9490986070

[Whittaker09]: James Whittaker, “Exploratory Testing,” Addison-Wesley, 2009, ISBN 978-0321636416

[Wiegers03]: Karl Wiegers, “Software Requirements 2,” Microsoft Press, 2003, ISBN 978-0735618794

## 8.5 Muut viitteet

Seuraavat viittaukset kohdistuvat Internetistä saatavilla olevaan tietoon. Nämä viittaukset tarkistettiin tämän Jatkotason sertifikaattisisällön julkaisun aikaan.

http://www.istqb.org
http://www.sei.cmu.edu/cmmi/
http://www.tmmi.org/
http://www.tpinext.com/