Lappeenrannan teknillinen yliopisto
Tietotekniikan osasto
Kandidaatintyö

Yritysten valmius soveltaa uusia ohjelmistotuotteiden testaus- ja laatustandardeja (ISO/IEC 29119 ja 25010)

Työn ohjaaja ja tarkastaja: Tekniikan tohtori Ossi Taipale

Lappeenranta, 20.4.2009

Tommi Kähkönen
Punkkerikatu 2 as 15
53850 Lappeenranta
Puh: 040-7051781

TIIVISTELMÄ

Lappeenrannan Teknillinen Yliopisto
Tietotekniikan osasto

Tommi Kähkönen

Yritysten valmius soveltaa uusia ohjelmistotuotteiden testaus- ja laatustandardeja (ISO/IEC 21119 ja 25010)

Kandidaatintyö

2009

64 sivua, 1 luettelo, 7 kuvaa, 7 taulukkoa

Tarkastaja: Tekniikan tohtori Ossi Taipale

Hakusanat: ohjelmistotuotannon standardi, ohjelmistotestaus, testausstandardi, ohjelmistotuotteen laatu, laatustandardi, laatumalli

Ohjelmistotuotannon standardit tarjoavat yritykselle vakiintuneita käytäntöjä ja mahdollistavat tehokkaamman kehitysprosessin, jolla tuotetaan laadukkaampia lopputuotteita pienemmillä kustannuksilla. Testaus aiheuttaa merkittävän osan ohjelmistoprojektin kustannuksista, ja tästä syystä testauksen suunnittelu ja toteutus standardoimalla testausprosesseja on suositeltavaa. Tässä kandidaatintyössä tarkastellaan yritysten valmiutta soveltaa uutta, kehitteillä olevia ohjelmistotuotannon testausstandardia ISO/IEC 29119 ja ohjelmistotuotteen laatustandardia ISO/IEC 25010. Tutkimusmenetelmänä tässä työssä käytetään laadullista tutkimusta.

ABSTRACT

Lappeenranta University of Technology
Department of Information Technology

Tommi Kähkönen

Readiness to use the new software product testing and quality standards (ISO/IEC 29119 and 25010) in companies

Bachelor's Thesis

2009

64 pages, 1 list, 7 figures, 7 tables

Supervisor: D.Sc. Ossi Taipale

Keywords: software engineering standard, software testing, testing standard, software quality, quality standard, quality model

Standards in software engineering offer settled practises and enable more efficient development process allowing quality products with lower costs. Software testing causes the major part of the costs of software development process and due to this fact companies should consider planning and executing testing by using standardized testing processes. The goal of this bachelor's thesis is to evaluate companies' ability to use the new software testing standard ISO/IEC 29119 and the new software product quality standard ISO/IEC 25010. The qualitative method is used as the research method in this thesis.

SISÄLLYSLUETTELO

LYHENTEET

ANTI	Application Strategies of New Technologies in Industrial Automation
BS	British Standards
CMMI	Capability Maturity Model Integration
IEC	The international Electrotechnical Comission
IEEE	Institute of Electrical and Electronics Engineers
ISO	The international Organization for Standards
IT	Information Technology
MASTO	Adaptive Reference Model for Software Testing Organizations
OU	Organizational Unit
ROI	Return of Investment
SQuaRE	Software product Quality Requirements and Evaluatiuon
TBRC	Technology Business Research Center
TPI	Test Process Improvement

1 JOHDANTO

Tämä kandidaatintyö toteutetaan osana MASTO (Adaptive Reference Model for Software Testing Organizations) -hanketta. Hankkeesta vastaa Lappeenrannan teknillisen yliopiston TBRC (Technology Business Research Center) -yksikkö. MASTO -hanke on jatkoa aikaisemmalle ANTI (Application Strategies of New Technologies in Industrial Automation) -hankkeelle, jossa selvitettiin ohjelmistotestaukseen vaikuttavia tekijöitä ja niiden välisiä riippuvuussuhteita. ANTI -hankkeessa havaittuihin tekijöihin kuuluvat prosessit, tietämyksenhallinta sekä testausautomaatio. MASTO-hankkeen tarkoituksena on tutkia tarkemmin näitä tekijöitä, niiden välisiä riippuvuuksia sekä niiden vaikutuksia ohjelmistotestaukseen.

1.1 Työn tavoitteet ja menetelmät

MASTO -projektin ensimmäisellä kierroksella on haastateltu kahtatoista suomalaista yritystä, joiden liiketoimintaan kuuluu olennaisena osana ohjelmistojen tuottaminen. Työn tavoitteena on kerätä haastatteluaineistosta viittaukset, jotka liittyvät uusiin, kehitteillä oleviin standardeihin. Nämä standardit ovat testausstandardi ISO/IEC 29119 ja ohjelmistotuotteen laatustandardi ISO/IEC 25010. Yrityksillä ei ole vielä käytössään näitä standardeja. Tämän työn tavoitteena on muodostaa alustava käsitys yrityksissä vallitsevasta tilanteesta ja arvioida yritysten valmiutta soveltaa uusia standardeja toiminnassaan. Havainnot raportoidaan tämän kandidaatintyön tulokset -osiossa. Tutkimusmenetelmänä tässä työssä käytetään laadullista tutkimusta ja aineistona MASTO -hankkeen ensimmäisen ja toisen kierroksen litteroituja haastatteluja sekä kyselylomakkeita. Haastatteluaineiston analysointiin käytetään aineistopohjaista menetelmää ja aineiston käsittelyyn Atlas.ti -ohjelmaa.

1.2 Työn rakenne ja rajaukset

Työn teoriaosuudessa esitellään ohjelmistotestauksen peruskäsitteitä sekä testaukseen liittyviä ongelmia. Ohjelmistotestaus on hyvin laaja käsite ja tästä syytä testauksen teoriaa ei tässä työssä käsitellä kovinkaan laajasti. Myöskään testausteoriaan liittyvää lähdekirjallisuutta ei ole lukumäärällisesti käytetty kovin laajasti, sillä testaamiseen liittyvät peruskysymykset ovat pysyneet samoina jo vuosikymmenten ajan tietotekniikan kehityksestä huolimatta [8]. Testausteoria keskittyy lähinnä testauksen yleisiin periaatteisiin ja ongelmiin. Seuraavaksi tämän työn teoriaosuudessa

motivoidaan standardien hyödyllisyys yritykselle kuvaamalla standardien rooleja ja etuja, joita yritys voi standardeja hyödyntämällä saavuttaa.

Tämän jälkeen siirrytään käsittelemään tämän työn kannalta keskeisempiä aiheita, joita ovat ISO/IEC 29119 -testausstandardi ja ISO/IEC 25010 -laatustandardi. Lähdemateriaalin osalta keskitytään standardien vedosversioiden dokumentteihin. Koska nämä standardit ovat myös laajoja kokonaisuuksia, keskitytään testausstandardin osalta neliosaiseen prosessimalliin ja laatustandardin osalta keskitytään puolestaan ohjelmistotuotteen laatumalliin. Esimerkiksi ISO/IEC 29119:n määrittelemiä dokumenttipohjia ja ISO/IEC 25010:n määrittelemää ”laatu käytettäessä” -mallia ei tässä työssä käsitellä. Standardien esittämisen jälkeen kuvataan lyhyesti laadullisen tutkimusmenetelmän periaatteet.

Tulokset -osassa raportoidaan haastatteluaineistosta havaitut seikat, jotka liittyvät standardien soveltuvuuteen. Haastatteluaineistossa keskitytään testausstrategiaan ja sen kehittämiseen liittyviin asioihin. Yritysten tämänhetkisiä testausstrategioita arvioidaan ja niiden perusteella mietitään, miltä osin yritys hyötyisi uudesta testausstandardista. Lisäksi arvioidaan uuden laatustandardin soveltuvuutta läpikäymällä yritysten omat arviointinsa laatuominaisuuksien painottumisesta ja laadun seuraamisesta

2 TESTAUS OHJELMISTOTUOTANTOPROSESSISSA

Testaus on ohjelmistoprosessin vaihe, joka vaikuttaa oleellisesti projektin kustannuksiin. Testaukseen liittyviä työvaiheita ovat testauksen suunnittelu, testiympäristön luonti, testin suorittaminen ja tulosten tarkastelu sekä virheiden jäljittäminen ja korjaaminen. Jopa puolet ohjelmistoprojektin kustannuksista voi aiheutua testauksesta. Yrityksen onkin syytä kiinnittää erityistä huomiota sekä kohdentaa voimavaroja testauksen onnistuneeseen suunnitteluun ja läpivientiin koko projektin ajan. Kaikenkattavaa testausta on mahdoton tehdä. Päätös siitä, mitä testataan ja milloin testaus lopetetaan, on aina kompromissi aikataulun, resurssien, välineiden ja testitulosten luotettavuuden arvioinnin välillä [3].

Testaukseen liittyvien haasteiden vuoksi testaus tulisi määritellä hyvin ja siihen liittyvät tehtävät tulisi sisällyttää ohjelmiston elinkaaren kaikkiin vaiheisiin. Kuitenkin hyvinkin usein testaus toteutetaan vailla selvää menetelmää ja ilman automaation tai työkalujen tukea. Vaikka monimutkainen ohjelmisto tekee täydellisestä testauksesta mahdotonta, hyvin suunniteltu testausmenetelmä ja viimeisimpien työkalujen käyttö voivat parantaa testauksen tuottavuutta ja tehokkuutta merkittävästi. Testaus tulisi sisällyttää osaksi ohjelmistotuotantoprosessia siten, ettei sitä katsottaisi pelkästään tehtäväksi, joka suoritetaan ohjelmiston kehityssyklin loppupuolella hieman ennen valmiin tuotteen toimittamista asiakkaalle. Mitä aikaisemmin virhe havaitaan, sitä halvemmaksi sen korjaaminen tulee. Perinteisessä vesiputousmallissa testaus ajoittuu prosessin loppupuolelle, mikä aiheuttaa virheiden myöhäisen havaitsemisen ja siten virheiden korjauskustannusten nousun. Iteratiivisissa menetelmissä testaus sisältyy jokaiseen kierrokseen, jolloin virheet havaitaan huomattavasti aikaisemmin kuin vesiputousmallissa [9].

2.1 Testauksen tasot

Testauksen tasoja ovat virheiden paikallistaminen ja korjaus (debugging), yksikkötestaus, integrointitestaus, järjestelmätestaus ja hyväksymistestaus. Virheiden paikallistaminen ja korjaus sekä yksikkötestaus katsotaan usein ohjelmoijan itsensä tehtäväksi. Muut testauksen tasot tulisi suorittaa erillisten testaajien toimesta, koska tällöin testaus on tehokkaampaa ja virheitä voidaan löytää helpommin. Virheiden paikallistaminen ja korjaus on testauksen alin taso. Siihen sisältyy ohjelman kääntäminen kääntäjän tai tulkin avulla ja virheiden etsiminen lähdekooditasolla. Virheet

liittyvät usein kirjoitettuun lähdekoodiin, ja ne voivat johtua tapahtumien kulun virheellisestä toteutuksesta, ohjelmointikielen ominaisuuksien väärinymmärryksistä tai epäjohdonmukaisesta ohjelmakoodin kirjoitustavasta [8], [9].

2.1.1 Yksikkötestaus

Yksikkötestauksessa (myös komponentti ja moduulitestaus tarkoittavat samaa asiaa) testattavana on yksittäinen moduuli, joka voi koostua esimerkiksi 100-1000 ohjelmarivistä. Moduuli on pienin ohjelman osa, jolla on määrittely. Yksikkötestauksessa moduulin toimintaa verrataan moduulin määrittelyyn. Yksikkötestauksen avulla voidaan hallita laajan ohjelmakokonaisuuden testausta jakamalla ohjelma pieniin elementteihin. Yksikkötestauksen tärkeänä tehtävänä on myös helpottaa myös virheiden paikallistamista ja korjausta, koska sen avulla virheet voidaan paikantaa yksittäisiin moduuleihin [2], [8].

2.1.2 Integrointitestaus

Integrointitestauksessa yhdistellään moduuleita ja testataan niiden yhteistoimivuutta. Painopiste on moduulien välisten rajapintojen ja vuorovaikutuksen toimivuuden tutkimisessa. Integraatiotestaus etenee usein rinnakkain yksikkötestauksen kanssa ja sitä onkin usein tarpeetonta tarkastella erillään moduulitestauksesta [3].

2.1.3 Järjestelmätestaus

Järjestelmätestauksen tarkoitus on verrata järjestelmää tai ohjelmaa sen alkuperäisiin tavoitteisiin, jotka on kuvattu määrittelydokumentaatiossa. Järjestelmätestaus on mahdotonta, mikäli tuotteella ei ole kirjoitettuja ja mitattavia tavoitteita. Mikäli tuote on ohjelma, järjestelmätestauksen tavoitteena on demonstroida, kuinka ohjelma kokonaisuudessaan täyttää siihen kohdistuneet vaatimukset. Mikäli tuote on järjestelmä, testaukseen kuuluu myös sen rajapinnat muihin järjestelmiin. Järjestelmätestaus toteutetaan valitussa ympäristössä valituilla laitteilla. Järjestelmätestaus kattaa myös järjestelmän ei-toiminnallisten ominaisuuksien testaamisen esimerkiksi turvallisuus-, suorituskyky-, yhteistoimivuus-, käytettävyys-, asennettavuus- ja luotettavuustestitien avulla.

Järjestelmätestausta ei yleensä tulisi suorittaa järjestelmää kehittävän organisaation toimesta [2], [3], [8], [9].

2.1.4 Hyväksymistestaus

Hyväksymistestaus on viimeinen testauksen muoto, joka suoritetaan ennen tuotteen toimittamista asiakkaalle. Hyväksymistestaus on prosessi, jossa ohjelmistotuotetta verrataan sen vaatimuksiin ja loppukäyttäjän tarpeisiin. Alfa- ja beetatestaus ovat hyväksymistestauksen tyyppejä. Alfatestaus suoritetaan yleensä toimittajan ympäristössä ja siihen osallistuvat kehittäjien lisäksi usein myös asiakkaat. Beetatestauksessa järjestelmä toimitetaan tuotteen luonteesta riippuen yhdelle tai useammalle asiakkaalle, jotka suorittavat testauksen omassa ympäristössään ja ilmoittavat havaituista virheistä kehittäjille. Hyväksymistestaus voi sisältyä järjestelmätestaukseen, ja siihen voi liittyä myös ei-toiminnallista testausta [2], [3], [8], [9].

2.1.5 Ei-toiminnallinen testaus

Sekä järjestelmätestaukseen että hyväksymistestaukseen liittyy usein myös ei-toiminnallista testausta. Testauksen suunnittelussa tulisi huomioida, mitkä ei-toiminnallisen testauksen tyypeistä täytyy huomioida testauksessa. Ei-toiminnallisen testauksen tyyppejä ovat esimerkiksi kuormitus-, käytettävyys-, turvallisuus-, suorituskyky-, installoitavuus-, luotettavuus-, ja toipuvuustestaus. Lisäksi on olemassa myös muita ei-toiminnallisen testauksen tyyppejä. Päätös siitä, mitä näistä testeistä tehdään ja mihin kiinnitetään huomiota, määräytyy sovelluksen vaatimuksien mukaan [8].

2.2 Testauksen suunnittelu ja hallinta

Testauksen suunnittelussa aliarvioidaan usein testauksen vaatimat resurssit, kuten ihmiset, aika ja laitteet. Myös testauksen ajoittuminen ohjelmistokehityssyklin loppupuolelle vaikeuttaa testaamiseen käytettävien resurssien hallintaa. Merkittävänä ongelmana on myös testauksen virheellinen määrittely. Testauksen tavoitteena tulisi olla virheiden löytäminen mieluummin kuin todistella ohjelman virheettömyyttä. Hyvin laadittu testaussuunnitelma on tärkeä osa testausprosessin hallintaa. Seuraava taulukko kuvaa hyvän testaussuunnitelman sisällön [8].

1	Tavoitteet	Testauksen jokainen vaihe tulisi määritellä.
2	Päättämiskriteerit	Jokaiselle vaiheelle tulisi määritellä päättämiskriteerit.
3	Aikataulut	Jokaiselle vaiheelle tulisi määritellä aikataulut; milloin testitapaukset suunnitellaan, kirjoitetaan ja toteutetaan.
4	Vastuut	Jokaiselle vaiheelle tulisi määrittää henkilöt, jotka suunnittelevat, kirjoittavat, toteuttavat, vahvistavat testitapaukset sekä korjaavat löytyneet virheet.
5	Testitapauskirjastot ja standardit	Etenkin suurissa projekteissa, systemaattiset menetelmät testitapausten tunnistamiselle, kirjoittamiselle ja varastoimiselle tulisi määritellä.
6	Työkalut	Tarvittavat testaustyökalut täytyy tunnistaa. Lisäksi tulee suunnitella, kuka toteuttaa tai hankkii nämä työkalut sekä miten ja milloin niitä tulee käyttää.
7	Laitteiston käyttöaika	Tähän sisältyy suunnitelma testauksen vaatimalle laitteiston käyttöajalle. Testaukseen tarvittavaa laitteistoa on esimerkiksi kääntämisohjelmien suorittamiseen tarvittavat palvelimet, työasemat installoitavuustestaukseen, web-palvelimet web-pohjaisia sovelluksia varten ja tarvittavat verkkolaitteet.
8	Laitteistokokoonpano	Suunnitelma, mikäli testaus vaatii erityisiä laitteita tai laitteistokokoonpanoja.
9	Integrointi	Järjestelmän integrointisuunnitelma määrittelee järjestyksen, jonka mukaan integrointi toteutetaan.
10	Jäljitysmenetelmät	Jäljitysmenetelmät tarkoittavat keinoja esimerkiksi virheellisten moduulien jäljittämiseen. Lisäksi testauksen edistymistä tulee arvioida aikataulun, resurssien ja päättämiskriteereiden perusteella.
11	Testausmenetelmät	Testausmenetelmiin sisältyy mekanismien määrittely virheiden raportoinnille, virheiden korjauksen edistymisen seuraamiselle ja korjausten lisäämiselle järjestelmään. Myös aikataulut, vastuut, työkalut ja resurssit täytyy huomioida osana testausmenetelmien suunnittelua.
12	Regressiotestaus	Regressiotestaus suoritetaan, kun ohjelmaan tehdään toiminnallinen parannus tai virheenkorjaustoimenpide. Tarkoituksena on arvioida muutoksen vaikutusta järjestelmän muihin osiin. Tavallisesti regressiotestauksessa suoritetaan uudelleen jokin joukko testitapauksia. Suunnitelma regressiotestaukselle; kuka tekee, miten tekee ja milloin tekee, on välttämätön.

Luettelo 1: Testaussuunnitelman sisältö [8]

2.3 Testauksen kustannukset

Testauksesta aiheutuneet kulut ohjelmistotuotannossa ovat huomattavat. On ensiarvoisen tärkeää löytää keinoja testauskustannusten alentamiseen. Mitä lähempänä tuotteen toimittamista asiakkaalle

ollaan, sitä kalliimmaksi tulee virheiden korjaus. Esimerkiksi järjestelmätestauksessa havaittu virheen korjaaminen voi aiheuttaa muutoksia useisiin moduuleihin. Jotta huomattaisiin mahdolliset muutostarpeet, kaikki moduulit tulisi testata uudelleen ja lopuksi tulisi suorittaa vielä järjestelmätestaus, mieluiten kokonaan uudelleen. Regressiotestauksen suorittaminen voi tulla erittäin kalliiksi, mikäli sitä ei saada automatisoitua [1], [3].

Testauksen kustannuksia voidaan alentaa ja testauksen tehokkuutta parantaa kehittämällä testausprosessia, testausstrategioita testausautomaatiota ja testauksen tietämystä. Testauksesta aiheutuvien kustannusten suurimpana tekijänä ovat henkilöstökustannukset. Toinen kustannuksia merkittävästi lisäävä tekijä on testausautomaatio. Testauksen tehokkuutta voidaan myös parantaa standardeja hyödyntämällä ja toteuttamalla testaus alihankkijoilla.

2.4 Testaus ja laatu

Kehittämällä edellä mainittuja tekijöitä voidaan parantaa myös lopputuotteen laatua. Testausta ei kuitenkaan pitäisi mieltää tekijäksi, jolla ohjelmiston laatu rakennetaan. Testaus pitäisi mieluummin nähdä keinona mitata ja varmistaa kehitysprosessin tuloksena syntynyt laatu. Testauksella voidaan varmistaa laatuominaisuuksien läsnäolo. Laatuominaisuuksia ovat esimerkiksi toiminnallisuus, luotettavuus ja suorituskyky. Testaus tulisi nähdä omana prosessinaan kehitysprosessin rinnalla. Prosessien välillä tulisi olla selkeä yhteys sekä riittävästi kommunikointia, jotta samoja asioita ei tehtäisi useita kertoja. [1], [9].

Yleisten menettelytapojen kehittäminen vaatii kuitenkin muutoksia yrityksen toimintaan, ja tästä syystä testausprosessin kehittäminen voi olla vaikeaa. Eräs kehitys- ja testausprosessien parantamisen avaintekijä on testauksen aikainen sisällyttäminen toimintaan, mikä tarkoittaa asiakasvaatimusten, arkkitehtuurien ja toiminnallisten määrittelyiden testaamista. Myös laatu tulisi integroida kehitysprosessiin siten, että ohjelmisto suunnitellaan siten, että laatuominaisuuksien testaaminen on mahdollista. Testausprosessia tulisi mitata ja mittausten tuloksia arvioida [1].

3. STANDARDIT

Kansainväliset standardit helpottavat henkilöiden ja yritysten välistä yhteistyötä. Jotta ihmiset ja yritykset voisivat kommunikoida keskenään, tarvitaan yhteistä sanastoa, yhteneväistä työskentelyprosessia sekä mahdollisuutta mitata ihmisten ja organisaatioiden suorituskykyä. Standardeja tarvitaan tukemaan prosesseja ja menetelmiä, sekä luomaan hyviä käytäntöjä ja tiedon perustoja. Standardien käytöstä hyötyvät sekä asiakas että yritys. Asiakkaan näkökulmasta standardeja käyttävä yritys vaikuttaa asiantuntijalta, jonka tuotteisiin voi luottaa. Etenkin, jos tuote on hyvin monimutkainen, standardien käyttö lisää asiakkaan luottamusta yritystä kohtaan. Standardien voidaan sanoa myös suojelevan ostajaa. Teknologiatuotteiden lisääntyvä monimutkaisuus aiheuttaa tärkeiden ominaisuuksien pysymisen piilossa ja tulemisen esille vasta ostamisen jälkeen. Standardit voivat tarjota tarkkaa tietoa tuotteiden sopivuudesta tietyille käyttäjille. Yritykselle standardit tarjoavat keinot toiminnan ohjaamiseen ja tiedon perustan luomiseen. Standardit luovat yhteisen termistön, joka helpottaa eri osapuolten kommunikointia tuotantoprosessin aikana [2], [7].

3.1 Ohjelmistotuotannon Standardit

Ohjelmistotuotannon standardit voivat olla tärkeässä roolissa yrityksen toiminnassa. Monet ohjelmistotuotannon standardit ovat kuitenkin luonteeltaan vapaaehtoisia ja eroavat tässä suhteessa joidenkin liiketoiminta-alojen edellyttämistä ns. pakollisista standardeista. Ohjelmistoyrityksen hyödyntävät standardeja, tehdäkseen parempia tuotteita ja parantaakseen tuotteen kilpailukykyä markkinoilla. Standardit voivat parantaa yrityksen liiketoimintaprosessia, jolloin tuotteita voidaan valmistaa pienemmillä kustannuksilla. Standardien roolit ovat hyvin erilaisia. Myös yrityksen tavoitteet standardien soveltamisessa voivat olla hyvinkin erilaisia.

Yritys voi tarvita prosesseja ja mittareita parantaakseen sen kykyä tuottaa kilpailukykyisempiä ohjelmistoja osa-alueilla, joita ovat esimerkiksi laatu, asiakastyytyväisyys, kiertonopeus, tuottavuus, prosessin kypsyys ja teknologia. Standardi voi tarjota kriteerit mittaamiselle ja arvioinnille. Yritys, joka on erikoistunut myymään ohjelmistopalveluita voi tarvita yhtenäistä kehystä määrittelemään hankkijan ja toimittajan vastuut ja suhteet. Standardi voi tarjota kehyksen kommunikoinnille asiakkaan ja toimittajan välillä vähentäen väärinymmärryksiä sekä lyhentäen aikaa, joka kuluu

sopimukseen pääsemiseen. Standardi voi luoda normit, joihin voidaan viitata, mieluummin kuin että ne kuvattaisiin sopimuksessa [7].

Standardi voi esimerkiksi tarjota ytimekkään selityksen monimutkaiselle konseptille. Monimutkaisessa hankkeessa standardit luovat yhteisymmärryksen erottamalla monimutkaiset teknologiset asiat liiketoiminnan näkökulmista. Erityisesti kahden osapuolen neuvotellessa monimutkaisesta asiasta, on hyödyllistä käyttää standardia, joka määrittelee yksityiskohdat. Tämäntyyppisten standardien tavoitteena on luoda yhtenäinen merkintätapa tai terminologia, jonka avulla saavutetaan käsitteiden yhteisymmärrys. [7].

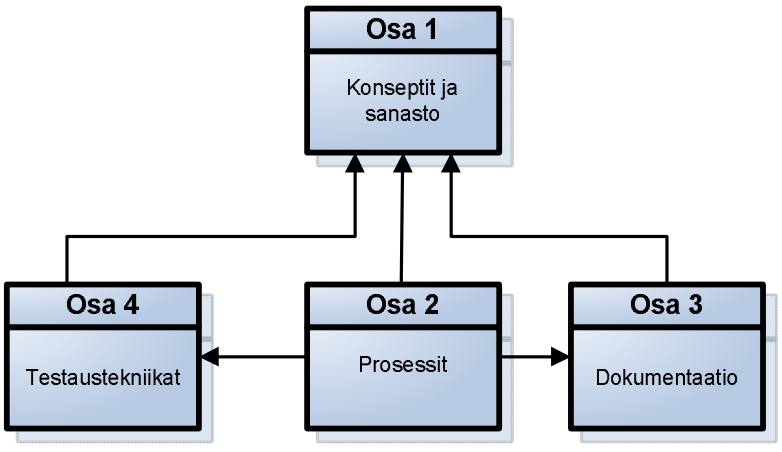
Joskus yritykset haluavat hyödyntää ohjelmistokehitysmenetelmiä ja tekniikoita, jotka on yleisesti todettu tehokkaiksi. Yritys käyttää tällaisia standardeja tehostaakseen toimintaansa ja saavuttaakseen laadullisia parannuksia kun ohjelmistoja kehitetään entistä paremmin, nopeammin ja tehokkaammin. Yritykset voivat tarvita myös tavan osoittaa, että heidän käytäntönsä noudattavat hyväksi havaittuja menetelmiä. Esimerkiksi ISO 9000 laatujärjestelmän noudattaminen antaa yritykselle tietynlaisen laadun leiman [7].

Parantaakseen testauksen suorittamista ja tehokkuutta sekä tuottaakseen laadukkaampia ohjelmistoja, yritys voi ottaa käyttöönsä ohjelmistotuotannon testaus- ja laatustandardeja. Koska standardit ovat luonteeltaan vapaaehtoisia, yritys voi valikoida niistä parhaiten itselleen sopivat osat. Ohjelmistotuotannon laatuun ja testaukseen liittyviä standardeja on olemassa paljon. ISO 9000 sarjan laatujärjestelmästandardien perusolettamus on, että hyvin suunniteltu prosessi tuottaa todennäköisemmin laadukkaampia tuotteita kuin huonosti suunniteltu prosessi. IEEE on määritellyt lukuisia testausta tukevia standardeja. On kuitenkin havaittu tarvetta uusia laatu- ja testausstandardit. Kehitteillä oleva testausstandardi ISO/IEC 29119 ja laatustandardi ISO/IEC 25010 tarjoavat yritykselle apuvälineitä testauksen parempaan suorittamiseen sekä laadun hallintaan. Seuraavassa esitellään näiden standardien pääpiirteet [2], [7].

4. TESTAUSSTANDARDI ISO/IEC 29119

Uutta testausstandardia alettiin kehittää, koska olemassa olevat standardit eivät kata kaikkia ohjelmiston elinkaaren vaiheita. BS 7925.2:n määrittelemät testaustekniikat kattavat pelkästään yksikkötestauksen. Lisäksi korkeamman tason menetelmät, kuten käyttötapaustestaus ja ei-toiminnallinen testaus puuttuvat aiemmista testausstandardeista kokonaan. Myös testauskäytännön ja testausstrategioiden kehittäminen, sekä staattinen testaus eivät ole määriteltyinä olemassa olevissa standardeissa. Lisäksi prosessien ja proseduurien määrittelyissä on havaittu ristiriitoja [2].

Testausstandardi ISO/IEC 29119 jaetaan neljään osaan. Nämä osat ovat 1) konseptit ja sanasto, 2) prosessit, 3) dokumentointi ja 4) testaustekniikat. Ensimmäiseen osioon sisältyy johdatus ohjelmistotestaukseen, testauksen lähestymistavat sekä testauksessa käytettävän sanaston määritteleminen, jotta kaikki osapuolet pystyvät kommunikoimaan keskenään. Toinen osa puolestaan määrittelee prosessit testaukseen suunnitteluun, hallintaan ja toteutukseen. Kolmas osa tarjoaa tarvittavat dokumenttipohjat standardin eri osien dokumentointiin. Neljäs osio määrittelee tekniikat testitapausten suunnitteluun. Tekniikoihin sisältyvät staattiset testaustekniikat (arvioinnit, katselmoinnit), dynaamiset testaustekniikat (musta- ja lasilaatikkotestaus) ja ei-toiminnalliset testaustekniikat (turvallisuus-, suorituskyky-, käytettävyystestaus). Lisäksi neljänteen osaan kuuluvat testauksen mittaustekniikat, kuten testauksen kattavuuden mittaaminen [5].



Kuva 1: ISO/IEC 29119:n osat ja osien väliset suhteet [5]

Testausstandardin tavoitteena on määritellä yleinen prosessimalli, jota mikä tahansa yritys voi hyödyntää suorittaessaan minkä tyyppistä testausta tahansa. Standardi määrittelee testausprosessin, dokumentaation ja tekniikat, jotka soveltuvat kaikkiin ohjelmistokehitysmenetelmiin, kuten

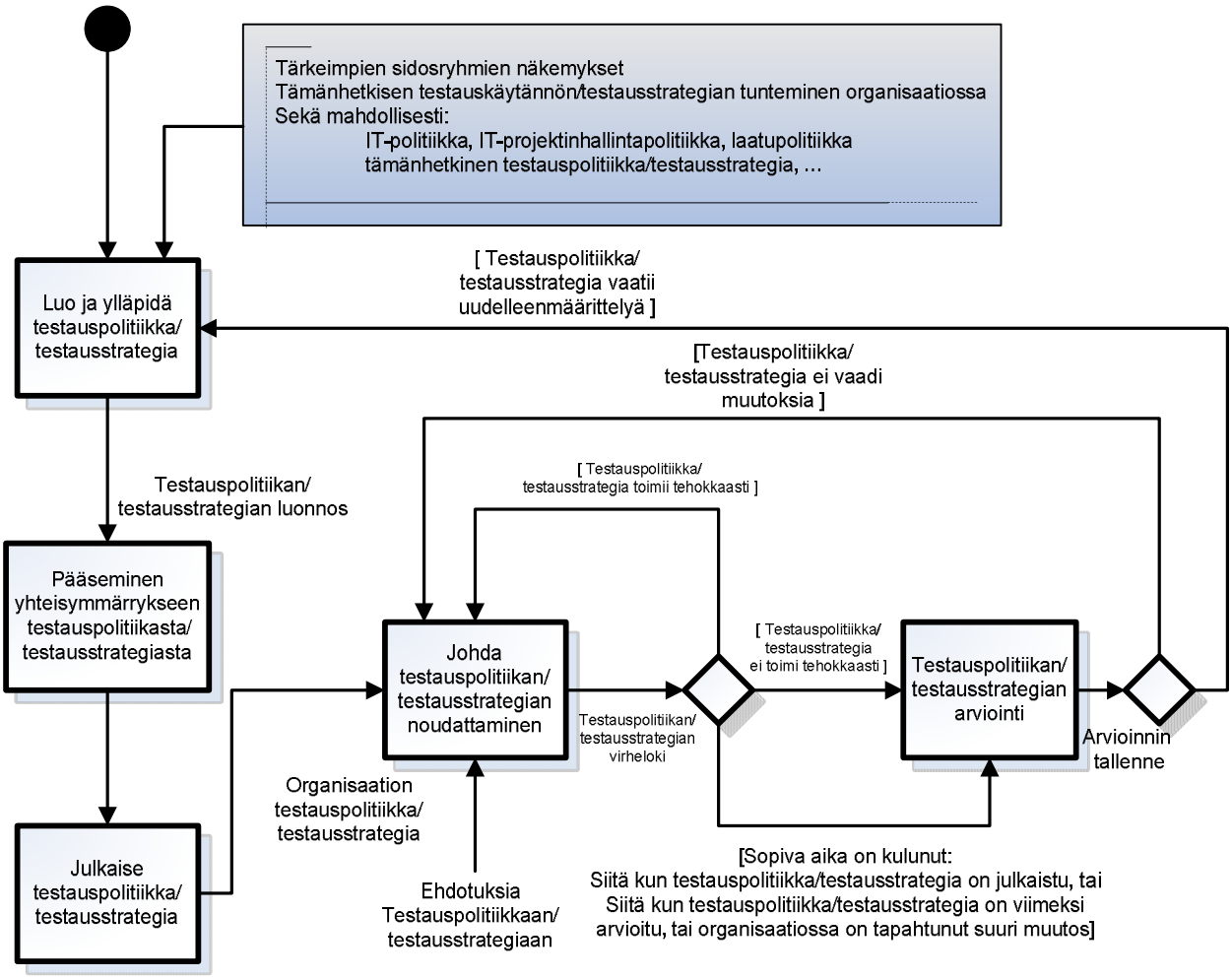
vesiputousmenetelmään, spiraalimenetelmään sekä ketteriin menetelmiin. Standardi tarjoaa kehykset erilaisille testausprosesseille, jotka sopivat käytettäväksi koko ohjelmiston elinkaaren aikana. On kuitenkin huomattava, että kaikki yritykset tai projektit eivät välttämättä tarvitse kaikkia standardin määrittelemiä prosesseja käyttöönsä. Hyödyntääkseen standardia yrityksien on valittava ne standardin osat, jotka sopivat heidän toimintamalliinsa parhaiten.

4.1 ISO/IEC 29119 Osa 2: Prosessit

Standardin määrittelemä testausprosessi koostuu neljästä kerroksesta. Nämä kerrokset ovat 1) testauspolitiikka, 2) testausstrategia, 3) testauksen hallinta ja 4) testauksen tasot. Seuraavassa kuvataan nämä tasot tarkemmin [5].

4.1.1 Testauspolitiikka

Ylimmän tason tavoitteena on määrittää prosessi testauspolitiikan luomiselle ja ylläpidolle. Prosessi tarjoaa kehyksen myös testausstrategian ja projektin testauksen hallinnan perustamiseen, arviointiin ja parantamiseen. Testauspolitiikka on kirjoitettu dokumentti, joka määrittää, mitkä ovat yrityksen tavoitteet testauksessa ottamatta kuitenkaan kantaa siihen, miten testaus suoritetaan. Politiikka määrittää yrityksen testauskäytännöt ja säännöt. Testauspolitiikan pitäisi liittyä yrityksen kaikkiin projekteihin ja kaikentyyppiseen testaukseen. Testauspolitiikkadokumentin tulisi olla saatavilla kaikille henkilöille, joiden toimintoihin sillä on vaikutusta. Dokumentin tulisi olla luonteeltaan ei-tekninen ja pituudeltaan 1-2 sivua. Sen tulisi olla linjattuna yrityksen muiden käytäntöjen, kuten laatupolitiikan kanssa [5].



Kuva 2: Prosessi testauspolitiikan ja testausstrategian luomiseen ja ylläpitoon. [5]

Yllä oleva kuva määrittelee prosessin testauspolitiikan luomiselle. Mikäli organisaatiolla ei ole olemassa olevaa testauspolitiikkaa tai se vaatii suurta uudistamista, tulisi kuvassa 2 kuvattu prosessi käydä läpi. Testauspolitiikan luominen ja ylläpito vaatii tärkeimpien osakkaiden näkemykset sekä tiedon yrityksen nykyisestä testauskäytännöstä. Myös muut yrityksen käytännöt, kuten IT-käytäntö tai laatupolitiikka, voivat olla tukemassa testauspolitiikan luomista. Testauspolitiikan luominen voidaan katsoa testauksen johtohenkilön tehtäväksi, mutta siihen voivat osallistua myös IT-päällikkö, projektipäälliköt sekä kehitysjohtajat. Testauspolitiikan luomisen ja ylläpidon tuloksena syntyy dokumentoitu luonnos testauspolitiikasta. **Liitteessä 1** on esitelty kohdat, joita testauspolitiikka voi sisältää [5].

Testauspolitiikan luomisprosessin seuraavan vaiheen tavoitteena on, että kaikki osakkaat ovat arvioineet luonnoksen ja että heidän näkemykset ja palaute on otettu huomioon. Kun testauspolitiikasta on päästy yhteisymmärrykseen, se on valmis julkaistavaksi. Julkaisuvaiheen

tavoitteena on saattaa kaikille organisaation henkilöille tietoon, että testauspolitiikka on julkaistu, ja sitä tulee käyttää. Testauspolitiikka tulee julkaista sopivassa muodossa, jotta se on helposti kaikkien sitä tarvitsevien henkilöiden saatavilla [5].

Prosessin neljännen vaiheen tavoitteena on varmistaa, että kaikki osakkaat noudattavat testauspolitiikkaa. Testauspolitiikan käyttöä seurataan pitämällä yllä lokia, johon kirjataan esiin tulleita seikkoja testauspolitiikan käytöstä. Mikäli havaitaan, että testauspolitiikkaa ei noudateta, tai siinä on ongelmakohtia, siirrytään vaiheeseen ”testauspolitiikan arviointi”. Mikäli testauspolitiikkaa noudatetaan tehokkaasti ja havaitaan, että se toimii, ei arviointia tarvitse suorittaa. Testauspolitiikan arviointi on kuitenkin syytä suorittaa sopivin väliajoin, jotta se saadaan pidettyä ajan tasalla. Testauspolitiikan arvioinnissa käydään läpi testauspolitiikan loki, ja arvioidaan vaadittavien muutosten suuruutta. Mikäli muutokset ovat pieniä, ne hyväksytään ja dokumentoidaan olemassa olevaan testauspolitiikkaan. Jos muutoksien havaitaan olevat suuria, siirrytään takaisin prosessin ensimmäiseen vaiheeseen ”testauspolitiikan luonti ja ylläpito” [5].

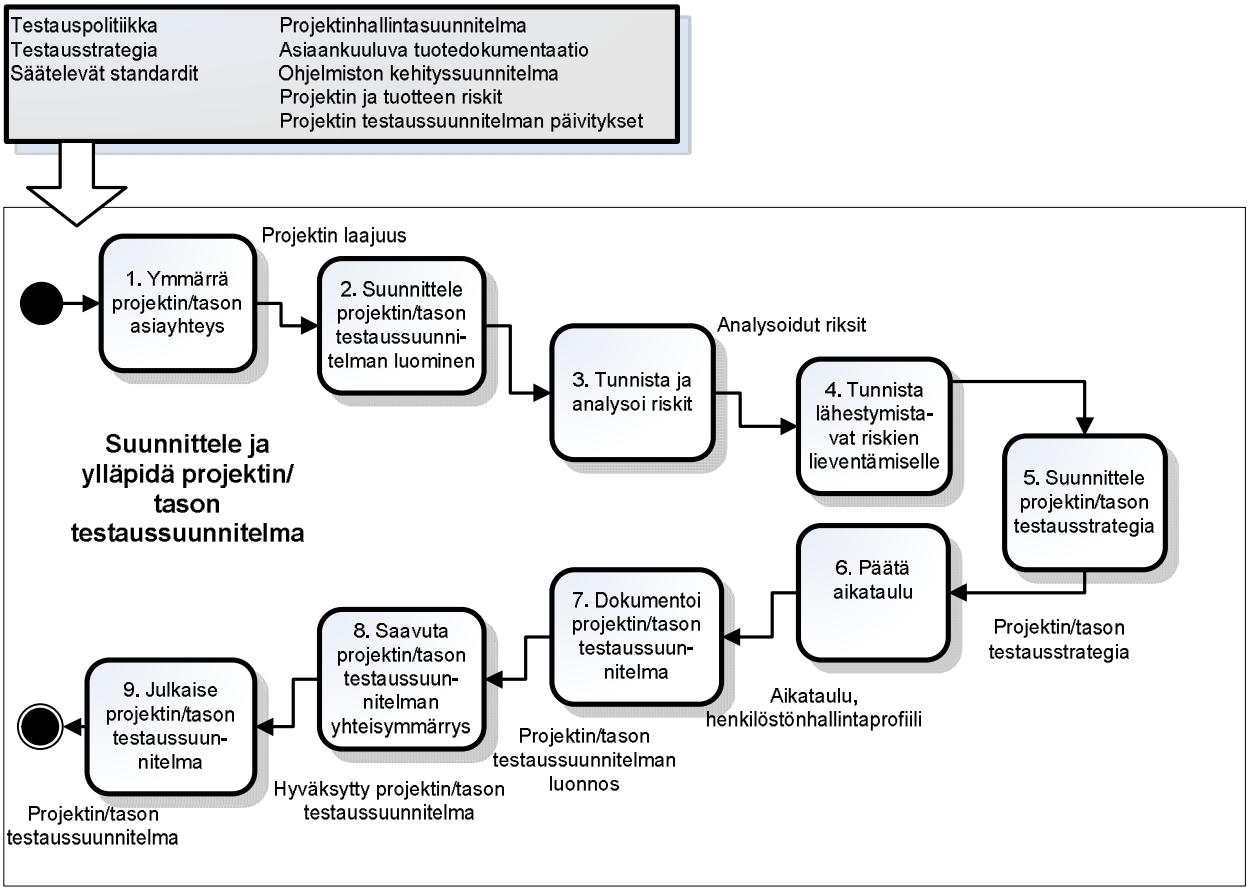
4.1.2 Testausstrategia

Neli-tasoisen prosessimallin toisen tason tavoitteena on määritellä prosessi testausstrategian luomiselle ja ylläpidolle. Yrityksen testausstrategian tarkoituksena on määrittää käytäntö ohjelmistotestaukselle. Testausstrategia on yksityiskohtainen ja tekninen dokumentti, joka määrittelee, miten testaus suoritetaan. Lisäksi testausstrategia määrittää joukon uudelleenkäytettäviä testausohjesääntöjä, joita voidaan noudattaa yrityksen kaikissa projekteissa. Testausstrategia tarjoaa myös kehyksen projektikohtaisen testaussuunnitelmien perustamiseen ja arviointiin [5].

Testausstrategia koskee yrityksen kaikkiin projekteihin liittyvää testausta. Testausstrategian johdonmukainen toteutus johtaa organisaation johdonmukaiseen menettelyyn testauksessa. Yksittäiset projektit voivat poiketa testausstrategiasta, mutta poikkeavuudet tulisi perustella ja dokumentoida. Testausstrategian luominen ja hallinta toteutetaan samaa prosessimallia käyttäen kuin testauspolitiikan kohdalla. Tämä prosessi on esitetty kuvassa 2. Mikäli yrityksellä ei ole erillistä testauspolitiikkaa, testausstrategia voi tarvittaessa sisältää samoja kohtia kuin testauspolitiikkakin. Testausstrategian mahdollinen sisältö on kuvattu **liitteessä 2** [5].

4.1.3 Testauksen hallinta

Testauksen hallinnan prosessi määrittelee, kuinka strategiset päätökset tehdään ja kuinka testaus suunnitellaan, seurataan, hallitaan sekä raportoidaan projektitasolla. Alla olevassa kuvassa on esitetty prosessi projektin testaussuunnitelman luomiseen. Testauksen hallintaan liittyviä tehtäviä on kuvattu **liitteessä 3a** [5].



Kuva 3: Projektin/tason testaussuunnitelman suunnittelu ja ylläpito. [5]

Projektin testaussuunnitelman suunnittelun ja ylläpidon tavoitteena on suunnitella, dokumentoida ja saattaa projektikohtainen testauksen lähestymistapa projektin kannalta oleellisten osakkaiden tietoon. Projektin testaussuunnitelma mahdollistaa resurssien, ympäristöjen ja muiden testauksen vaatimuksien tunnistamisen aikaisessa vaiheessa. Yrityksen tulisi laatia projektin testaussuunnitelma, mikäli sellaista ei ole olemassa, uusi testausprojekti on saatettu alulle tai on havaittu, että projektin testaussuunnitelma vaatii päivittämistä. Projektin testaussuunnitelmaan vaikuttavia tekijöitä ovat testauspolitiikka, testausstrategia, säätelevät standardit, projektisuunnitelma, tuotekohtainen dokumentaatio, projektin ja tuotteen riskit sekä mahdolliset muut tekijät. Pääasiallisessa vastuussa projektin testaussuunnitelman toteuttamisesta on projektin

testauspäällikkö. Muita osallistuvia henkilöitä voivat olla projektipäällikkö, laatupäällikkö ja kehitysjohtaja [5].

Kun projektin testaussuunnitelma luodaan ensimmäisen kerran, kaikki aktiviteetit (kuva 3) tulisi suorittaa. On kuitenkin huomattava, että tarvittaessa joitakin aktiviteetteja joudutaan käymään uudelleen läpi ennen kuin projektin testaussuunnitelma voidaan julkaista. Projektin aikana sitä saatetaan myös joutua muuttamaan sitä mukaa, kuin muut suunnitelmat muuttuvat ja uutta tietoa tulee saataville. Riippuen muutosten suuruudesta joitakin aktiviteetteja joudutaan läpikäymään uudelleen projektin testaussuunnitelman ylläpitämiseksi [5].

Mikäli havaitaan jokin uusi projektiin tai tuotteeseen kohdistuva riski tai jonkin tunnetun riskin luonne muuttuu oleellisesti, projektin testaussuunnitelmaprosessia tulisi jatkaa kohdasta 3 (tunnista ja analysoi riskit). Jos taas uskotaan, että projektin testausstrategiaa täytyy muuttaa jonkin muun syyn kuin riskin takia (esimerkiksi päätetään käyttää uutta testausympäristöä) prosessia tulisi jatkaa kohdasta 5 (suunnittele projektin testausstrategia). Mikäli projektin henkilöstöä tai aikataulutusta päätetään muuttaa jonkin muun syyn kuin riskin takia (esimerkiksi testausvälineiden saatavuus kehityspuolelta on muuttunut), prosessia pitäisi jatkaa kohdasta 6 (määritä henkilöstö ja aikataulu) [5].

4.1.3.1 Projektin asiayhteyden ymmärtäminen

Projektin testaussuunnitelman luomisprosessin ensimmäisen vaiheen tavoitteena on saavuttaa ymmärrys projektin tavoitteista ja ohjelmistokehityksestä, joka tehdään projektin aikana. Lisäksi tavoitteena on saada alkukäsitys testauksesta, jota projekti vaatii. Projektin asiayhteyden ymmärtämiseksi täytyy läpikäydä projektisuunnitelma, josta selviää testauksen budjetti ja resurssit. Projektikohtaisesta tuotedokumentaatiosta, kuten järjestelmän vaatimusdokumentista, saadaan selville projektin laajuus ja mahdollisia vaatimuksia testaukselle. Ohjelmiston kehityssuunnitelmasta saatava tieto puolestaan vaikuttaa testauksen aikatauluihin ja vaiheisiin. Myös projektikohtaiset, säätelevät standardit voivat määrittää erityissääntöjä, jotka voivat vaikuttaa testaukseen. Ensimmäisen vaiheen tuloksena on luonnos projektin testaussuunnitelmasta, joka sisältää ensimmäisen dokumentaation järjestelmän kokonaiskuvasta, testauksen tavoitteet sekä testauksen odotetun laajuuden. Projektin testauspäällikön tulee tässä vaiheessa ymmärtää projektin testauksen vaatimukset korkealla tasolla [5].

4.1.3.2 Testaussuunnitelman hahmottaminen

Seuraavan vaiheen tavoitteena on tunnistaa, suunnitella ja aikatauluttaa aktiviteetit, joita tarvitaan projektin testaussuunnitelman luomiseen. On myös tunnistettava henkilöt, jotka osallistuvat näihin aktiviteetteihin. Henkilöiden osallistuminen voidaan varmistaa organisoimalla työpajoja ja kokouksia [5].

4.1.3.3 Riskien tunnistaminen ja analysointi

Riskien tunnistamisen ja analysoinnin tavoitteena on tunnistaa ja analysoida ohjelmistotestaukseen liittyvät riskit, joita voidaan lieventää testauksen avulla. Riskit voivat liittyä sekä projektiin että itse tuotteeseen. Tämän vaiheen ensimmäinen tehtävä on arvioida niitä riskejä, joita on tunnistettu. Näitä ovat projektin riskirekisterissä olevat riskit tai projektin testaussuunnitelman päivittämisen yhteydessä havaitut riskit. Seuraavaksi tunnistetaan testaukseen liittyvät - tai testauksella lievennettävät lisäriskit. Tämä voi tapahtua arvioimalla tuotespesifikaatioita ja muuta asiaankuuluvaa dokumentaatiota työpajojen, haastatteluiden tai muiden sopivien menetelmien avulla. Havaitut riskit luokitellaan vähintäänkin siten, että erotetaan toisistaan projektiin ja tuotteeseen liittyvät riskit. Lisäksi arvioidaan riskin toteutumisen todennäköisyyttä sekä riskin vakavuutta. Lopulta vaiheen tulokset dokumentoidaan projektin testaussuunnitelman luonnokseen ja tarvittaessa projektin riskirekisteriin [5].

4.1.3.4 Lähestymistavat riskien lieventämiselle

Neljännen vaiheen tavoitteena on etsiä niitä testauksen lähestymistapoja, joilla voidaan lieventää edellisessä vaiheessa havaittuja ja analysoituja riskejä. Tavoitteena on löytää testauksen tyypit, tekniikat, tasot sekä testauksen päättämisen kriteerit, joilla riskejä voidaan lieventää. Neljännen vaiheen tulokset dokumentoidaan projektin testaussuunnitelman luonnokseen [5].

4.1.3.5 Projektin testausstrategian suunnittelu

Seuraavaksi suunnitellaan projektin testausstrategia, jolla toteutetaan lieventämistoimenpiteet riskeille, joilla on korkein prioriteetti. Toimenpiteiden suorittamiseen vaadittavien resurssien, kuten ajan, osaamisen, työkalujen ja testausympäristön tarpeiden määrää tulee arvioida. Myös projektin ja

tuotteen rajoitukset tulee huomioida. Rajoituksia voivat olla esimerkiksi säätelevät standardit, organisaation testauspolitiikan ja testausstrategian noudattaminen, sopimukselliset vaatimukset, projektin ajan ja kustannuksien aiheuttamat vaatimukset, sopivan ammattitaidon omaavien testaajien saatavuus, sekä työkalujen ja testausympäristöjen saatavuus. Projektin testausstrategia dokumentoidaan ja hyväksytetään tärkeimmillä osapuolilla [5].

4.1.3.6 Henkilöstön ja aikataulun määrittäminen

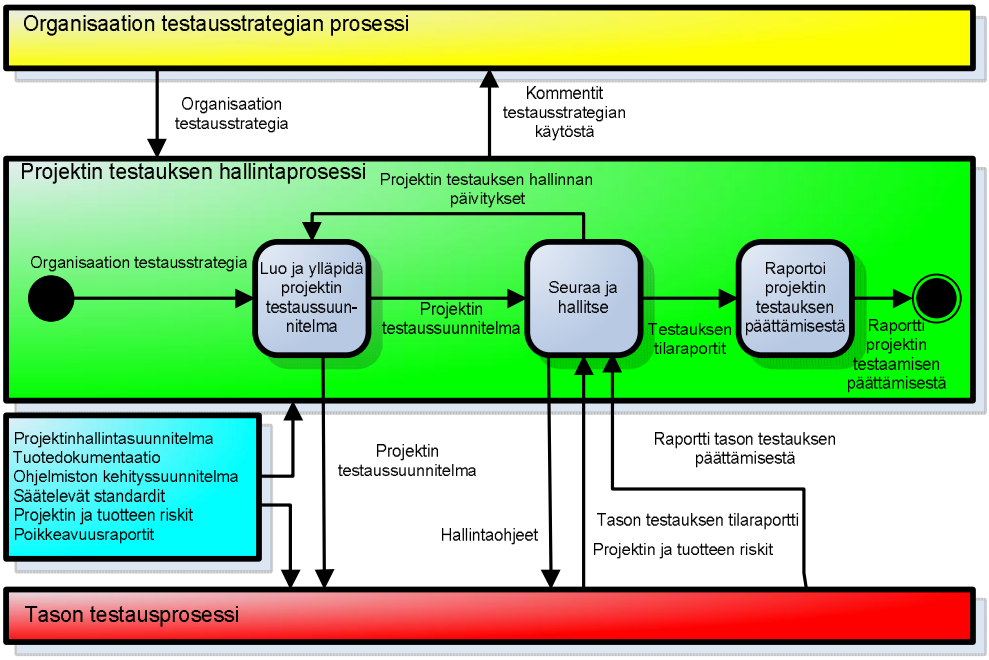
Kun testausstrategia on hyväksytty, määritetään henkilöstönhallintaprofiili ja aikataulu, jotka toteuttavat testausstrategian. On tunnistettava testauksen suorittamiseen vaadittavat roolit ja taidot sekä mahdollinen tarve rekrytoinnille tai kouluttamiselle. Jokaiselle vaadittavalle testauksen tasolle (joka sisältää testauksen tyypit ja tekniikat) määritetään aikataulu. Myös muiden osakkaiden hyväksyntä henkilöstön ja aikataulun suunnitelmalle on hankittava [5].

4.1.3.7 Testaussuunnitelman dokumentointi

Testaussuunnitelma täytyy dokumentoida, jotta eri osapuolten välinen kommunikointi projektissa käytettävästä testauksen lähestymistavasta on mahdollista. Testaukseen liittyvien riskien, niiden lieventämiseen ehdotettujen toimintojen, tunnistetun testausstrategian sekä henkilöstön ja aikataulun määrittelemisen perusteella kirjoitetaan dokumentti, joka on luonnos projektin testaussuunnitelmasta. **Liitteessä 3b** on kuvattu mahdollinen projektin testaussuunnitelman sisältö [5].

4.1.3.8 Yhteisymmärryksen saavuttaminen ja testaussuunnitelman julkaiseminen

Tämän vaiheen tarkoituksena on saada varmuus siitä, että kaikki projektin osapuolet ovat arvioineet projektin testaussuunnitelman luonnoksen, ja että heidän antama mahdollinen palaute on huomioitu. Osapuolien näkemykset voidaan saada selville esimerkiksi pitämällä kokouksia, perustamalla työpajoja tai tekemällä haastatteluja. Mahdolliset konfliktit eri näkemyksien välillä tulee ratkaista ennen projektin testaussuunnitelman hyväksymistä. Testaussuunnitelma julkaistaan sopivassa muodossa siten, että se on saatavilla kaikille projektin Osapuolille. Osapuolille tulee ilmoittaa, että testaussuunnitelma on julkaistu ja sitä tulee noudattaa [5].



Kuva 4: Testauksen hallinta. [5]

Toisena osana projektin testauksen hallintaprosessiin kuuluu ”seuraa ja hallitse” -vaihe, jonka tehtävänä on hallita testausta projektitasolla sekä varmista, että toiminta noudattaa projektin testaussuunnitelmaa ja että suunnitelmaa päivitetään, mikäli tarvetta ilmenee. Tämän vaiheen tavoitteita ovat myös alimman tason testauksen valvominen, testauksen tasojen (esim. järjestelmätestaus) ja tyyppien (esim. suorituskykytestaus) valvominen ja seuranta sekä testauksen etenemisen seuraaminen suunnitelman mukaan. Lisäksi tulee seurata projektiin ja tuotteeseen kohdistuvia riskejä [5].

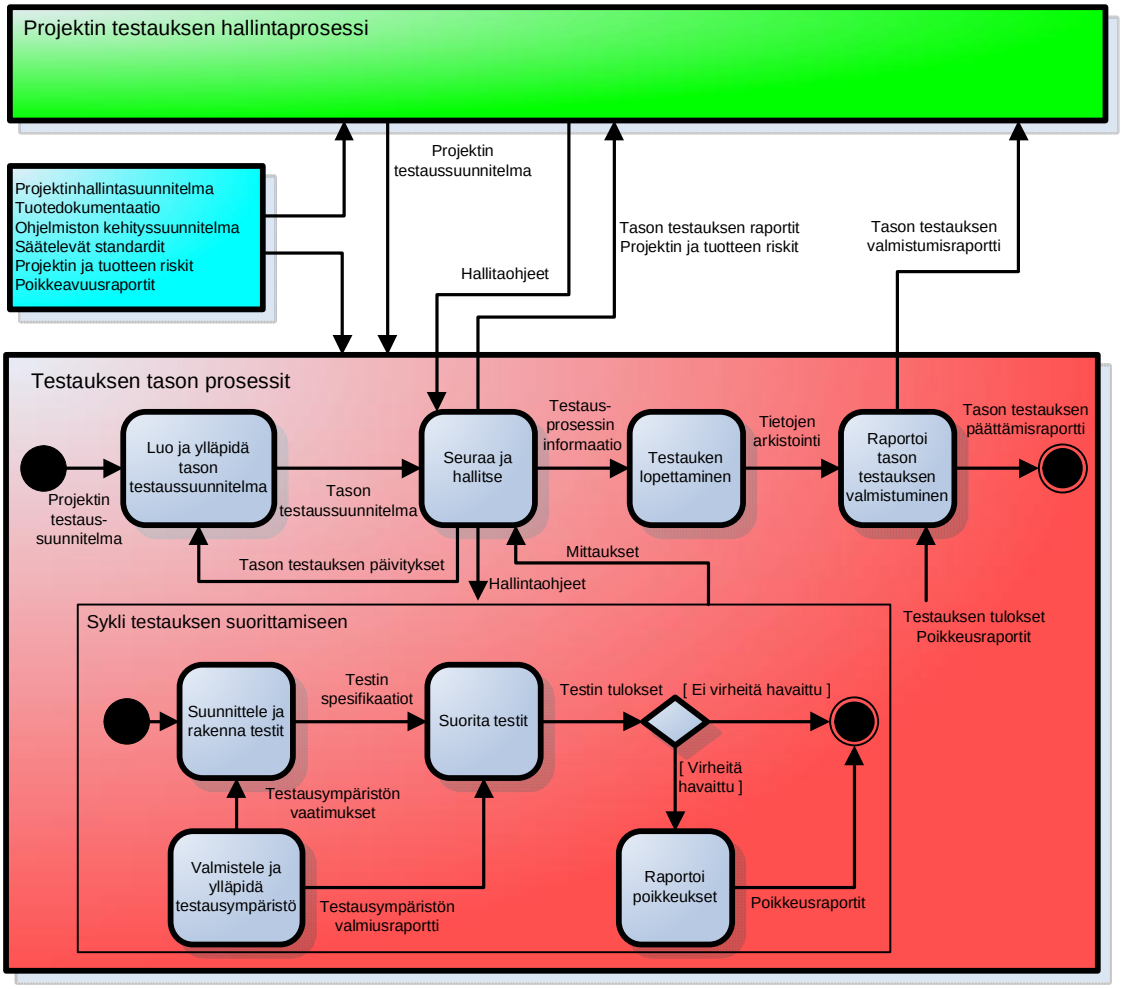
Jokaisen testauksen tason seurantaan tulisi nimittää erillinen vastuuhenkilö. Lisäksi jokaista testauksen tasoa ja tyyppiä tulisi seurata tilaraportoinnin avulla. Raporttien avulla voidaan myös ilmoittaa osapuolille projektin testauksen etenemisestä ja saada lopulta hyväksyntä testauksen päättämiseen. ”Seuraa ja hallitse” -vaiheeseen kuuluu myös havaita testaussuunnitelman päivittämisen tarve sekä mahdolliset muutostarpeet alimman tason testaukseen. Kun projektin testauksen päättämiskriteerit on saavutettu, voidaan siirtyä seuraavaan vaiheeseen testauksen hallintaprosessissa. Projektipäällikkö tekee lopullisen päätöksen siitä, onko projektin testaus suoritettu [5].

Testauksen hallinnan viimeinen vaihe on raportoida projektin testauksen valmistumisesta, jossa testauksen tulokset dokumentoidaan ja esitellään tärkeimmille osapuolille. Dokumentointia varten

on läpikäytävä projektin testaussuunnitelma, tasojen testaussuunnitelmat, testauksen tilaraportit, raportit eri testauksen tasojen valmistumisesta ja raportit poikkeavuuksista. Näistä kerätään oleellinen tieto, joka kirjoitetaan dokumenttimuotoon. Dokumentti sisältää esimerkiksi mitä testattiin, mitä ei testattu (avoimet riskit, poikkeavuudet, joita ei ole korjattu sekä vaatimukset, joita ei ole testattu), poikkeavuudet projektin testaussuunnitelmasta jäljitettävyyttä varten, testaukseen käytetty aika, resurssit, työkalut ja ympäristöt, päivitykset projektin testaussuunnitelmaan sekä poikkeavuudet organisaation testauspolitiikasta ja testausstrategiasta [5].

4.1.4 Testauksen toteutus

ISO/IEC 29119 testausstandardin 4-kerroksisen prosessimallin alin taso määrittelee prosessin testauksen tason (esimerkiksi yksikkö-, järjestelmä-, integraatio- tai hyväksymistestaus) tai tietyn testauksen tyypin (esimerkiksi suorituskyky-, turvallisuus-, tai toiminnallisuustestaus) suorittamiselle. Alin taso sisältää myös prosessit testauksen suunnitteluun, johtamiseen, hallintaan sekä tuloksien raportointiin. Näiden tehtävien suorittaminen on testauksen johtajan vastuulla. Alimman tason toimintoja ovat myös testitapausten analysointi, suunnittelu ja toteutus, poikkeuksien raportointi sekä testausympäristön rakentaminen ja ylläpito [5].



Kuva 5: Prosessit testauksen toteuttamiselle. [5]

Tason testaussuunnitelma luodaan projektin testaussuunnitelman luomiseen käytettävän prosessimallin mukaisesti (kuva 3). Tyypillisen tason testaussuunnitelman sisältö on kuvattu **liitteessä 4**. Seuranta ja hallinta -vaiheen tavoitteena on johtaa tason tai tyypin testaus ja varmistaa, että tason testaussuunnitelmaa noudatetaan ja päivitetään tarpeen mukaan. Lisäksi vaiheen tehtäviä ovat roolien jakaminen testaajille, tason testauksen tilaraporttien luominen, sekä hyväksymisen saaminen testauksen päättämiseen. Tarvittaessa tulee myös toteuttaa ylemmältä tasolta tulevat hallintaohjeet, raportoida uudet riskit ja muutokset projektin testauspäällikölle sekä muuttaa tapaa jolla tietyn tason tai tyypin testaus suoritetaan [5].

4.1.4.1 Testien suorittaminen

Kuvassa 5 on kuvattu sykli testauksen suorittamiseen. Syklin ensimmäisen vaiheen tavoitteena on suunnitella ja dokumentoida testitapaukset, jotka suoritetaan käyttäen apuna tason

testaussuunnitelmaa ja tuotedokumentaatiota. Vaiheeseen osallistuvia henkilöitä ovat testauksen suunnittelijat, tason testauksen vastuuhenkilö, sekä kehittäjät ja testaajat. Vaiheen tuloksena syntyvät dokumentoidut testien spesifikaatiot sekä mahdolliset vaatimukset testausympäristölle. Jotta vaadittu kattavuus saavutettaisiin, ensimmäiseen vaiheeseen saatetaan joutua palaamaan mikäli testiä suoritettaessa tai poikkeuksia raportoidessa ilmenee tarvetta uusille testitapauksille. Testien suorittamisvaiheessa toteutetaan spesifikaation testitapaukset määritellyssä testausympäristössä. Testin tulokset tallennetaan ja niitä verrataan odotettuihin tuloksiin. Lisäksi päätetään läpäisikö testitapaus testin. Testin suorittaminen voidaan keskeyttää, mikäli testitapauksessa tai testausympäristössä havaitaan virhe, tai mikäli testattava kohde on poistunut järjestelmän määrittelystä muutoksen vuoksi. Lisäksi syynä testin suorittamisen keskeyttämiseen voi olla muutokset alimman tason testaussuunnitelmaan. Testien suorittamisvaiheeseen voidaan myös joutua palaamaan uudelleen, mikäli havaitaan, että uusia testitapauksia tarvitaan vaaditun kattavuuden saavuttamiseen. Vain osajoukko testitapauksista voidaan siis suorittaa testauksen suorittamisprosessin yhden kierroksen aikana [5].

Poikkeuksien raportointi tulee suorittaa, mikäli testi epäonnistui, tai jotakin muuta odottamatonta tapahtui. Poikkeus tulee dokumentoida ja tieto täytyy myös välittää tärkeimmille osapuolille, jotta he voivat jäljittää poikkeuksen ja ryhtyä tarvittaviin toimenpiteisiin. Projektin testausjohtaja tekee päätöksen testauksen lopettamisesta. Lopettamispäätös perustuu testausstrategiaan ja testauksen toteutusprosessin synnyttämään informaatioon. Testauksen lopettamisvaiheen tehtävänä on varmistaa, että kaikki testaukseen liittyvä materiaali, kuten testaussuunnitelmat, testispesifikaatiot ja tiedot testausympäristön infrastruktuurista arkistoidaan sopivalla tavalla. Viimeisen vaiheen tavoitteena on dokumentoida kaiken suoritetun testauksen tulokset ja tiedottaa ne tärkeimmille osapuolille. Tason testaussuunnitelmasta, tason testauksen tilaraporteista, poikkeavuusraporteista ja testituloksista kerätään oleellinen tieto, joka dokumentoidaan tason testauksen päättämisraporttiin [5].

5. LAATUSTANDARDIT

Tietokoneita käytetään yhä laajemmalla sovellusalueella. Ohjelmien oikeanlainen toimivuus on usein kriittistä liiketoiminnan menestymisen tai jopa ihmisen turvallisuuden kannalta. Laadukkaiden ohjelmistojen kehittäminen onkin tästä syystä erittäin tärkeää. Kattava ohjelmistotuotteen laadun määrittely ja arviointi on avaintekijä laadun varmistamisessa. Tämä voidaan saavuttaa määrittelemällä sopivat laadun ominaisuudet huomioimalla ohjelmiston käyttötarkoitus. On tärkeää, että jokainen oleellinen ohjelmistotuotteen laadun ominaispiirre määritellään ja arvioidaan yleisesti hyväksyttyjä mittareita käyttäen [4].

ISO/IEC 25010 sisältää uudistuksia aikaisempaan ISO/IEC 9126-1 -standardiin nähden. Turvallisuus, joka aikaisemmin oli toiminnallisuuden aliominaisuus, on muutettu itsenäiseksi ominaisuudeksi. Kannettavuus on jaettu siirrettävyyteen ja yhteensopivuuteen, sekä uudet aliominaisuudet, joita ovat sitkeys, hyödyllisyys, tekninen saavutettavuus, modulaarisuus, uudelleenkäytettävyys ja kannettavuus, on lisätty. Lisäksi joillekin ominaisuuksille ja aliominaisuuksille on annettu tarkemmat nimet. [4]

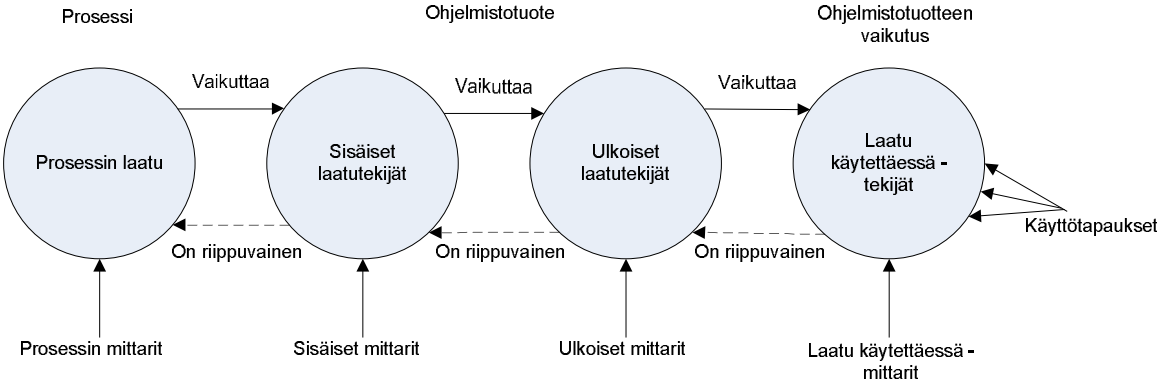
ISO/IEC 25010 -standardia tulisi käyttää yhdessä SQuaRE (Ohjelmistotuotteen laadun vaatimukset ja arviointi) -sarjan standardien kanssa. Näitä standardeja ovat ISO/IEC 25000 – ISO/IEC 25099. ISO/IEC 2501n esittää yksityiskohtaiset laatumallit ohjelmistolle. Se sisältää myös käytännön ohjeistuksen laatumallin käytölle. ISO/IEC 2502n sisältää viitekehyksen ohjelmistotuotteen laadun mittaamiseen. ISO/IEC 2503n auttaa laadun vaatimusten määrittelyssä, ja ISO/IEC 2504n tarjoaa vaatimukset, suositukset ja ohjeet ohjelmistotuotteen arviointiin. ISO/IEC 25050-25099 ovat laajennuksia, jotka tarjoavat lisää työkaluja laadun vaatimusten määrittelyyn. Muita standardeja, jotka tukevat 25010:n käyttöä ovat ISO/IEC 12207 (ohjelmiston elinkaaren prosessit), ISO 9001 (laadun varmistusprosessit) ja ISO/IEC 15504 (ohjelmistoprosessin arviointi) [4], [7].

5.1 ISO/IEC 25010

Laatustandardi määrittelee ohjelmistotuotteen laatumallin, joka koostuu kahdeksasta laadun ominaisuudesta. Nämä ominaisuudet voidaan edelleen jakaa aliominaisuuksiin, joita voidaan mitata sisäisesti tai ulkoisesti. Ohjelmistotuotteen sisäiset ominaisuudet sekä järjestelmän käyttäytyminen määrittelevät sen, miten hyvin ohjelmisto täyttää laatuvaatimukset. Laadulliset ominaisuudet tulevat

julki ohjelmiston käytön yhteydessä. Laatumallia voidaan soveltaa kaikentyyppisiin ohjelmistoihin. Ominaisuudet tarjoavat johdonmukaisen terminologian ohjelmiston laadulle sekä mahdollistavat myös vertailun suorittamisen - miten hyvin lopputuote täyttää siihen kohdistuneet laatuvaatimukset. Laatumallia voidaan myös käyttää muistilistana laadun kattavuuden varmistamiseen [4].

Laatumallia voidaan käyttää tukemaan ohjelmiston määrittelyä ja arviointia ohjelmistotuotantoprosessiin osallistuvien henkilöiden näkökulmista. Nämä henkilöt voivat osallistua ohjelmiston hankkimiseen, vaatimuksien määrittelyyn, kehitykseen, käyttöön, arviointiin, tukemiseen, ylläpitoon tai laadun varmistukseen. Malli antaa tukea kehitysprosessin eri toimintoihin, kuten vaatimuksien tunnistamiseen, vaatimusmäärittelyn kattavuuden varmistamiseen, suunnittelun tavoitteiden tunnistamiseen, testauksen tavoitteiden tunnistamiseen, laadun varmistuksen kriteereiden tunnistamiseen, sekä valmiin ohjelmistotuotteen hyväksymiskriteereiden tunnistamiseen [4].

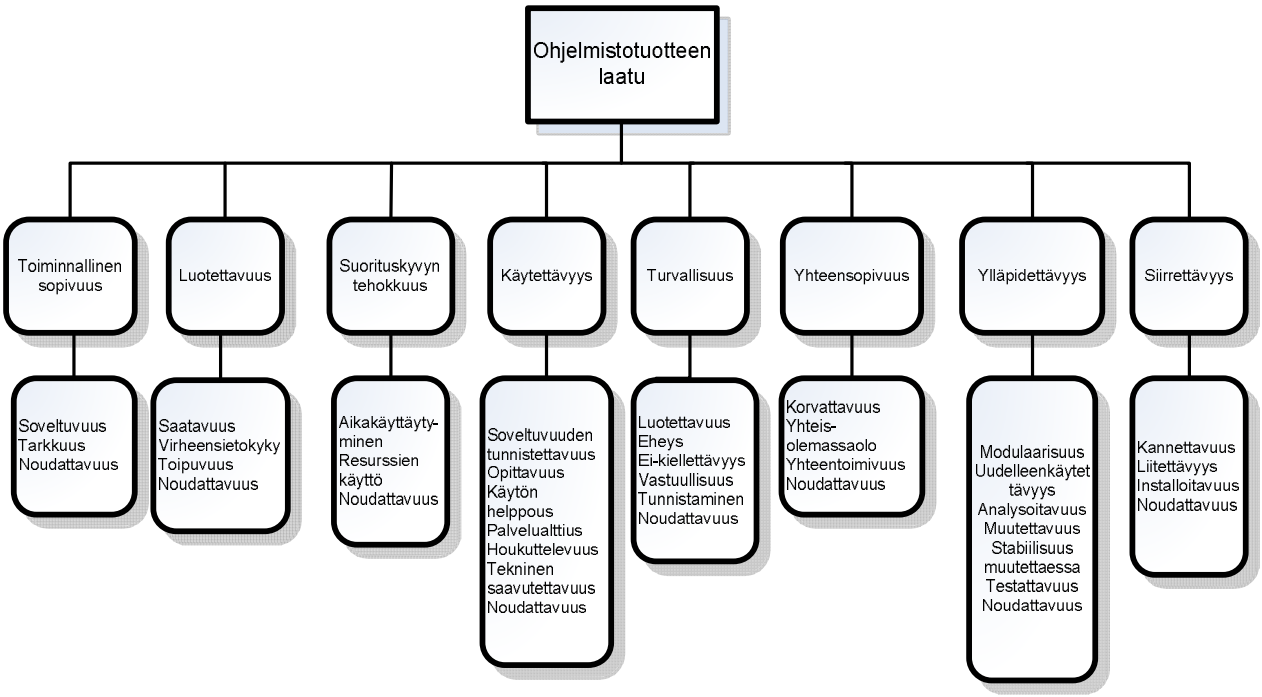


Kuva 6: Laatu ohjelmiston elinkaaren aikana [4]

Järjestelmän laatu määräytyy sen elementtien laadusta ja niiden keskinäisestä vuorovaikutuksesta. Ohjelmiston laatu määräytyy sen mukaan, miten hyvin ohjelmistotuote täyttää siihen kohdistuneet suorat ja epäsuorat tarpeet. Ulkoinen ohjelmiston laatu tarjoaa ”mustalaatikko” –näkymän ohjelmistosta ja osoittaa ne ominaisuudet, jotka liittyvät ohjelman ajamiseen tietokoneessa ja käyttöjärjestelmässä. Sisäinen ohjelmiston laatu tarjoaa ”lasilaatikko” –näkymän ja osoittaa ohjelmiston ominaisuudet, jotka tyypillisesti ovat saavutettavissa kehitysprosessin aikana. Sisäinen ohjelmiston laatu liittyy pääasiallisesti ohjelmiston staattisiin ominaisuuksiin. Sisäinen ohjelmiston laatu vaikuttaa ulkoiseen laatuun, joka taas vaikuttaa laatuun ohjelmistoa käytettäessä. Laatumalli tarjoaa kehyksen, jolla varmistetaan, että kaikki laadun näkökohdat huomioidaan sisäisessä laadussa, ulkoisessa laadussa, sekä laadussa käytettäessä [4].

5.1.1 Ohjelmiston laatumalli

Ohjelmistotuotteen laatu tulisi arvioida määriteltyä laatumallia käyttäen. Laatumallia tulisi myös käyttää, kun ohjelmistolle asetetaan laatuvaatimuksia. Ohjelmistotuotteen laatu tulisi jakaa hierarkkisesti ominaisuuksiin ja aliominaisuuksiin, joita voidaan käyttää tarkistuslistana laatuun liittyvissä asioissa, ja joita voidaan mitata sisäisillä tai ulkoisilla mittareilla. Laatuominaisuuksien suhteellinen tärkeys määräytyy tuotteen ja toimialan mukaan, joten mallia tulisi sovittaa tapauskohtaisesti ennen kuin se otetaan käyttöön. Liiketoimintatavoitteet ja tuotteen sekä suunnitteluprosessin luonne määräävät myös sen, miten resursseja kohdennetaan laadun mittaamiseen [4].



Kuva 7: Ohjelmistotuotteen laatumalli [4]

Jokaiselle laadun ominaisuudelle ja aliominaisuudelle, ohjelmiston kykenevyys määritetään sisäisten laatuominaisuuksien avulla. Näitä ominaisuuksia voidaan mitata. ISO/IEC 9126-3 (jonka tulee korvaamaan ISO/IEC 25022) antaa esimerkkejä sisäisistä mittareista. Laadun ominaisuuksia voidaan mitata myös ulkoisesti. Esimerkkejä ulkoisista mittareista määrittelee ISO/IEC 9126-2 (jonka tulee korvaamaan ISO/IEC 25023). Seuraavassa on kerrottu tarkemmin laatumallin ominaisuuksista ja aliominaisuuksista [4].

5.1.1.1 Toiminnallinen sopivuus

Toiminnallinen sopivuus tarkoittaa ohjelmiston kykyä tarjota funktioita, jotka toteuttavat käytön aikana ohjelmistoon kohdistuvat suorat ja epäsuorat tarpeet. Soveltuvuudella tarkoitetaan, miten hyvin ohjelman tarjoama funktiojoukko soveltuu määriteltyihin tehtäviin ja käyttäjän tavoitteisiin. Tarkkuus puolestaan määrittää ohjelman tuottamien tulosten oikeellisuuden ja vaaditun täsmällisyyden asteen. Noudattavuudella tarkoitetaan astetta, jolla ohjelmisto noudattaa standardeja, merkintätapoja ja lain säädöksiä, jotka liittyvät toiminnallisuuteen [4].

5.1.1.2 Luotettavuus

Kun ohjelmistoa käytetään tietyssä käyttöympäristössä, luotettavuus määrittää ohjelmiston kyvyn ylläpitää määritetty suorituskyvyn taso. Saatavuus kertoo ohjelmiston käyttövalmiuden, kun ohjelmaa halutaan käyttää. Virheensietokyky määrittää suorituskyvyn tason, jonka ohjelma pystyy säilyttämään virheen sattuessa. Toipuvuudella tarkoitetaan ohjelmiston kykyä palauttaa haluttu suorituskyvyn taso tai virhetilanteessa menetetty data. Noudattavuus määrittää asteen, jolla ohjelmisto noudattaa standardeja, merkintätapoja ja lain säädöksiä, jotka liittyvät luotettavuuteen [4].

5.1.1.3 Suorituskyvyn tehokkuus

Suorituskyvyn tehokkuus määrittelee, millaisen suorituskyvyn ohjelmisto tarjoaa käytetyistä resursseista ja tietyistä olosuhteista riippuen. Resursseilla tarkoitetaan toisia ohjelmia ja järjestelmän laitteistokokoonpanoa. Aikakäyttäytyminen kuvaa ohjelmiston vaste- ja käsittelyaikoja ja suoritustehoa, kun ohjelma suorittaa tiettyjä toimintoja tietyissä olosuhteissa. Resurssien kulutus puolestaan kuvaa ohjelmiston kykyä käyttää sopivia resursseja sopivassa suhteessa toimintoja suorittaessaan. Noudattavuus määrittää asteen, jolla ohjelmisto noudattaa standardeja, merkintätapoja ja lain säädöksiä, jotka liittyvät suorituskykyyn [4].

5.1.1.4 Käytettävyys

Käytettävyydellä tarkoitetaan ohjelmiston kykyä tulla ymmärretyksi, opituksi ja käyttäjälleen houkuttelevaksi, kun ohjelmistoa käytetään tietyissä olosuhteissa. Sopivuuden tunnistettavuus kuvaa, miten hyvin käyttäjä voi tunnistaa ohjelmiston olevan hänen tarpeisiinsa sopiva. Opittavuus puolestaan määrittää, miten hyvin käyttäjä oppii ohjelmiston käytön. Helppokäyttöisyys määrittelee ohjelman käytön ja hallinnan helppouden tason. Helppokäyttöisyyteen saattavat vaikuttaa myös muut tekijät, joita ovat sopivuus, muunneltavuus, liitettävyys, ja installoitavuus. Palvelualttius kuvaa ohjelmiston kyvyn tarjota apua silloin, kun käyttäjä tarvitsee ohjeistusta. Palvelualttiuteen sisältyy myös ohjeiden löydettävyys, kattavuus ja tehokkuus. Houkuttelevuudella viitataan niihin ohjelmiston tekijöihin, jotka lisäävät käyttäjän mielihyvää ja tyytyväisyyttä. Näitä ominaisuuksia ovat esimerkiksi värien käyttö ja graafinen ulkoasu. Tekninen saavutettavuus määrittelee ohjelmiston käytettävyyden ihmisille, joilla on jokin pysyvä vamma. Myös ikääntymisen katsotaan kuuluvan seikaksi, joka huomioidaan teknisen saavutettavuuden kohdalla. Noudattavuus määrittelee ohjelmiston kyvyn noudattaa standardeja, merkintätapoja, tyyliohjeita ja säädöksiä, jotka liittyvät käytettävyyteen [4].

5.1.1.5 Turvallisuus

Turvallisuuteen liittyy järjestelmän suojaaminen vihamieliseltä tai vahingolliselta pääsyltä, käytöltä, muuntamiselta, tuhoamiselta tai paljastumiselta. Luotettavuudella tarkoitetaan sitä tasoa, jonka ohjelmisto tarjoaa suojautuakseen luvatonta käyttöä vastaan, joka voi olla tarkoituksellista tai tarkoituksetonta. Eheys määrittää tason tiedon tarkkuuden ja täydellisyyden turvaamiselle. Ei-kiellettävyydellä tarkoitetaan sitä, että suoritetut tapahtumat voidaan todistaa tapahtuneeksi, eikä niitä voida myöhemmin kieltää. Vastuullisuudella puolestaan tarkoitetaan kykyä jäljittää tietyn itsenäisen yksikön suorittamia toimintoja. Tunnistaminen määrittää asteen, kuinka hyvin jokin resurssi voidaan todistaa väitetyksi. Noudattavuus määrittelee asteen, jolla ohjelmistotuote noudattaa turvallisuuteen liittyviä standardeja, merkintätapoja ja säädöksiä [4].

5.1.1.6 Yhteensopivuus

Yhteensopivuudella tarkoitetaan kahden tai useamman ohjelmistokomponentin kykyä vaihtaa tietoa tai suorittaa niiden tarvitsemia toimintoja, siten että ne jakavat saman laitteisto- tai ohjelmistoympäristön. Korvattavuudella tarkoitetaan ohjelman vaihdettavuutta toiseen, samassa ympäristössä toimivaan saman tarkoituksen täyttävään ohjelmaan. Yhteis-olemassaolo määrittää tason, jolla kaksi itsenäistä ohjelmaa voivat toimia keskenään samassa ympäristössä käyttäen samoja resursseja ilman haittavaikutuksia. Yhteentoimivuudella puolestaan tarkoitetaan tasoa, jolla ohjelmistoa voidaan käyttää yhteistyössä muiden ohjelmien kanssa. Noudattavuudella tarkoitetaan ohjelmiston kykyä noudattaa standardeja, merkintätapoja ja säädöksiä, jotka liittyvät yhteensopivuuteen [4].

5.1.1.7 Ylläpidettävyys

Ylläpidettävyys määrittelee ohjelmistotuotteen muokattavuuden. Ohjelmistoon toteutettavat muutokset voivat olla korjauksia, parannuksia tai ohjelman mukauttamista ympäristön, vaatimusten tai toiminnallisen määrittelyn muutoksiin. Modulaarisuus tarkoittaa ohjelmiston koostumusta komponenteista siten, että muutos yhteen komponenttiin ei aiheuta suuria vaikutuksia muihin komponentteihin. Uudelleenkäytettävyys määrittele, miten hyvin ohjelmiston osia voidaan käyttää muissa ohjelmistoissa tai muiden osien rakentamisessa. Analysoitavuus puolestaan kertoo muokattavien osien tunnistamisen tai asteen, jolla ohjelmistotuotetta voidaan määritellä puuteiden ja virheiden suhteen. Muutettavuudella tarkoitetaan ohjelman muokkaamisen helppoutta. Vakaus muutettaessa puolestaan määrittelee, kuinka hyvin ohjelma voi välttää odottamattomat vaikutukset, kun siihen tehdään muutoksia. Testattavuus määrittää, kuinka hyvin muutettu ohjelma on testattavissa. Noudattavuus puolestaan tarkoittaa, kuinka hyvin ohjelmistotuote noudattaa standardeja, merkintätapoja ja säädöksiä ylläpidettävyydessä [4].

5.1.1.8 Siirrettävyys

Siirrettävyys määrittää, miten hyvin ohjelmistotuote voidaan siirtää ympäristöstä toiseen. Kannettavuus kuvaa järjestelmän tai komponentin siirtämisen helppoutta laite- tai ohjelmistoympäristöstä toiseen. Liitettävyys puolestaan tarkoittaa adaptoituvuutta erilaisiin

määriteltyihin ympäristöihin ilman lisätyötä. Asennettavuus määrittää tason ohjelmiston asennukseen ja poistoon tietyssä ympäristössä. Noudattavuus kuvaa, miten hyvin ohjelmistotuote noudattaa siirrettävyyteen liittyviä standardeja, säädöksiä ja merkintätapoja [4].

6. LAADULLINEN TUTKIMUS

Laadullisen tutkimuksen avulla tutkitaan todellista elämää. Tutkimuksen kohteena on usein ihmisen toiminta ja vuorovaikutus osana jotakin suurempaa kokonaisuutta. Laadullisen tutkimuksen periaatteiden mukaisesti todellisuus on moninainen, ja sitä ei voi pilkkoa osiin. Tapahtumat ovat toisistaan riippuvia ja tapahtumien väliltä voidaan löytää monenlaisia suhteita. Laadullinen tutkimus suoritetaan usein luonnollisessa ympäristössä, jossa tutkija ei pysty kontrolloimaan kaikkia lähteisiin liittyviä virheitä. Laadullisten tutkimusmenetelmien tavoitteena on vastata kysymyksiin, mikä ilmiö on, millainen ilmiö on, ja mitkä muut ilmiöt vaikuttavat ilmiöön. Tutkimusaineisto kootaan todellisen elämän tilanteissa. Tiedon keruun välineenä suositaan ihmistä, eli tutkija luottaa tekemiinsä haastatteluihin ja omiin havaintoihinsa enemmän kuin mittaamiseen. Tutkija ei lähde testaamaan hypoteeseja vaan käyttää induktiivista päättelyä ja analyysia tarkastellessaan aineistoa monipuolisesti ja yksityiskohtaisesti. Aineistonkeruumenetelminä esimerkiksi teemahaastattelut ja ryhmähaastattelut sekä erilaisten tekstien analysointi ovat laadullisen tutkimuksen kannalta sopivia. Kohdejoukko eli tutkittavat henkilöt valitaan laadullista tutkimusta tehdessä tarkoituksenmukaisesti. Lisäksi ominaista on, että tutkimussuunnitelma ja tutkimuskysymykset voivat muotoutua uudelleen tutkimuksen kuluessa. Tutkija käsittelee jokaista tutkittavaa tapausta ainutlaatuisena ja tulkitsee myös tapauksesta saamaansa aineistoa sen mukaisesti [6].

6.1 Aineistopohjainen menetelmä

Tässä kandidaatintyössä käytettiin aineistopohjaista -menetelmää haastatteluaineiston analysointiin. Aineistopohjainen menetelmä koostuu kolmesta datan analysoinnin vaiheesta. Ensimmäisessä vaiheessa (open coding, avoin koodaus) kerätään aineistosta tutkimuksen kannalta kiinnostavat luokat. Toinen vaihe (axial coding, aksiaalinen koodaus) sisältää luokkien välisten yhteyksien etsimisen. Kolmannen vaiheen (selective coding, valikoiva koodaus) aikana etsitään ja kuvataan tutkimusaineiston ydinluokka [6], [11].

6.1.1 Avoin koodaus

Avoimella koodauksella on tarkoitus tuottaa käsitteitä, sekä nimetä tapahtumia ja ilmiöitä. Käsitteitä ryhmitellään luokkiin, jonka jälkeen luokkien ominaisuuksia kehitetään edelleen aksiaalisen koodauksen aikana. Avoin koodaus aloitetaan yleensä heti, kun datan kerääminen on

aloitettu. On mahdollista, että avoimen koodauksen aikana datasta nousee esiin uusia käsitteitä, joiden avulla voidaan suunnata tutkimusaineiston keruuta [6].

6.1.2 Aksiaalinen koodaus

Aineistopohjaisen menetelmän toisessa vaiheessa aineistot ryhmitellään etsimällä luokkien välisiä yhteyksiä tarkastelemalla ilmiöiden kausaaliehtoja, kontekstia, toiminta- ja vuorovaikutusstrategioita sekä niiden seurauksia. Kausaaliehdolla tarkoitetaan tapahtumaa tai tilannetta, joka johtaa luokitellun ilmiön toteutumiseen. Kontekstilla tarkoitetaan vallitsevia olosuhteita, jossa ilmiö tapahtuu. Toiminta- ja vuorovaikutusstrategiat ovat strategioita, joiden avulla ilmiötä on suunniteltu hoidettavaksi ja hallittavaksi. Seuraukset puolestaan ovat lopputuloksia tai suoritteita vuorovaikutuksesta tai toiminnasta. Aineistoa kerätään, käsitteellistetään ja jaetaan luokkiin niin kauan, kunnes uusia käsitteitä, ominaisuuksia tai dimensioita ei löydy. Tällöin on saavutettu aineiston teoreettinen kylläisyys ja aksiaalisen koodauksen lopetusehto [6].

6.1.3 Valikoiva koodaus

Valikoivan koodauksen tehtävänä on löytää aineistosta ydinluokka. Tässä vaiheessa tutkijalle selviää, mistä tutkimuksessa on kysymys eli teoria syntyy aineiston perusteella. Tässä vaiheessa tutkimuksen tulokset voidaan raportoida [6].

6.1.4 Teorian tuottaminen

Datan analysointivaiheiden jälkeen, aineistopohjaisen menetelmän neljännessä vaiheessa kategoriat yhdistetään ydinkategorian ympärille. Tutkija tiivistää oman näkemyksensä tutkimuksesta perustuen aksiaalisen koodauksessa selvinneisiin ilmiöiden välisiin suhteisiin. Tutkijalla on paljon erilaisia työkaluja tutkimusaineiston jäsentämiseen. Esimerkiksi erilaiset diagrammit, taulukot ynnä muut paljon havaintoja yhteen näkymään tiivistävät esitystavat ovat hyviä työkaluja laadullisen aineiston esittämiseen [6].

7 TULOKSET

Tässä työssä käytetään termin ”yritys” sijasta termiä ”organisaatioyksikkö”(organizational unit, lyh. OU). ISO/IEC 15504-1 –standardin määritelmän mukaisesti OU on tarkastelun kohteena oleva organisaation osa, joka on tyypillisesti osa suurta organisaatiota, kuten esimerkiksi testausosasto. Pienen yrityksen kohdalla termi voi puolestaan tarkoittaa koko yritystä. OU suorittaa yhtä tai useampaa prosessia yhdenmukaisella prosessin viitekehyksellä ja sen toimintaa ohjaa yhdenmukainen joukko liiketoiminnan tavoitteita. Tätä termiä käytetään, jotta saataisiin normalisoitua yrityskoon vaikutus, jolloin yritysten välillä voidaan suorittaa vertailua niiden kokoerosta huolimatta [12].

7.1 Yrityshaastattelut

MASTO -hankkeen haastatteluissa haastateltiin suomalaisia yrityksiä, joiden liiketoimintaan liittyy keskeisesti ohjelmistokehitys. Yritykset olivat kooltaan muutaman henkilön pienyrityksestä aina satoja tai jopa tuhansia ihmisiä työllistäviin suuryrityksiin. Haastattelukierroksia tehtiin useita, mutta tässä kandidaatintyössä käsitellään kahden ensimmäisen kierroksen haastatteluita. Ensimmäisellä kierroksella haastateltavia organisaatioyksiköitä oli 12. Haastatteluilla pyrittiin selvittämään, miten ohjelmiston kriittisyys ja organisaatioyksikön liiketoimintasuuntautuneisuus vaikuttavat ohjelmistotestauksen kustannuksiin ja ohjelmiston laatuun. Haastattelulla pyrittiin selvittämään myös, miten tuotantomenetelmät, standardit ja ulkoistetut toiminnot vaikuttavat testauksen tehokkuuteen, kustannuksiin sekä lopputuotteen laatuun. Organisaatioyksiköiltä kysyttiin heidän käyttämiään ohjelmistojen suunnittelu- ja toteutusmenetelmiä, testauksen strategioita ja resursseja. Lisäksi kysyttiin, hyödyntävätkö organisaatioyksiköt standardeja, ja mitä osia toiminnassaan organisaatioyksiköt ulkoistavat. Omana aiheenaan käsiteltiin myös testausautomaatiota ja testaukseen käytettäviä työkaluja. Organisaatioyksiköiltä kysyttiin myös, mitä laatuominaisuuksia he painottavat, sekä millä tavalla muut tekijät, kuten asiakkaan, tuotteen kriittisyyden ja muualta hankittujen komponenttien vaikutusta laatuun. Toisella kierroksella haastateltavia organisaatioyksiköitä oli yhteensä 26. Toisen kierroksen haastatteluissa käytiin läpi kyselylomake, jossa organisaatioyksiköt olivat arvioineet omaa toimintaansa MASTO -projektin teemojen osalta.

Seuraavissa kappaleissa arvioidaan organisaatioyksiköiden valmiutta soveltaa uusia ohjelmistotuotannon testaus- ja laatustandardeja. Aluksi käsitellään yleisesti standardien soveltuvuutta käytäntöön organisaatioyksiköiden näkökulmasta. Tämän jälkeen käsitellään testausstandardin soveltuvuutta arvioimalla organisaatioyksiköiden edistyksellisyyttä testauksessa. Lisäksi esitetään, mihin neliosaisen prosessimallin osaan organisaatioyksiköt panostaisivat testauksessaan. Tämän jälkeen arvioidaan yksityiskohtaisemmin testausstandardin soveltuvuutta seuraavien kohtien osalta:

i. Henkilöstö ja resurssit
ii. Testauksen elinkaari
iii. Testausstrategia, sen ongelmat ja kehityskohteet
iv. Kriittisyyden vaikutus testausstrategiaan
v. Testauksen ulkoistaminen ja osaamisen tarve
vi. Testausautomaatio ja työkalut
vii. Testauksen mittaaminen

Testausstandardin jälkeen siirrytään käsittelemään laatustandardia. Organisaatioyksiköiden tärkeimmiksi valitsemien laatuominaisuuksien painottuminen kuvataan diagrammien avulla. Lisäksi laatustandardin soveltuvuutta arvioidaan analysoimalla, miten hyvin organisaatioyksiköt huomioivat laatuominaisuuksia toiminnassaan.

7.2 Standardien soveltaminen käytäntöön

Standardien soveltaminen käytäntöön ei ole aina itsestään selvää. Valmiit standardit saatetaan kokea liian raskaiksi ja turhaa työtä lisääviksi. Kukaan haastatelluista organisaatioyksiköistä ei osoittautunut noudattavansa orjallisesti jotakin tiettyä standardia. Sen sijaan monilla organisaatioyksiköillä on käytössään talon sisällä muodostunut ohjeisto, johon on mahdollisesti otettu osia joistakin olemassa olevista standardeista, kuten esimerkiksi ISO-9000 sarjasta tai CMMI:stä. Uuden standardin soveltaminen vaatii aina muutoksia OU:n totuttuihin toimintatapoihin. Esimerkiksi Case 5:n haastateltava sanoo seuraavaa:

> ”Tällä organisaatiokoolla tällaisen jonkun standardin tai menetelmän käytöllä - mitä näitä nyt on kaikennäköisiä - me emme saavuttaisi mitään muuta kuin hillitön sekasotku. ”

Organisaatioyksikön sisäisiä, totuttuja käytäntöjä voi olla vaikeaa muuttaa. Onhan yleisesti todettu, että muutoksia vastustetaan huomattavasti enemmän kuin puolletaan. Standardia käyttöön otettaessa nouseekin erityisen tärkeään asemaan johdon kyky vakuuttaa ihmiset muutoksen hyödyllisyydestä ja välttämättömyydestä. Case 9:n haastateltava kuvasi, miten heillä suhtaudutaan muutoksiin:

> "Ja tietysti se että mitä hallinto muuttaa niin kyllähän sillä, yleensä mehän niin kuin itku kurkussa tullaan niistä, koulutustilaisuuksista ulos että kun, tällaisia muutoksia on nyt tulossa tehtäväksi. Ja ne pakosta tulevat muutokset on ihan hankalimpia.. "

Mikäli organisaatioyksikkö päättää luoda standardin avulla uuden käytännön, jota toiminnassa hyödynnetään, tulisi tämä käytäntö dokumentoida, ja sen tulisi olla saatavilla kaikille OU:ssa työskenteleville henkilöille. Esimerkiksi Case 9 on luonut sisäisen ohjeiston, mutta sitä ei ole kuitenkaan dokumentoitu. On selvää, että dokumentoimattoman käytännön noudattaminen on tehottomampaa, kun ohjeisto ei ole saatavilla fyysisessä muodossa. Lisäksi tämä voi vaikeuttaa uusien henkilöiden perehdyttämistä talon toimintatapoihin. Mikäli OU luo esimerkiksi testauspolitiikan, on koko käytännön luominen turhaa, ellei OU:ssa ole vastuuhenkilöä, joka seuraisi että politiikkaa noudatetaan. Esimerkiksi Case 1 teki laatusuunnitelman, mutta heillä ei ollut henkilöä, joka olisi valvonut, että suunnitelmaa noudatettaisiin. ISO/IEC 29119:n määrittelemät testausprosessit vaativat jatkuvaa seurantaa, jotta varmistutaan siitä, että politiikkaa todella noudatetaan, ja jotta käytäntöä voidaan muuttaa tarvittaessa.

Joskus OU:n sisäisten toimintatapojen muuttaminen voi vaatia paljon aikaa. Case 7:n haastateltava totesi, että heillä täysin uuden toimintamallin käyttöönotto ei sujunut helposti:

> "Sanotaan, että keskimäärin kahdeksan vuotta vie, että se alkaa toimimaan yhtä hyvin kuin edellinen malli, oli se sitten mikä hyvänsä, tai toimi se sitten miten hyvänsä.. "

Lähes kaikilla haastatelluilla organisaatioyksiköillä oli sisäinen toimintamalli. Ainoastaan kolme organisaatioyksikköä sanoi, että he eivät käytä mitään kehitysprosessia tai testausta tukevia standardeja. Kaksi haastatelluista organisaatioyksiköistä totesi suoraan, että sisäistä toimintamallia tulisi kehittää.

Esimerkki onnistuneesta sisäisestä käytännöstä välittyy Case 11:n kohdalla. OU käyttää dokumentoinnissa apunaan standardia. Haastateltava ei muista perustuuko se johonkin olemassa olevaan standardiin. Tätä käytäntöä on kuitenkin kehitetty edelleen talon sisällä. Käytäntö

määrittelee OU:n työskentelyprosessin, kuten suunnittelun, testauksen ja muistioiden pohjat. OU on saanut luotua standardin avulla yhtenäiset säännöt, ja tiedonkulku henkilöiden välillä sujuu ongelmitta. Haastateltava toteaa, että etenkin isommassa projektissa tällainen menettelytapa on ehdoton. Myös Case 10 ilmoitti käyttävänsä standardia toimintansa tukena:

> "Meillä on joku standardi käytössä. Mä en kyllä muista sen nimeä, sen. Pitäis muistaa. "
>
> "Voi se olla joku tällänen ISO joku? "
>
> "Se on joku ISO joku, joku numero perässä. Sitä se on ja sen mukaan mennään kyllä aika tiukasti ja varmaan on mentäväkin kun niitä auditoidaan jatkuvasti, niin tota, niin, niin. Mä pidän ite sitä aika hyvänä, koska siitä me saadaan semmoset aika selkeät, niinku guideline:t tälle meidän toiminnalle ja tota niin, se on ehkä myös sellanen kun näistä auditoinneista aina välillä tulee sitte huomautuksia, niin se saa meitä ohjaamaan sitte taas meidän toimintaa niiltä osin, mitä me ehkä ollaan, tai on saattanu päästä jopa lipsumaan tai sitten ei olla vaan huomioitu tarpeeks ja näin poispäin, et kyl mä pidän niit ihan. "

Merkittävänä tekijänä ISO/IEC 29119 ja 25010 -standardien – tai minkä tahansa uusien standardien soveltuvuuteen vaikuttaa OU:n sisäinen ilmapiiri – miten OU:ssa suhtaudutaan muutoksiin. Lisäksi johdon taidoilla on suuri merkitys, kun uusia käytäntöjä luodaan. Myös OU:n aikaisempi kokemus standardeista lienee soveltuvuuteen vaikuttava tekijä. Mikäli OU:lla on aiempaa kokemusta standardien käytöstä, on heidän helpompi soveltaa myös ISO/IEC 29119 ja 25010 -standardeja toiminnassaan.

8 ISO/IEC 29119 - SOVELTUVUUS

ISO/IEC 29119 testausstandardin soveltuvuuden osalta haastatellut OU:t ovat hyvin erilaisessa asemessa. Yrityksen koko ja ikä vaikuttavat merkittävästi siihen, kuinka pitkälle OU:n testausprosessit ovat kehittyneet. Tämä vaikuttaa myös siihen, miltä osin OU:lla on tarve soveltaa testausstandardia. Esimerkiksi eräässä haastatellussa OU:ssa testaus suoritetaan ilman minkäänlaista testausstrategiaa ja ilman erillisiä testaajia, kun taas toisella OU:lla on oma testausorganisaatio ja testauksen suunnittelu ja toteutus ohjautuu projektikohtaisesti, ja lisäksi testauksen elinkaari noudattelee kehityksen elinkaarta. Jälkimmäiselle OU:lle testausstandardilla on huomattavasti vähemmän lisättävää kuin ensimmäiseksi mainitulle.

Ensimmäisellä haastattelukierroksella organisaatioyksiköiltä kysyttiin, mihin testauksen tasoon he panostaisivat. Yhdeksän OU:n mielestä tärkein kohde oli joko testausstrategia tai testauksen hallinta, ja näistä yhdeksästä OU:sta neljä valitsi sekä strategian että hallinnan tärkeimmiksi kehityskohteiksi. Neljä OU:ta valitsi testauksen toteutuksen tärkeimmäksi. Testauspolitiikkaa ei painottanut yksikään OU. Tämä johtunee siitä, että testauspolitiikan ja testausstrategian välinen ero ei ole kovinkaan suuri – voihan testausstrategia sisältää osittain samoja asioita kuin testauspolitiikkakin.

Kooltaan haastattelun suurimmat OU:t, joilla on oma testausorganisaatio, ja joilla ei ilmennyt suurempia ongelmia testauksenstrategiassa ja hallinnassa, valitsivat testauksen toteutuksen tärkeimmäksi kehityskohteeksi. Tämä ei ole yllättävää, sillä näillä organisaatioyksiköillä ylemmän tason prosessit ovat jokseenkin kunnossa. Case 11 totesi seuraavaa:

> ”Kyl mä sanon, et se menis sinne ehkä alimpaan kuitenkin, et ne jotka sitä oikeesti tekee. Vaik sielt ylhäältä tuleekin ne niinkun et miten strategiat ja muut mutta, kyl se varmaan kuitenkin eniten sinne alimmalle tasolle, ketkä oikeesti sit tekee ne, ni vaikuttaa. ”

Myös case 7 painotti testauksen toteutuksen tasoa:

> ”Testausmenetelmiin ja ihan siihen käytännön testaamistyöhön niin kuin, pitäis luultavasti kouluttaa paljon enemmän. ”

Pienillä, tai nopeasti kasvaneilla organisaatioyksiköillä sen sijaan ei ole useinkaan riittävästi resursseja käytettävissä testaukseen, ja tästä syystä onkin ymmärrettävää, että he taas panostaisivat testauksen strategiaan ja hallintaan. Esimerkiksi case 6:n haastateltava sanoo:

> "Minä suuntaisin tutkimuksen ihan siihen, että miten periaatteellisesti, mihin kannattaa kiinnittää huomiota, sillä tavalla että olisi niillä ihmisillä, jotka tekevät päätöksiä resursseista, ihmisistä, resursoinneista, palkataanko testausosastoa ja muuta, niin olisi työkaluja niin kuin nähdä sitä, että millä tavalla se vaikuttaa siihen toimintaan. Tämä on minun näkemys ja, ja ne isommat ongelmat mitä tässäkin on tullut esille tämän keskustelun tai haastattelun aikana, liittyy nimenomaan siihen, että ne on näitä strategiaan. "

Case 1 toteaa seuraavaa:

> "Kyl mulla ainaki tulis mieleen se testausstrategia että, siihen että, miten niinku.. Nyt kun ei sit kuitenkaan oo resursseja testata kaikkee, niin mihin se sitte, mikä kannattaa sitte… "

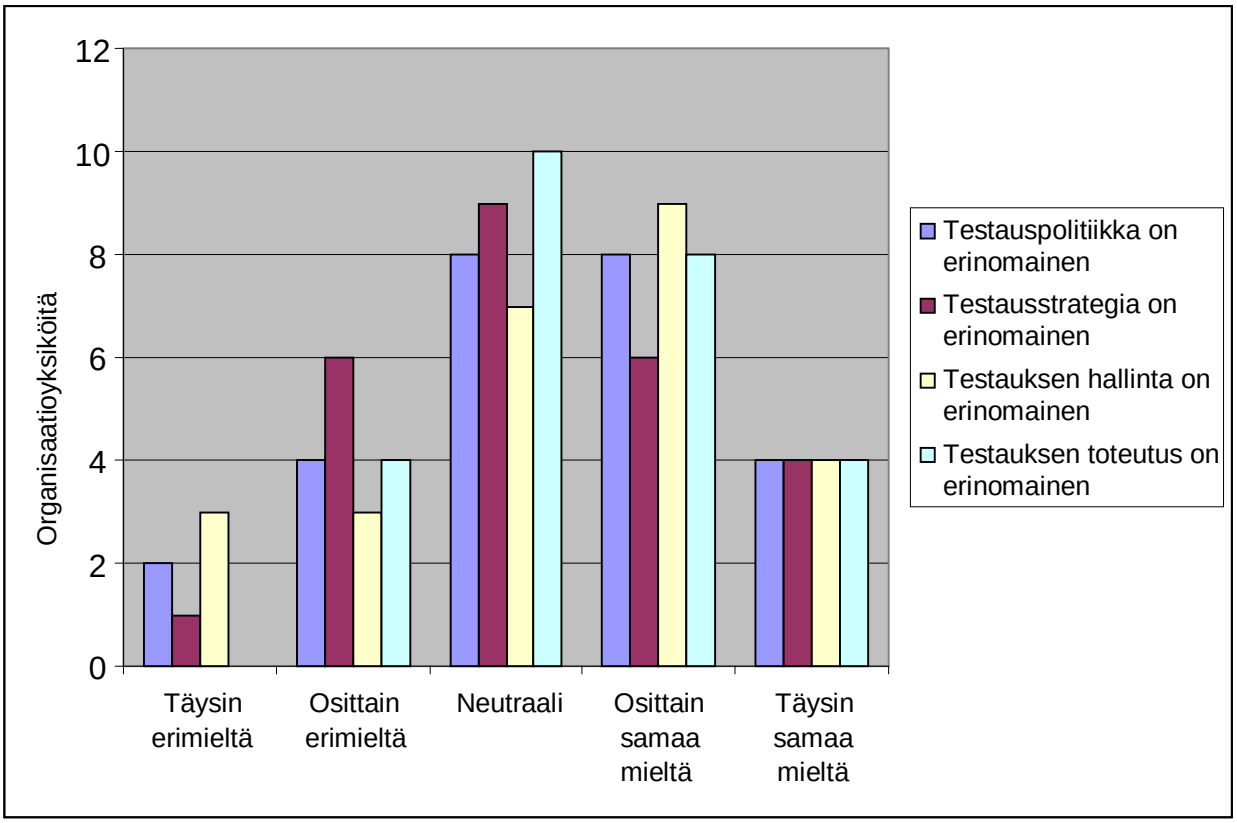
Case 9 korostaa testauksen hallinnan merkitystä:

> "Jos se on hyvin hallittua niin ehkä se mekaaninen työki siellä alareunassa sitte onnistuu paremmin. "

Myös Case 8 korostaa testauksen hallinnan merkitystä, jolloin testausta ei koettaisi ylimääräisenä työnä. Haastateltava totesi seuraavaa:

> "Sillä me päästään nimenomaan siihen, että sitä ei koeta taakaks sitä testausta ja ei aina koeta, että täytyy hirveesti nyt resurssoida testaukseen, vaan se testaus tulis niinku ikään ku siinä, siinä sivussa niinku ihan automaattisesti. Kun, kun me nyt jotenki tää tuote kuitenki on pitäny aina testata, kun se ensimmäinen toteutus on tehty, niin jos sen testauksen vois siinä tehdä kyljessä, niin sillon tämmöstä niinku kauheeta strategioita ja politiikkoja ei välttämättä tarvis niinkun ainakaan tällasel niinku lyhyellä, lyhyellä aikavälillä niinku miettiä niin paljo. "

Alla olevassa kaaviossa toisen haastattelukierroksen OU:t ovat arvioineet testausstandardin määrittelemien prosessien toteutumista heidän omassa toiminnassaan. Yli puolet organisaatioyksiköistä on joko neutraalilla kannalla tai eri mieltä testausprosessien erinomaisuudesta. Kuitenkin myös melko suuri osa organisaatioyksiköistä katsoo, että heidän testausprosessit ovat kunnossa. Muutama sanoo niiden olevan jopa erinomaisella tasolla. Testausstrategian erinomaisuudesta OU:t ovat eniten eri mieltä.



Taulukko 1: Organisaatioyksiköiden arvioinnit heidän omien testausprosessiensa erinomaisuudesta

8.1 Henkilöstö ja resurssit

Organisaatioyksiköiden käytettävissään olevat resurssit vaikuttavat merkittävästi OU:n mahdollisuuteen soveltaa standardeja. ISO/IEC 29119 standardi tukee OU:ta testauksen suunnittelemisessa, vaikka resursseja ei olisikaan käytössä tarpeeksi. Vähäisten testausresurssien priorisointi on ehdottoman tärkeää ja OU:n tulisi ehdottomasti luoda itselleen selkeä testauskäytäntö ja strategia, jotta vähäiset resurssit saataisiin kohdennettua oikeisiin kohteisiin.

Testausstandardin määrittelemät prosessit vaativat organisaatioyksiköltä henkilöstöä osallistumaan testaukseen. Standardi suosittelee, että OU:ssa olisi testauspäällikkö, joka osallistuu keskeisesti testausprosessien, esimerkiksi testausstrategian ja testauksen hallinnan toteuttamiseen. Standardin määrittelemän prosessimallin alin taso suosittelee erillisiä testaajia ja erillisiä vastuuhenkilöitä testauksen eri tasoille ja tyypeille. Lisäksi standardit suosittelevat, että OU:ssa olisi erillinen henkilö valvomassa, että luotua testauspolitiikkaa tai strategiaa noudatetaan ja niitä myös päivitetään tarpeen mukaan.

Henkilöstövaatimusten osalta testausstandardin soveltaminen laajassa mittakaavassa voi olla hankalaa pienille organisaatioyksiköille, joilla testaukseen erikoistuneita henkilöitä ei ole, ja joilla on pulaa myös muista testauksen resursseista. Tämänkaltaisissa organisaatioyksiköissä työskentelevillä henkilöillä on usein monia erilaisia työtehtäviä, kuten kehitys ja markkinointi, ja testausta tehdään vain jos siihen jää aikaa. 7 organisaatioyksikköä totesi, että heillä ei ole riittävästi resursseja testaukseen henkilöstön osalta. Nämä organisaatioyksiköt ovat pääsääntöisesti kooltaan huomattavasti pienempiä, kuin haastateltavista organisaatioyksiköistä ne, joilla testauksen henkilöstöä on riittävästi. Joukkoon mahtuu kuitenkin pari suurempaa OU:ta, joilla henkilöstöpula on aiheutunut nopean kasvun tai fuusioitumisen myötä. Alla on poimittuna esimerkkejä testausresurssien puutteesta:

> "Ku nä on, puristettu hyvin pieneks tää organisaatio, ei oo niinku varaa. Ja kyl mä tavallaan ymmärrän et ei, me ollaan tavallaan aika pienii, jos meitä on nyt joku, onks meit nyt viis kehittäjää täs suurin piirtein ja toimitusosastolla joku kolme neljä henkee. "Case 1

> "Um, no. We don't have enough resources. " –Case 3

> "Joo ei, me ollaan te-, me ollaan tehty niinku koko juttu ihan alusta, ja sitte testaaminenkin on jouduttu tekeen tällä, tällä niinku köyhän miehen versiolla. " –Case 4

> "Riippuen vähä, kyl siinä niinku on ehkä jonkinmoista kehitystä tullu, mutta tuota se on ollu vähä resurssi-, resurssipuutteenki vuoksi kyl semmonen valitettavasti, mistä… " –Case 5

> "No ainakaan niit ei oo liikaa. " –Case 9

Henkilöstön osalta 5 haastatelluista organisaatioyksiköistä totesi, että heillä ei ole testauksen resurssipulaa henkilöstön osalta. Case 7 totesi, että vaikka testauksen henkilöitä on tarpeeksi, niin henkilöt jakautuvat välillä epätasaisesti projektien kesken. Case 11 mainitsi testauksen resurssipulan testausympäristöjen osalta. Vaikka testaajia onkin tarpeeksi, joutuvat matalamman prioriteetin projektit joskus odottamaan vuoroaan testausympäristöjen puutteen vuoksi. Case 8 totesi, että heillä on resursseja testaukseen tarpeeksi, mutta toimintaa tulisi tehostaa:

> "No mun näkökulma on ehkä se, että, että sitä resursseja saattas jopa olla tarpeeks, mutta se toiminnan tehokkuus on, on niinkun ehkä se ensimmäisenä parannettava asia elikä, eikä pelkästään niin, että ihmisten toimintaa pitäs jotenkin tehostaa, et heidän pitäs klikkaillla hiirellä kaks kertaa enemmän, kun nykyään, vaan siis nimenomaan se, että työkaluja tehostettas ja tämmösiä testausta tukevaa toimintaa, niin se ois niinku yks tämmönen tarve, selkeesti. Niin sillon ei tarvita niin älyttömästi sitä porukkaa sinne ja toisaalta sen työn mielekkyys, sen testaajan työn mielekkyys, kun on kunnon välineet "

Nämä OU:t, joilla testauksen henkilöstöä on tarpeeksi, työllistävät kokonaisuudessaan satoja tai jopa tuhansia henkilöitä ja ovat siis haastatelluista organisaatioyksiköistä suurimpia. Pienillä organisaatioyksiköillä ei ole useinkaan varaa palkata erillisiä testaajia. Testaajien palkkaaminen ei kuitenkaan välttämättä ole ratkaisu testauksen ongelmiin pienissä organisaatioyksiköissä. Testauksen tehokkuutta voidaan lisätä ja kustannuksia alentaa panostamalla testauksen suunnitteluun.

8.2 Testauksen elinkaari

Testauspolitiikan (tai testausstrategian) kohta ”testauksen elinkaari” määrittää testausprosessin suhteen kehitysprosessiin. Perinteisen vesiputousmallin ongelmana on usein se, että kehitys vie testaukselta aikaa, koska testauksen ajoittuessa kehitysprosessin loppupuolelle kaikkea haluttua testausta ei ehditä tekemään projektin aikataulujen tullessa vastaan. Testauksen aikainen sisällyttäminen tuotekehitysprosessiin alentaa kustannuksia ja parantaa testauksen tehokkuutta [10]. Esimerkiksi Case 2 totesi, että heillä testaus ei ole riittävästi resursoitu, mutta henkilöiden lisäämisen sijasta heidän tulisi panostaa enemmän projektin alkuvaiheen testaukseen. Haastateltava toteaa, henkilöstön lisäys olisi ratkaisu ongelmaan, mutta liian kallis sellainen. Haastateltavan mielestä projektin alkuvaiheen testaukseen tulisi panostaa:

> ”se että mitä siinä nyt haluttais tehtävän et suoraan se että lisätään testaajia ja muuta niin tietysti aiheuttaa sen verran kustannuksia ainakin varmasti, mut ei välttämättä suoraan lisää sitä hyötyä että aika paljon toivoisin ehkä enemmänkin sitten vielä siihen, että enemmän pystyttäis panostamaan sinne ihan projektin alkuvaiheen testaukseen, määrityksien testaamiseen, suunnitelmien testaamiseen ja siis laadukkaampien määrityksien ja suunnitelmien tekemiseen ja niiden, niiden testaamiseen.. ”

8.2.1 Ohjelmistosuunnittelu- ja toteutusmenetelmän vaikutus testaukseen

Testauksen ja kehityksen väliseen suhteeseen vaikuttaa olennaisena tekijänä OU:n käyttämä ohjelmistojen suunnittelu- ja toteutusmenetelmä. Yhdeksän OU:ta ensimmäisen kierroksen haastatelluista ilmoitti käyttävänsä perinteistä menetelmä (vesiputous, tai sen kaltainen). Myös ketteriä menetelmiä käytettiin. Yksi OU (Case 3) sanoi käyttävänsä SCRUM -menetelmää, ja kaksi muuta (Case 6 ja Case 9) ilmoitti käyttävänsä ketterien menetelmien kaltaisia menetelmiä. Kaksi OU:ta (Case 7 ja Case 12), jotka käyttävät perinteistä menetelmää ilmoittivat, että ketterät

menetelmät ovat tekemässä tuloaan heidän organisaatioyksikössä. Esimerkiksi Case 5 ja Case 10 mainitsivat aikataulujen olevan suuri ongelma testauksessa. Molemmilla organisaatioyksiköillä on käytössään vesioutousmenetelmä.

> "Joskus on aika vaikee löytää, niinku projektiaikataulut on, meil on hirveesti töitä, niin mahotonta löytää enää aikaa sille testaukselle. "-Case 5

> "Ja just, ehkä se, että meillä, lähes, no nyt viime aikoina ainakin, niin näissä versioissa, vaikka meillä kuuluu version tähän, valmisteluun se, että aina on jonkun verran myös regressiotestausta, että käydään läpi olevaa toiminnallisuutta ja huolehditaan, ettei oo mikään menny rikki, niin tästä ollaan hyvin monesti jouduttu tinkimään ihan vaan sen takia, että aika loppuu kesken ja tää on oikeestaan semmonen asia, mihin pitäis jollain tavalla panostaa sitten. "-Case 10

Organisaatioyksiköiltä kysyttiin heidän arviotaan ketterien menetelmien vaikutuksesta testaukseen. Case 2 totesi, että ketterät menetelmät jakaisivat testausresursseja paremmin. Case 3 totesi, että uusi testaaja pääsee projektiin paremmin mukaan ketterää menetelmää käytettäessä. Case 6 oli sitä mieltä, että ketteriin menetelmiin liittyvän jatkuvan kommunikoinnin ansiosta virheet havaitaan aiemmin. Myös Case 11 ja Case 12 arvioivat, että ketterät menetelmien käyttö parantanee testattavuutta. Kaikki OU:t eivät vastanneet ollenkaan kysymykseen, jossa heitä pyydettiin arvioimaan ketterien menetelmien vaikutusta testaukseen. Näillä organisaatioyksiköillä oli käytössä enemmän perinteisiä suunnittelu- ja toteutusmenetelmiä.

Organisaatioyksiköiden, joilla testaus ajoittuu kehitysprosessin loppuun, tulisi laatia testausstrategia siten, että testaus ja kehitys olisivat paremmin integroituina toisiinsa, ja että testaus kytkeytyisi kehitysprosessiin jo varhaisessa vaiheessa. Esimerkiksi Case 12 arveli, että heillä ei ole testauksen resurssiongelmaa sen vuoksi, että testaus on integroitu tiiviisti tuotekehitykseen. Haastateltava toteaa, että heidän rakentamansa toimintamallin ansiosta testaukseen on löytynyt aina riittävästi henkilöitä. OU:ssa testausta ei nähdä erillisenä prosessina, vaan testaus on sulautettu kiinteäksi osaksi tuotekehitystä. Case 6 sanoi käyttävänsä nykyään enemmän ketterän menetelmän kaltaisia menetelmiä, mutta testaus ajoittuu edelleen projektin loppupäähän. Haastateltava ilmoitti tarpeesta testausprosessin elinkaaren muuttamiseen:

> "Mut mä haluaisin myös kehittää sitä testausprosessia, et miten se, miten se toimii. Nyt se on hirveen hankalaa kun ei oo dedikoituja testaajia ja nää ihmiset, joitten, jotka tekee sitä testausta, niillä on muita tehtäviä ja esimerkiks myyntityöt ja asiakaspalvelutyö kuormittaa näit henkilöitä niin paljon, et se on hyvin pieni osuus vuodesta joka on yleensä ennen pää-releasea, joka keskitytään siihen testaamiseen. Mä haluaisin mennä lähemmäs semmosta jatkuvaa, jatkuvaa iteratiivista, testataan, kehitetään,

> testataan, kehitetään. Nyt tehdään enemminkin sillä tavalla, että testataan tosi, kehitetään tosi paljon ja sit testataan intensiivisesti, korjataan intensiivisesti. Ja tää ei mun mielest sovi hyvin tämmöseen ketterämpään ajatusmalliin, vaan se ehkä on enemmän tämmönen vanha vesiputousmallin mukainen tapa toimia. ”

8.2.2 Organisaatioyksikön suhtautuminen testaukseen

Testauspolitiikka on OU:lle tärkeä siinä mielessä, että se määrittelee testauksen tavoitteet korkealla tasolla. Tämä vaikuttanee siihen, miten OU:n eri sidosryhmät suhtautuvat testaukseen. Esimerkiksi Case 4:n ongelmana oli saada testaukseen käytettäviä resursseja projektin rahoittajalta. Rahoittaja ei asettanut suurta painoarvoa testaukselle, ja tästä syystä OU ei voi toteuttaa testausta siinä laajuudessa kuin olisi tarve. Ääriesimerkkinä edellä mainitulle seikalle toimi Case 3, jonka haastateltava sanoi, että heillä testaukseen suhtaudutaan vakavammin, kuin mitä hankkeen kriittisyys edellyttäisi. Haastateltava sanoi, että tämänkaltaisen ajattelun tuloksena asiakkaalle toimitetussa ohjelmistossa on erittäin vähän virheitä. Standardi ehdottaa sidosryhmien osallistumista testauspolitiikan luontiin. On tärkeää saada sidosryhmät ymmärtämään testauksen tärkeys ja tavoitteet. Testauspolitiikka kehittänee ymmärrystä eri sidosryhmien välillä.

8.3 Testausstrategia ja sen kehittäminen

Seuraavissa kappaleissa käsitellään tarkemmin haastateltujen organisaatioyksiköiden testausstrategioita ja niiden ongelmakohtia sekä parannusehdotuksia, joita haastateltavat henkilöt ehdottivat testausstrategian kehittämiseksi. ISO/IEC 29119:n mukaan OU:lla tulisi olla testausstrategia, joka sisältää muun muassa päätökset siitä, mitä testataan, priorisoinnit testattaville asioille sekä testauksen elinkaaren suhteen kehityksen elinkaareen. Lisäksi standardi suosittelee, että testaukseen liittyvien päätöksien ja valintojen tulisi ohjautua myös projektikohtaisesti. Haastatellut OU:t ovat testausstrategian osalta hyvin erilaisessa asemassa. Kuitenkin jokainen OU mainitsi jonkun asian, jota heidän testausstrategiassa tulisi kehittää.

8.3.1 Case 12

Case 12:lla ei ole resurssipulaa testaajien henkilöstön osalta. OU:ssa testaus on integroitu osaksi tuotekehitystä. Haastateltava sanoo että mahdollisuudet testauksen kattavuuteen ovat hyvät, koska testaus on erikseen ohjeistettu ja se toteutetaan määritetyn testausmallin mukaisesti. OU:lla on myös

projektikohtainen testausvastaava. Tämä henkilö suunnittelee ja ohjaa testauksen sekä tekee suurimman osan testitapauksista yhdessä kehittäjien kanssa. Testausvastaavan osaaminen vaikuttaa merkittävästi testauksen onnistumiseen.

Tämän OU:n mielestä kriittisyys vaikuttaa testausstrategiaan siten, että hankkeen ollessa kriittisempi, asiakas osallistuu testaukseen jo järjestelmätestauksen vaiheessa. Myös regressiotestaus on tehokkaampaa hankkeen ollessa kriittisempi. Haastateltava haluaisi kehittää testauksessa automaatiota, jolla saataisiin yksikkö- ja regressiotestit automatisoitua.

Case 12 on varsin pitkällä testauksen suunnittelussa. Heillä on määritelty testauksen menettelytapa ja lisäksi testaus etenee rinnakkain tuotekehityksen kanssa. Ongelmana on kuitenkin, että testauksen onnistuminen on liikaa riippuvainen testausvastaavan osaamisesta. Jakamalla tätä vastuuta myös muille henkilöille esimerkiksi määrittelemällä testauksen tasoille ja tyypeille erilliset vastuuhenkilöt, testauksen onnistuminen luultavasti parantuisi. Testausstrategiassa OU:n tulisi panostaa kohtiin ”regressiotestaus ja uudelleentestaus” ja ”testauksen automaatio ja työkalut” sekä laatimalla suunnitelmat automaatiotyökalujen käytölle.

8.3.2 Case 11

OU:ssa on testauksesta vastaava henkilö, joka päättää, mitä testataan ja miten. Tämä henkilö laatii suunnitelman, joka käydään testaajien kanssa läpi. Testaajat valmistelevat testitapausdokumentit, joiden katselmointiin testausvastaava osallistuu. OU:ssa kiinnitetään huomiota siihen, mitä testauksessa tulisi huomioida projektikohtaisesti. Laadittu testaussuunnitelma käydään läpi palaverissa, ja lisäksi suunnitelma hyväksytetään myös asiakkailla. Asiakkaat pystyvät myös seuraamaan testauksen etenemistä. OU:lla on käytössään testauksen hallintatyökalu, joka helpottaa dokumenttien käsittelyä. Haastateltava kuitenkin kehittäisi edelleen tätä työkalua, jotta eri osapuolten välistä kommunikointia saataisiin tehostettua. Testauksen ongelmaksi haastateltava mainitsee testausympäristöjen tarpeen, koska nykytilanteessa toiset projektit joutuvat odottamaan.

Case 11 on haastatelluista eräs, jolla testauksen suunnittelu on varsin pitkällä. Standardin näkökulmasta erityisen edistyksellistä Case 11:n kohdalla on sidosryhmien osallistuminen testaussuunnitelman katselmointiin. Lisäksi myös projektikohtaista testauksen suunnittelua tehdään, mikä on myös testausstandardin ehdottama tapa toimia. Testausstandardin osalta tämän OU:n tulisi

panostaa projektin testausstrategian testausympäristöjen sekä tason testaussuunnitelman aikataulusuunnitteluun jotta vältyttäisiin mahdollisilta päällekkäisyyksiltä testausympäristöjen käytössä.

8.3.3 Case 10

Aiemmin OU:lla oli yksi testauspäällikkö ja kehittäjät hoitivat varsinaisen testauksen. Vaikka OU on hiljattain perustanut itsenäisen testausyksikön, kehittäjät osallistuvat myös integraatiotestaukseen siltä osin, kun heidän aikataulunsa tämän mahdollistaa.

OU:n testausstrategiassa kehittäjä tai projektipäällikkö valmistelee testitapaukset, jotka katselmoidaan palavereissa. Testausstrategia ei ole riskipohjainen, mutta projektikohtaisia riskejä arvioidaan. Testausprosessissa seurataan myös virheitä, ja niistä tehdään tilastointia, jonka avulla arvioidaan testauksen riittävyys. Tämä strategia on haastateltavan mielestä toiminut kohtuullisen hyvin, mutta haastateltava kaipaisi enemmän ohjeistusta testausstrategian toteuttamiseen, jotta tiedettäisiin, onko testaus tarpeeksi kattavaa ja tuleeko tarvittavat asiat testattua riittävällä tasolla. Haastateltavan mielestä myös testauksen automatisointiin tulisi panostaa, koska tällä hetkellä OU:lla ei ole minkäänlaista testausautomaatiota käytössä.

Case 10:n tulisi määritellä paremmin testauksen laajuus, käytettävät testauksen tyypit, tekniikat sekä tasot sekä testauksen päättämisen kriteerit. Myös testien valintaan ja priorisointiin sekä testauksen automaatioon ja työkaluihin tulisi panostaa. Itsenäisen testausorganisaation myötä testaajien vastuut tulee määritellä tarkasti. OU:n tulisi selvästi panostaa testausstrategiansa tarkempaan määrittelemiseen, sekä myös toiminnan ohjaamiseen enemmän testausstrategian mukaiseksi.

8.3.4 Case 9

Case 9:n kohdalla kehittäjät varmistavat moduulien toimivuuden, jonka jälkeen erilliset testaajat hoitavat järjestelmätestauksen. Lisäksi ulkopuolinen ryhmä vastaa käyttöliittymätestauksesta ja seuraa testien tulosten kehittymistä. OU on yrittänyt kehittää testausstrategiaa löytämällä parempaa testauksen toimintamallia. Haastateltavan mielestä ongelmana on testauksen ajoittuminen

kehitysprosessin loppuvaiheelle, joka on kriittistä aikaa, ja täysin riippuvainen muutamasta testauksen avainhenkilöstä. Näiden henkilöiden ollessa poissa testaus voi jäädä vajaaksi.

OU:lla on käytössä kohtuullisen tehokas virheiden raportointijärjestelmä. Testausstrategiaa tulisi kuitenkin kehittää myös virheiden luokittelun osalta. Nykyään käytössä oleva luokittelu on haastateltavan mielestä liian abstrakti. Haastateltava haluaisi kehittää myös testausautomaatiota regressiotestauksen osalta, mutta arvelee kustannusten olevan ongelmana. Testauksen hallintapuoli on sen sijaan kohtalaisen hyvin automatisoitu.

Case 9:n tulisi kehittää testauksen elinkaarimallia, jotta testaus ja kehitys eivät olisi niin erillään toisistaan. Virheiden hallintaprosessia tulisi kehittää ja automatisointi tulisi käsitellä testausstrategian näkökulmasta.

8.3.5 Case 8

Case 8:n kohdalla tuotteen kriittisyys ja sovellusalue vaikuttavat päätökseen, mitä testejä tehdään. OU on tuotesuuntautunut - tuotteet tehdään aina saman alustan päälle. Tästä syystä testausresurssit keskittyvät tuoteytimeen. OU:n testausstrategiaan kuuluu, että ohjelmasta on olemassa aina julkaisukelpoinen versio, jonka testaukseen panostetaan mahdollisimman paljon. Aikaisemmin testausstrategia määräytyi enemmän projektikohtaisesti. Haastateltava kokee ongelmakohdaksi regressiotestauksen, jossa strategia ja käytäntö eivät aina kohtaa. Vaikka haastateltavan mielestä OU:lla on riittävästi testauksen resursseja käytössä, testausta tukevaa toimintaa tulisi tehostaa ja testaustyön mielekkyyttä tulisi kehittää kunnollisilla testaustyökaluilla. Lisäksi regressiotestausta ja automaatiota tulisi kehittää.

Testausstandardin osalta OU:n tulisi huomioida ”testauksen automaatio ja työkalut” –kohdan kehittäminen testausstrategiassaan. Myös regressiotestauksen projektikohtaista strategiaa tulisi kehittää. Lisäksi OU voisi hyödyntää testausstandardia myös muilta osin nykyisen testausmenetelmänsä tukemiseen.

8.3.6 Case 7

Case 7 on haastatelluista organisaatioyksiköistä testausstrategian kannalta pisimmälle kehittynyt. OU:lla on nykyään erillinen testausyksikkö. Hankkeen kriittisyys ohjaa testauksen suunnittelua ja aikataulutusta. Hankkeen yhteydessä tehdään riskikartoituksen, jossa on arvioituna projektin riskien vakavuudet ja todennäköisyydet. OU:n testausstrategiassa testaus huomioidaan alusta lähtien. Testaajat ovat mukana jo vaatimusmäärittelyn katselmoinnissa ja myös vaatimusten testattavuutta arvioidaan. Testaajat tekevät testaussuunnitelman, jossa päätetään, mitä testataan. Nämä päätökset katselmoidaan, ja päätöksiin pääsevät vaikuttamaan myös muut projektin osapuolet. Testaajat osallistuvat projektiryhmien palavereihin.

Haastateltavan mielestä testauksen toteutuksessa ei ole ongelmia vaan testauksen suunnitteluun pitäisi panostaa siten, että testaajat ovat mukana mahdollisimman aikaisessa vaiheessa. Haastateltava ilmoittaa myös tarpeen riittävälle kommunikoinnille projektin aikana. Projekti tarvitsisi suunnittelijoista ja testaajista koostuvan yhteyshenkilöjoukon, joka kokoontuisi säännöllisin välein projektin aikana. Tällä hetkellä tällainen tapaaminen järjestetään vain hätätilanteessa.

Case 7 ei välttämättä tarvitse testausstandardin soveltamista toimintaansa, koska heillä on käytössään talon sisäinen, erittäin pitkälle kehittynyt testausprosessi.

8.3.7 Case 6

Case 6:n kohdalla hankkeen kriittisyys vaikuttaa suoraan siihen, mitä osia painotetaan ja mihin resursseja kohdennetaan testauksessa. Testausstrategia määrittelee kriittisille osille regressiotestit. Resurssien puutteen ja rajallisuuden vuoksi vähemmän kriittiset osat jäävät vähemmälle testaukselle. OU:lla ei ole omaa testausorganisaatiota, vaan regressiotestauksen hoitaa asiakkaaseen yhteydessä oleva tilaaja. OU:n laatupäällikkö organisoi tilaajan tekemää testausta ja kehitysorganisaatio korjaa näissä testeissä havaitut virheet. Haastateltavan mielestä testaus on kohtuullisen kattavaa. Testaus keskittyy uusien ominaisuuksien varmentamiseen ja tästä syystä kattavuus ei välttämättä aina kuitenkaan ole riittävä.

Testauksen automaatiota tulisi haastateltavan mielestä lisätä. Lisäksi haastateltava mainitsee regressiotestaukseen liittyviä ongelmia, kuten regressiotestien kirjoittamisen hankaluuden ja regressiotestauksen kattavuuden riittämättömyyden. Lisäksi vastuu siitä, kuka hoitaa regressiotestauksen, on huonosti määritelty. Myös yksikkötestauksen kattavuudessa on parannettavaa. Haastateltava haluaisi kehittää testausprosessia. Nykyisessä toimintamallissa testaukseen keskitytään hieman ennen ohjelman pääversion julkaisua ja haastateltava haluaisikin testauksesta ja kehityksestä vuorottelevan ja iteratiivisen prosessin.

Case 6:n kohdalla testaus ei ole haastateltavan mielestä riittävästi resursoitu. OU:lla ei ole henkilöitä, joiden päätehtävänä on testaus. Myyntityö ja asiakaspalvelu kuormittavat henkilöitä niin paljon, että heillä ei jää aikaa testaukseen. Tästä aiheutuu, että testaus tehdään kiireessä hieman ennen ohjelman pääversion julkaisua. Ratkaisuna ongelmaan olisi palkata lisää henkilöstöä, joka OU:n nykyisessä tilanteessa olisi myös haastateltavan mukaan mahdollista. Haastateltava arveleekin nyt olevan oikea hetki itsenäisen testausosaston perustamiselle.

Case 6 hyötyisi testausstandardista monelta osin. Testaus- ja kehitysprosessin välinen suhde tulisi määritellä uudelleen testausstrategiaan. Henkilöiden vastuut tulisi määrittää projektin testaussuunnitelmassa, jotta vastuut olisivat selkeät. Lisäksi testausstrategiaa tulisi kehittää kattavuuden arvioinnin helpottamisen osalta. Mikäli OU perustaa lähitulevaisuudessa itsenäisen testausosaston, testausstandardia tulisi hyödyntää määrittelemään selkeät prosessit, jotta kommunikointi kehitys- ja testausosaton välillä sujuu ongelmitta.

8.3.8 Case 5

Case 5:llä ei ole erillisiä testaajia resurssien puutteen vuoksi. Testaukselle vaikea löytää aikaa testaukselle projektin kiireellisten aikataulujen vuoksi. Lisäksi ei ole ketään ulkopuolista tahoa, joka varsinaisesti vaatisi testausta. Case 5:n haastateltava sanoo suoraan, ettei heidän testaussuunnitelma ole aina toiminut ja sitä tulisi kehittää. Testauksen suunnitteluun tulisi panostaa jo määrittelyvaiheessa. Nykymenetelmällä esimerkiksi ohjelmistorajapintoihin liittyviä seikkoja saattaa jäädä testaamatta. Osittain ongelmiin on tullut ratkaisuja, mutta testauksen resurssipula on vaikeuttanut asioita.

OU on rakentanut viimevuosina toimintakulttuuria, jonka tavoitteena on kehittää testauksen suunnittelua, itse testausta ja sen dokumentointia. Erityisesti dokumentointi on ollut ongelmana, välillä se puuttuu kokonaan. Automaatiotestausta tehdään, mutta tuloksia ei ole dokumentoitu. Tässä haastateltava näkee selkeän kehityskohteen. Vaikka OU käyttää toimintansa tukena laatujärjestelmää, tehty testaus pitäisi todistaa. Haastateltava sanookin seuraavaa:

"Meille sellainen testaus, mistä ei löydy mitään dokumenttia, on hukkaan heitettyä. "

Case 5:n tulisi kehittää testausstrategiaansa kokonaisuudessaan. Testauksen ei tulisi olla liian erillään kehityksestä. OU:ssa pitäisi olla henkilö, jonka päätehtävänään olisi vaatia ja hallita testausta. Dokumentointikäytäntö vaatisi yhtenäistämistä. Testauksen laajuus tulisi määritellä tarkemmin, jotta kattavuus pystyttäisiin varmentamaan paremmin.

8.3.9 Case 4

Case 4:n testausstrategiassa testausprojekti sisältyy kehitysprojektiin. Testausstrategia sisältää päätökset siitä, mitä testataan ja mitkä testauksen kohteet ovat kriittisiä. Kriittisten osien testaus viedään läpi järjestelmällisellä projektinhallinnalla. OU on kehittänyt testausdokumentaatiota ja kasvattanut ulkopuolista testausta. Käyttäjät ja asiakkaat osallistuvat testaukseen. Resurssien puutteen vuoksi OU:lla ei ole varaa toteuttaa testausta siinä laajuudessa kun olisi tarve. OU kaipaisi lisää testausympäristöjä käyttöönsä, jotta turvallisuus- ja käytettävyystestausta voitaisiin tehdä enemmän.

Case 4:n suurin ongelma testauksessa on resurssien puute. Haastattelussa ilmeni, että resurssien puute sanelee hyvin pitkälti testauksen toteuttamistavat.

8.3.10 Case 3

Case 3:n haastateltava toteaa, että heillä testaus on melko kattavaa. OU etsii yhä tehokkaampia tapoja testauksen suorittamiseen. OU:n testauspolitiikassa pyritään kaikenkattavaan testaukseen, jonka vuoksi varsinaista menetelmää testitapauksien päättämiseen ei ole. OU:n testauspolitiikan ansiosta hankkeeseen suhtaudutaan vakavammin kuin mitä hankkeen kriittisyys todellisuudessa on. Tästä aiheutuu, että lopputuotteeseen päätyy hyvin pieni määrä virheitä. Testaus on myös pitkälti automatisoitua. OU:lla on päätoimisia testaajia ja lisäksi he ovat hankkineet ulkoa testauksen

konsultteja, mutta testaushenkilöstöä ei tällä hetkellä kuitenkaan ole tarpeeksi, koska OU on hiljattain fuusioitunut.

Parannusehdotuksina haastateltava mainitsee testauksen dokumentoinnin, jota parantamalla tiedon siirto uusien henkilöiden välillä sujuisi paremmin. Haastateltava parantaisi myös testauksen raportointia, koska informaatio manuaalisista testeistä on testaajien päässä. Ongelmia aiheutuu silloin, testaaminen täytyy siirtää jollekin toiselle testaajalle. Tämä siirto vie tarpeettoman kauan aikaa ilman asianmukaista dokumentointia.

Case 3 voisi hyödyntää testausstandardin dokumentaatio- ja kommunikointimalleja, koska OU:n suurimmat ongelmat testauksessa liittyvät tiedonkulkuun ja dokumentointiin. Koska kaikenkattava testaus on mahdotonta, testitapausten päättämiseen tarvittavan menetelmän puuttuminen voi aiheuttaa ongelmia, mikäli OU saa toteutettavakseen erittäin laajan ja monimutkaisen ohjelmistoprojektin.

8.3.11 Case 2

Case 2:lla testauksessa ei ole käytössä mitään vakiintuneita tai tarkkoja ohjeistuksia testauksen suunnittelussa. Testausstrategiana on siis enemmän tai vähemmän ad-hoc -menetelmät. Kriittisiä toimintoja on testattu koodikatselmointien avulla. Erillistä testaushenkilöstöä ei ole vaan testaus on yhdistetty tuotekehityshenkilöstön tehtäväksi. Vastuu testauksesta on projektipäälliköillä ja asiakaspuolen henkilöllä. Kehittäjät tekevät yksikkötestausta normaalin työn ohessa ja lopullinen testaus tehdään projektipäälliköiden ja asiakkaiden toimesta testausympäristössä. Aikaisemmin tämä testausstrategia on jokseenkin toiminut, kun asiakkaita on ollut samalla tuotteella vain muutama. Nykyään samalla tuotteella on useita asiakkaita, ja haastateltava toteaa, että testausta täytyy tehostaa, koska ohjelmistovirheet tulevat yhä kalliimmiksi.

Haastateltava haluaisi lisää koodin läpikäyntiä ja katselmointia sekä parempaa dokumentointia, jotta piilossa oleva tiedon määrä vähentyisi, ja uudet testaajat pääsisivät helpommin sisälle testaustyöhön. Myös hyväksymistestaus tulisi suorittaa nykyistä järjestelmällisemmällä tavalla. Haastateltava toteaa, että henkilöstön lisäämiseen OU:lla ei ole varaa, ja että OU:n tulisi panostaa enemmän projektin alkuvaiheen testaamiseen määrityksien ja suunnitelmien osalta.

Case 2 on haastatelluista organisaatioyksiköistä sellainen, jolle testausstandardista olisi suurta hyötyä, koska varsinaista testausstrategiaa ei ole. Case 2:n tulisi luoda testauspolitiikka ja testausstrategia ohjaamaan testaustoimintaa.

8.3.12 Case 1

Case 1:llä ei ole testausstrategiaa eikä erillisiä testaajia. OU:lla ei ole varaa palkata erillisiä testaajia ja testaus on työtehtävä, joka ei varsinaisesti kuulu kenenkään henkilön toimenkuvaan. Kehittäjät itse testaavat niin kuin parhaaksi näkevät ad-hoc menetelmää käyttäen. Testaus ei ole kovinkaan laajamittaista eikä kattavaa. Uudet ominaisuudet pyritään kuitenkin varmentamaan, mutta muutosten vaikutusta ei testata mitenkään – OU:lla ei ole siis lainkaan regressiotestausta. Case 1 tarvitsisi paitsi lisää resursseja testaukseen, myös tietämystä siitä, mitä pitää testata. Lisäksi OU tarvitsisi myös työkaluja regressiotestaukseen.

Case 1:n tulisi soveltaa testausstandardia toiminnassaan monelta osin. OU:n tulisi luoda testausstandardia apuna käyttäen testauspolitiikka testausstrategia, jotta ad-hoc päätöksenteko vähentyisi.

8.4 Kriittisyyden vaikutus testausstrategiaan

ISO/IEC 29119:n määrittelemä testauksen hallintaprosessi pitää sisällään projektikohtaisen testaussuunnitelman luomisen. Prosessissa tulee arvioida projektikohtaiset riskit, sekä keinot, joilla näitä riskejä lievennetään. Organisaatioyksiköiltä kysyttiin, muuttuisiko heidän testausstrategiansa, mikäli hanke olisi kriittisempi. Kahdeksan OU:ta totesi kriittisyydellä olevan selkeä vaikutus siihen, miten testauksen resursseja kohdennetaan sekä siihen, mitä testataan. Näistä organisaatioyksiköistä eräs, jolla ei erillistä testausstrategiaa ollut, totesi että kriittisyydellä on vaikutus testauksen määrään. Mikäli he toteuttaisivat kriittisempiä ohjelmistoja, tulisi heidän saada erillisiä testaajia käyttöönsä. Resurssien ollessa rajalliset, kriittisiä osia painotettaessa vähemmän kriittiset osat voivat jäädä vähemmälle testaukselle. Case 6 totesi seuraavaa:

> ”Kyllä se vaikuttaa ihan suoraan siihen, et mihin panostetaan. Ja se vaikuttaa myös siihen, että, et sitten ei enää jää aikaa ns. vähemmän kriittisen, sanotaan käyttöliittymätestaus on sitten luonteeltaan hyvinkin manuaalista, jota on vaikee automatisoida, niin se ehkä sitten täntyyppiset asiat usein kärsii. ”

8.5 Testauksen ulkoistaminen

Testauspolitiikassa ja –strategiassa kohdat ”itsenäisyyden aste” määrittää, miltä osin testaus hoidetaan OU:n sisällä ja kuinka suuri osa testauksesta toteutetaan alihankintana. Testauspolitiikassa ja –strategiassa määritelty kohta ”testaajan osaaminen” tulisi myös määritellä uusien henkilöiden palkkaamisen kannalta riittävän kattavasti. Mikäli OU:lla ei ole omia testaajia, voisi henkilöstöongelman ratkaista siirtämällä testaus ulkopuolisen toimijan vastuulle. Testauksen ulkoistamisen ongelmana on testaajalta vaadittava tiedon, osaamisen ja kokemuksen tarve. Lisäksi testauksen toteuttaminen alihankkijoilla on kallista. Case 4 totesi ulkoistamisen olevan mahdollista heidän kohdallaan, mikäli heillä olisi varaa siihen. Haastateltava arvioi, että mikäli testauksen ulkoistamisen olisi ollut mahdollista, erään projektin valmistuminen olisi nopeutunut kahdella vuodella.

8.5.1 Testaajalta vaadittava osaaminen

OU:n liiketoimintasuuntautuneisuudesta riippuen testaajahenkilölle kohdistuu erilaisia vaatimuksia. Liiketoimintasuuntautuneisuus vaikuttaa OU:n tietämyksen hallinnan strategiaan. Testaajalta vaaditaan usein teknisen osaamisen lisäksi myös toimialaosaamista, ja näiden osaamisten painottuminen riippuu siitä, onko OU tuote- vai palvelusuuntautunut [10]. Toimialaosaamista ei ulkopuolisella testaajalla välttämättä ole, ja sitä voi useimmiten kartuttaa vain kokemuksen avulla. Joissakin tapauksissa testaajilta vaadittiin sekä teknistä osaamista että toimialaosaamista.

Case 12:n kohdalla testaus vaatii liiketoimintatuntemuksen lisäksi tietoutta ympäröivistä järjestelmistä. OU ilmoitti, ettei se ei hanki testauspalveluja ulkoa. Case 3:n testaajilta vaaditaan teknistä osaamista. Haastateltava sanoo toimialaosaamisen tulevan ajan kanssa. Toimialaosaamisen tarvetta kuvaa kuitenkin erittäin hyvin se, että tässä OU:ssa paljon toimialatuntemusta omaavat henkilöt osallistuvat joskus testaukseen, vaikka eivät ole varsinaisia testaajia. OU hankkii ulkoa testauksen konsultteja, koska omia resursseja ei ole riittävästi. Case 11:n testaushenkilö tarvitsisi hyvin laajaa toimialaosaamista, mutta myös järjestelmätietoutta. Tämä tieto on pääasiallisesti talon sisäistä tietoa. OU:lla on käytössä perehdyttämisohjelma uusia työntekijöitä varten. Myös Case 7:n testaajalta vaaditaan hyvin laajaa toimialatuntemusta ja myös tällä OU:lla on käytössään laaja

perehdyttämisohjelma uusia henkilöitä varten. Case 7 hankkii ulkoisia testaajia. Case 10:n testaajalta puolestaan vaaditaan sekä sovellusalueosaamista että teknistä osaamista.

8.5.2 Dokumentointi edellytyksenä ulkoistamiselle

Haastatteluissa ilmeni, että usein testaajilta vaaditaan myös paljon ”hiljaista tietoa”. Tällä tarkoitetaan sellaista tietoa, jota ei ole dokumentoitu. Mikäli OU:lla ei ole riittävää testauksen dokumentointia, on tiedon siirtäminen henkilöiden välillä ongelmallista ja myös uusien testaajien perehdyttäminen voi vaikeutua. Eräs OU koki ongelmaksi testaajien siirtymisen eri projektien välillä, kun valtaosa testaukseen tarvittavasta tiedosta on testaajien päässä.

Case 9:n testaushenkilöstöltä vaaditaan erittäin laajaa toimialantuntemusta ja uuden testaajan kouluttaminen on tästä syystä haastavaa. Haastateltava kuitenkin sanoo, että projektin riittävä määrittely ja dokumentointi vähentää tarvetta tuntea toimialaa. OU on hankkinut ulkoisia henkilöitä testaamiseen. Case 5:n kohdalla testaajalta vaaditaan sekä teknistä osaamista, että toimialatuntemusta. Testaukseen on haastateltavan mielestä vaikea löytää sopivaa henkilöä, koska dokumentoidun tiedon lisäksi testaaja tarvitsee myös hiljaista tietoa. Case 8 toteaa, että heidän tapauksessa testausta ei voi toteuttaa alihankkijoilla, koska testaus vaatii vahvaa tuotteen ja markkinoiden tuntemusta. Ulkoistaminen olisi joissakin tapauksissa mahdollista, mutta edellyttäisi tällöin parempaa dokumentointikäytäntöä OU:n sisäisestä toiminnasta.

Myös Case 2 toteaa, että heidän kohdallaan testauksen toteuttaminen alihankintana ei onnistuisi suoraan, koska dokumentointi ei ole kaikissa projekteissa riittävällä tasolla. Testaussuunnitelmaa ja dokumentointia tulisi haastateltavan mielestä kehittää. Case 4 totesi niin ikään, että testauspalveluiden hankkiminen ulkoa edellyttää kattavaa dokumentointikäytäntöä. Case 1:n kohdalla ulkoiselta toimijalta vaaditaan teknistä osaamista, ja OU on pyrkinyt dokumentoimaan tehtävät mahdollisimman tarkasti. Testauspalveluita ei kuitenkaan ole hankittu ulkoa resurssien puutteen vuoksi.

Testauksen ulkoistamisen edellytyksenä on siis kattava dokumentointikäytäntöä. Organisaatioyksiköt, jotka toteuttivat jossakin määrin testauksen ulkoistamista, esimerkiksi Case 10 ja Case 11, totesivat, ettei ulkoistamisella ole vaikutusta testausstrategiaan. ISO/IEC 29119 -standardin määrittelemät prosessit ja niiden dokumentointi vähentänee hiljaisen tiedon määrää, ja

siten myös helpottanee uusien henkilöiden palkkaamista ja mahdollista testauksen toteuttamista alihankkijoilla.

8.6 Testausautomaatio ja työkalut

Aikaisemmissa tutkimuksissa testausautomaation katsottiin olevan merkittävä tekijä, jolla testauksesta aiheutuvia kustannuksia alennetaan. Automatisoinnilla voidaan myös lisätä testauksen tehokkuutta [1]. Automaation toteuttamisen kannattavuuteen vaikuttavat kuitenkin monet tekijät. Yleisten ja itsenäisten tuotteiden kohdalla automatisointi onnistuu helpommin kuin erilaistettujen ja monimutkaisten tuotteiden kohdalla. Lisäksi, mikäli tuotteet ovat samankaltaisia, ja ne rakentuvat samankaltaisen tuoteytimen päälle, testauksen automatisointi on helpompi toteuttaa. Myös testitapausten uudelleenkäytettävyys lisää automatisoitavuutta [10].

Automaatio ei kuitenkaan sovellu kaikkiin prosesseihin. Koska automaatioympäristön toteuttaminen on kallista, kertakäyttöisen testausautomaation toteuttaminen ei ole kannattavaa. Haastatteluista havaittiin, että suurin automatisoinnin toteuttamisen ongelmista on siitä aiheutuvat kustannukset. Kun organisaatioyksiköiltä kysyttiin testausautomaation käytöstä, he vastasivat seuraavaa:

> "Niin siis, ei käytetä. Ja tota niin, siksi kun ei ole, no en tiedä. Siis mun näkemys on se, että siitä ois hyötyä, jos vaan ois resursseja, mutta sitten toisaalta taas, niin tota, en tiedä yhtään, että mitä se tulis maksamaan tavallaan, että sais tän automatisoinnin pyörimään, että voi olla, että se ei ois kuitenkaan niinkun kannattavaa. " –Case 10

> "..Niin et se vie resursseja hirveesti aina sitte. " –Case 1

> V: Niin, semmosen automaattitestausjärjestelmän kehittäminen on melekeen yhtä iso työ ku sen varsinaisen projektin kehittäminen.. " –Case 9

> "Se vois tiettyihin ohjelmiston osiin sopiakin, mutta tota niin, just se, että sen tavallaan käyntiin laittaminen ja kaikki työ, mitä siihen vaatii, että me saadaan tämmönen toimiva, automatisoitu ympäristö niinkun pyörimään, niin tota, uskoisin, että siihen ei olla välttämättä halukkaita panostamaan sen takia, et se voi olla kallistakin. "- Case 10

> "Toi automaatio on semmonen sana, joka pyörii vähän väliä meidänkin firman niinku, palavereissa, että tota niin, ja mä tiedän, että meidän ohjelmistossa on sellasia osia, joiden testaustausta pystyttäs automatisoimaan, kun siihen vaan löytyis resursseja, että tota, ja sekin on tiedossa, että kun tää

> automatisoi-, tai automaattiset testitapa saatais pyörimään, niin siinä taas sit toisaalta tulis jatkossa sit säästöä. Tällä hetkellä meidän osastolla ei, ei oo minkäännäköstä testauksen automatisointia käynnissä. ”
> -Case 11

8.6.1 Testausautomatisoinnin käyttökohteet

Standardin mukaan organisaatioyksikön tulisi testausautomaatiota suunniteltaessa määrittää testausstrategian kohta ”automaatio ja työkalut”. Standardi ehdottaa myös, että organisaatioyksikön tulisi projektikohtaisesti määritellä, mitä automaation lähestymistapaa käytetään ja mitä työkaluja testauksessa käytetään. Myös testitapausten uudelleenkäytettävyys tulisi arvioida. Haastatelluista organisaatioyksiköistä 9 mainitsi tarpeen testausautomaation lisäämiselle. Näiden organisaatioyksiköiden tulisikin kiinnittää huomiota automatisoinnin ja työkalujen käytön suunnitteluun ja integroida suunnitelma kiinteäksi osaksi heidän testausstrategiaansa. Seuraavissa kappaleissa esitellään haastateltujen organisaatioyksiköiden kokemuksia ja tarpeita testausautomaation ja työkalujen osalta.

Case 12:lla on positiivisia kokemuksia automaation ja työkalujen käytöstä. Kuormitustestausta ja yksikkötestausta on automatisoitu. Haastateltava kaipaisi automaation lisäämistä myös regressiotesteihin. Työkalut on olemassa, mutta aikaa ei ole ollut tämän kehittämiseen.

Case 11 mainitsi, että heillä on ollut testausautomaation työkaluja käytössä, mutta ei ole enää. Käytetty työkalu oli liian raskas, ja sen katsottiin jopa haittaavan testausta. Haastateltava ei kuitenkaan syytä tästä automaatiota vaan työkalua, ja toivoo löytävänsä sopivamman työkalun automatisoinnin toteuttamiseen.

Case 10:llä ei ole testausautomaatiota käytössään. Haastateltava sanoo, että siitä olisi varmasti hyötyä, mutta toisaalta siitä koituvia kustannuksia on vaikea arvioida. Kertakäyttöisen automaatioympäristön rakentaminen arveluttaa, koska uudelleenkäytettävyydestä ja ylläpidettävyydestä ei ole takeita. Tämä OU voisi arvioida testitapausten uudelleenkäytettävyyttä ja laatia testausstrategian automatisoinnille.

Case 9:llä on käytössään työkaluja testauksen hallinnassa mutta regressiotestauksen osalta, OU on vielä alkutekijöissään. OU on käyttänyt työkaluja suorituskyvyn ja muistinhallinnan testaamiseen.

Case 8: on automatisoinut yksikkötestausta ja käyttöliittymätestaus on automatisoitu osittain.

Case 6:lla on käytössään työkaluja testauksen automatisointiin. Regressio ja yksikkötestauksessa automaatiota on käytössä vähän.

Case 5 on käyttänyt automaatiotyökaluja kuormittavuustestaukseen. Myös toiminnallisuutta testataan automaation avulla. Case 5:n ongelmana on automaatiotestauksen dokumentoinnin puuttuminen. OU kaipaisi työkaluja kuormittavuus ja integraatiotestien testitapausten tekemiseen.

Case 4 on hankkinut käyttöönsä joitakin automaatiotyökaluja, mutta pääoman puute rajoittaa hankintaa. OU tarvitsisi työkaluja käyttöliittymätestaukseen.

Case 3 käyttää testausautomaatiota ja sitä ollaan lisäämässä. Yksikkötestit, suorituskykytestaus ja testaus muistivuodoilta on automatisoitu. Huolimatta siitä, että OU on enemmän palvelusuuntautunut, regressiotestauksen käyttö on mahdollista. Haastateltava toteaa automaatiolla olevan kuitenkin omat ongelmansa. Automaatio ei ole hyödyllistä, mikäli testitapaukset eivät ole uudelleenkäytettäviä.

Case 2:lla ei testausautomaatiota käytössä, mutta suunnitteilla kuitenkin kehittää regressiotestausta.

Case 1 toteaa, että testausautomaatiota olisi hyvä lisätä, mutta se on kallista. Tällä hetkellä testausautomaation käyttö ei kuitenkaan ole kovinkaan kattavaa. OU:lla tuoteydin pysyy kohtuullisen samana jolloin automaation toteuttaminen olisi myös mahdollista. OU on hankkinut testausohjelmiston, jolla olisi tarkoitus lisätä automaatiota. Lisäksi OU on hankkinut työkaluja muistinhallinnan testaamiseen, mutta kaipaisivat parempia työkaluja tähän tarkoitukseen ja lisäksi kuormitustestaukseen ja käyttöliittymätestaukseen.

8.6.2 Testauksen hallinta ja virheiden raportointi

Testaukseen liittyvät toimenpiteet, kuten virheiden hallinta ja raportointi ovat keskeisessä osassa ISO/IEC 29119 -standardin määrittelemän testauksen hallinnan ja testauksen toteutuksen prosesseissa. Testauksen hallintaan olisi syytä käyttää jotakin työkalua, jotta eri osapuolet pystyisivät kommunikoimaan esteettömästi projektin aikana ja turhaa aikaa ei kuluisi triviaalien tehtävien suorittamiseen. Monella OU:lla on käytössään jokin työkalu testauksen hallintaa varten, mutta joiltakin työkalut puuttuvat kokonaan.

Case 12 käyttää on käytössään oma ohjelmansa testauksen hallintaan. Tämä työkalu säästää merkittävästi aikaa ja tukee haastateltavan mukaan testausta todella hyvin. Myös Case 11:lla on käytössä testauksen hallintatyökalu, jonka haastateltava haluaisi yhä laajempaan käyttöön. Case 6:lla testauksen virheraportointi on osittain automatisoitu ja lisäksi virheraportoinnin prosessi on erikseen määritelty. Case 9:llä on käytössään järjestelmä, jonka avulla ohjataan ohjelmointityötä ja testausta. Järjestelmään kirjataan uudet ominaisuudet, jotka testataan ja järjestelmää käytetään myös virheiden jäljittämiseen.

Case 2:n kohdalla virheiden raportointityökalut ovat olleet koekäytössä, mutta eivät kuitenkaan ole levinneet laajempaan käyttöön. Virheraportoinnin tuloksia ei tilastoida, eikä analysoida tai vertailla projektien kesken. Myös Case 1 ilmoitti, että heillä ei ole käytössä virhetietokantaa, vaan virheraportointi tehdään sähköpostilla. Haastateltava mainitsi tarpeesta työkalulle, jolla seurattaisiin ja priorisoitaisiin virheitä.

8.7 Testauksen mittaaminen

Seuratakseen testauksen etenemistä ja tehdessään arviointia testauksen kattavuudesta organisaatioyksiköiden tulisi suorittaa testauksen mittaamista. Testauksen hallinnan tehtäviin kuuluu muun muassa menetelmien määritteleminen testauksen etenemisen seurantaan. Joillakin haastatelluilla organisaatioyksiköillä testauksen mittaaminen on erittäin kattavaa kun taas toisilla ei ole käytössään minkäänlaisia testauksen mittareita.

Esimerkiksi Case 3 ilmoitti, ettei heillä mitata testausta. Myös Case 5 ilmoitti, että heillä ei ole käytössä testauksen mittareita. Haastateltava katsoi tämän olevan huono asia ja mainitsi, että testauksen mittaamisesta on kuitenkin viimeaikoina puhuttu paljon. Case 1 puolestaan ilmoitti, että heillä mitataan ainoastaan testaukseen kulunutta aikaa. Case 9:llä ei ole käytössään varsinaisia mittareita, eikä OU tee vertailua prosessien välillä. Case 2:n haastateltava sanoi, että he eivät mittaa testaamiseen kulunutta aikaa.

Joissakin tapauksissa haastateltava henkilö ei osannut vastata testauksen mittaamiseen koskevaan kysymykseen. Case 6 tiesi, että testaukseen käytetään joitakin mittareita, muttei hänellä ollut niistä tarkempaa tietoa. Case 11 ei osannut sanoa mitään testauksen mittaamisesta.

Case 4 ja 12 ilmoittivat mittaavansa testausta. Case 7:n haastateltava sanoi, että heillä mitataan testauksen tehokkuutta ja lisäksi OU käyttää standardeja apuna testauksen mittaamisessa. Erityisen kattavaa testauksen mittaaminen oli myös Case 10: kohdalla. Haastateltava sanoi, että heillä tehdään tilastointia testaukseen käytetystä ajasta ja systeemistä löytyneistä virheistä. Tätä seurataan projektitasolla. Eri ohjelmaversioista löytyneitä virheitä seurataan ja OU:ssa mitataan myös talon sisällä löydettyjä virheitä sekä asiakkaan löytämiä virheitä. Virheiden vakavuuden myös luokitellaan. Tilastot käydään läpi kerran viikossa, jotta nähdään mihin suuntaan ollaan menossa ja toimintaa ohjataan sen mukaisesti.

Case 8 ilmoitti käyttävänsä testauksen mittaukseen erillistä ohjelmistoa. Virheiden vakavuusmittaristo on käytössä. OU seuraa myös testauksen läpimenoprosenttia ja epäonnistumisprosenttia. Lisäksi OU on harkinnut mittaavansa aikaa, kuinka paljon ohjelmiston kehittäjältä kuluu virheiden korjaamiseen. Tiedot tähän on olemassa mutta raportointi puuttuu.

9 ISO/IEC 25010 - SOVELTUVUUS

Laatuominaisuudet painottuvat eri organisaatioyksiköiden toiminnassa eri tavoin. Siihen, mitkä laatuominaisuudet organisaatioyksikkö kokee tärkeimmäksi, vaikuttavat toimiala, tuotteen luonne ja tuotteen kriittisyys. Esimerkiksi, jos sovelluksen virhetilanne aiheuttaa ihmishenkien vaarantamisen, täytyy luotettavuus huomioida ehdottomasti eri tavoin kuin vähemmän kriittisissä sovelluksissa. Myös lopputuotteen käyttäjäryhmä vaikuttaa laatuominaisuuksien painottamiseen. Vaikka kaikki laatuominaisuudet ovat tärkeitä loppukäyttäjän kannalta, OU ei voi kohdentaa yhtä paljon resursseja kaikkien laatuominaisuuksien täyttämiseen. Sovelluksen luonteesta riippuen tämä ei myöskään ole välttämätöntä. Jos lopputuotteen toimintaympäristö on erittäin muuttumaton ja sovellusta ei ole tarkoitus käyttää muualla kuin tietyssä ympäristössä, yhteensopivuus ja siirrettävyys ovat luonnollisesti vähemmän tärkeitä laatuominaisuuksia. Ensimmäisen kierroksen haastattelulla organisaatioyksiköiltä kysyttiin, vaikuttaako lopputuotteen kriittisyyden muuttuminen laatuominaisuuksien erilaiseen painottumiseen. Lähes kaikki OU:t paria poikkeusta lukuun ottamatta olivat sitä mieltä, että laatuominaisuudet painottuisivat eri tavoin kriittisyyden muuttuessa. Eräs haastatelluista totesi seuraavaa kun häneltä kysyttiin, vaikuttaako kriittisyys laatuominaisuuksien painottumiseen:

> ”Ja sen pitääkin vaikuttaa, että kyllä tässä pitää olla päämäärätavoitteellinen elikkä niin kun koko toiminnan pitää ohjautua siitä päämäärästä. ”

Laatuominaisuuksien painottumiseen vaikutti osittain myös haastateltavan henkilön toimenkuva. Esimerkiksi kehittäjän näkökulmasta ylläpidettävyys on usein ehdottoman tärkeä, ellei jopa tärkein laatuominaisuus. Etenkin, jos ohjelmistotuotteen elinikä on hyvin pitkä, ylläpidettävyys nousee keskeiseen asemaan kuten Case 5:n kohdalla:

> ”No ylläpidettävyys varmaan on, tai on osottautunut aika tärkeeksi, koska nää meiän systeemit toimitetaan, niin nää saattaa olla siel 20 vuotta käytössä ja sinä aikana voi olla, että, että osia siitä tuotteesta joudutaan niinku korvaamaan jollaki muulla tekniikalla jo siinä matkan varrella. ”

Myös Case 10 painotti ylläpidettävyyttä:

> ”Joo, ja ylläpidettävyys on tietenkin yks semmonen, joka niinku, meille kehittäjille merkkaa paljon. Kyl me pyritään kirjottaa niinku sen verran laadukasta koodia, että ku, sitä ku vähän niinku tulevaaki ajatellen. Menneestä oppineena. ”

Laatuominaisuuksien painottumiseen vaikuttaa myös toimiala. Tietyllä toimialalla jotkut laatuominaisuudet voivat korostua, kuten Case 7:n kohdalla:

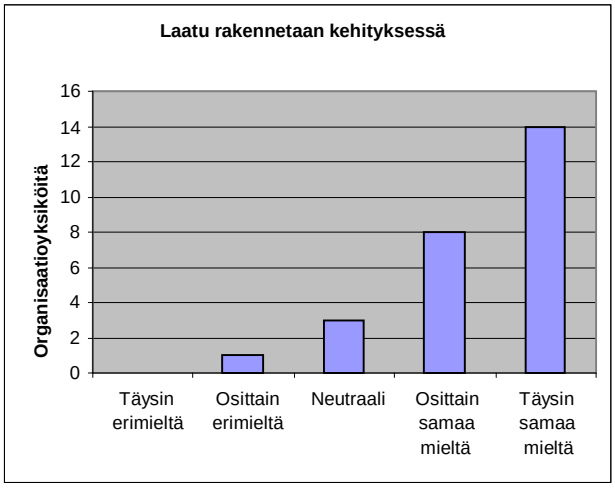
> ”Mutta tietoturvallisuus, tietoturvallisuus on edelleen, se luotettavuus on siellä kyllä ihan ykkösenä siitä huolimatta, must tällä toimialalla ne on aina. ”

Jotta laatumalli olisi hyödyllinen, täytyy olla selvillä kunkin laatuominaisuuden merkitys. Haastatteluissa ilmeni melko usein, että haastateltava ei tarkkaan tiennyt, mitkä asiat sisältyvät tiettyyn laatuominaisuuteen. Mikäli kaksi henkilöä on eri mieltä vaikkapa luotettavuus -ominaisuuden sisällöstä, voi tästä aiheutua ongelmia laatumallia sovellettaessa. Laatumallin soveltuvuuden kannalta täytyy siis olla selkeä ymmärrys siitä, mitä eri tekijöillä tarkoitetaan. Esimerkiksi Case 7: n haastateltava katsoi, että luotettavuuteen vaikuttaa kolme muuta laatuominaisuutta:

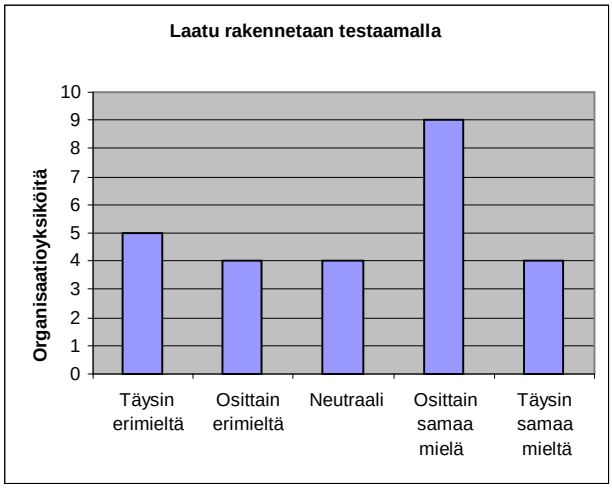
> ”Ja sit mä aattelen ton luotettavuuden sillä lailla että se on luotettava sekä, käytettävyyden, toiminnallisuuden että sen turvallisuuden kannalta, et siis sillä lailla toi luotettavuus nyt täs kohtaa kattais sit vähän enemmän.. ”

9.1 Laatu vs. Testaus

Laadun ja testauksen välisen suhteen kannalta mielenkiintoiset haastattelukysymykset olivat, rakennetaanko tuotteen laatu kehitysprosessissa vai testauksen avulla. Seuraavan sivun taulukossa on esitetty vastauksien jakautuminen. Vastauksista voidaan selvästi huomata, että valtaosa organisaatioyksiköistä on samaa mieltä väittämän ”laatu rakennetaan kehityksessä” kanssa. Laadun ja testauksen välinen yhteys jakaa mielipiteet hyvin voimakkaasti. Puolet haastatelluista organisaatioyksiköistä on osittain tai täysin samaa mieltä siitä, että laatu rakennetaan testaamalla.



Taulukko 2: Laatu ja kehitys



Taulukko 3: Laatu ja testaus

9.2 Laatuominaisuuksien painottuminen

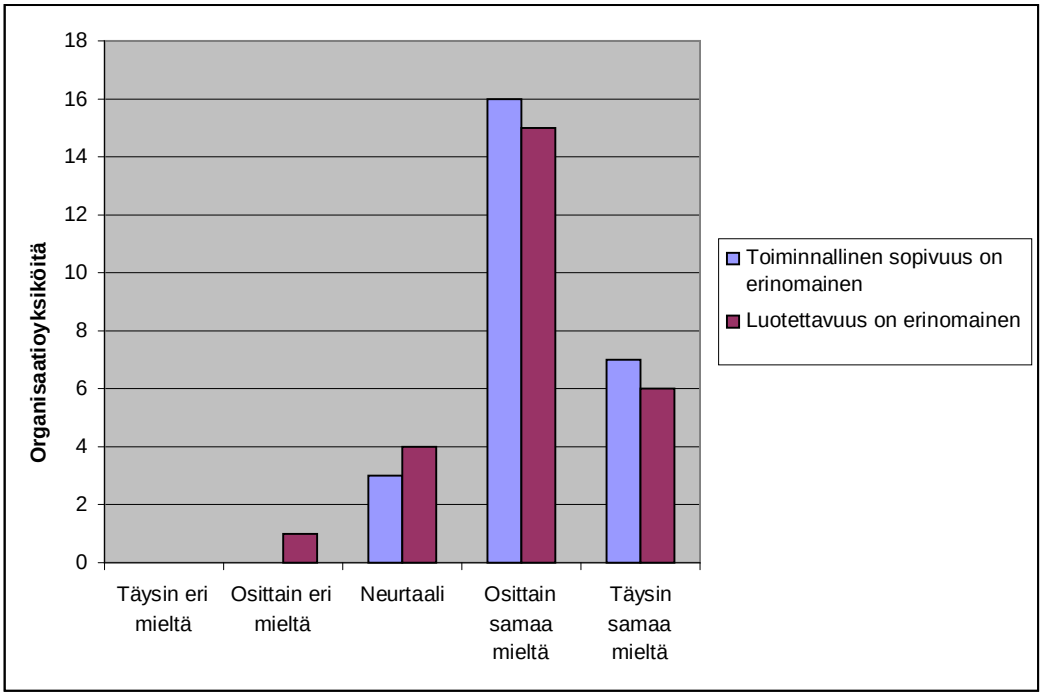
Laatuominaisuuksien tärkeysjärjestyksessä havaittiin joidenkin ominaisuuksien selkeä painottuminen. Ensimmäisellä kierroksella haastateltavilta kysyttiin, mitkä laatuominaisuudet ovat heidän näkökulmasta tärkeitä. Toiminnallisuus ja luotettavuus painottuivat selkeästi eniten – 10 OU:ta kahdestatoista valitsivat toiminnallisuuden tärkeimpien laatuominaisuuksien joukkoon. Kahdeksan OU:n mielestä myös luotettavuus kuului tärkeimpien ominaisuuksien joukkoon. Käytettävyys, tehokkuus, tietoturva ja ylläpidettävyys painottuivat vaihtelevasti tärkeimmiksi eri organisaatioyksiköiden toiminnassa. Siirrettävyyttä ja yhteensopivuutta ei yksikään OU valinnut tärkeimpien laatuominaisuuksien joukkoon.

Esimerkiksi Case 12 valitsi laatuominaisuuksista tärkeimmiksi toiminnallisuuden, luotettavuuden, tehokkuuden, käytettävyyden ja tietoturvallisuuden. Haastateltava jättäisi pois siirrettävyyden, koska heidän tuotteensa ei ole sellainen, jota myytäisiin monelle ostajalle. Järjestelmä ajetaan yhteen koneeseen ja käyttöympäristö on stabiili, joten myöskään yhteensopivuudelle tarvetta. Haastateltava totesi myös, että laatumääritelmä muuttuu kriittisyyden muuttuessa. Jos tehdään ohjelmistoja OU:n sisäiseen käyttöön, käytettävyyteen ei tarvitse kiinnittää niin paljon huomiota, kuin jos tuote tulisi julkiseen käyttöön. Tässä tapauksessa lopputuotteen käyttäjäryhmä vaikuttaa selvästi siihen, mitä laatuominaisuuksia painotetaan.

Voidaan todeta, että tärkeimmät laatuominaisuudet eivät painotu ainoastaan yrityskohtaisesti, vaan myös projektikohtaisesti. Ensimmäisen kierroksen haastatelluista organisaatioyksiköistä 3 ilmoitti,

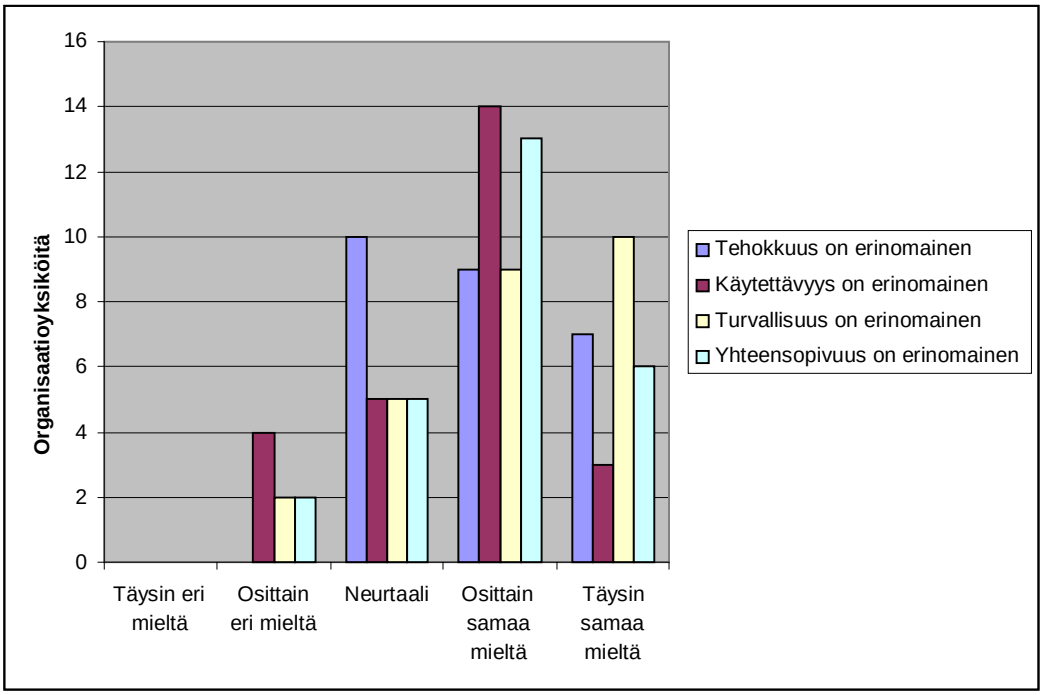
että heidän liiketoiminnassaan hankkeiden kriittisyys vaihtelee merkittävästi. Näistä organisaatioyksiköistä kaksi mainitsi, että hankkeen kriittisyys vaikuttaa sekä testausstrategiaan että laatuominaisuuksien painottamiseen. Laatumallia voitaisiin näiden organisaatioyksiköiden kohdalla käyttää mahdollisesti myös projektikohtaisesti ohjaamaan tuotekehityksen ja testauksen tavoitteita.

Toisella haastattelukierroksella organisaatioyksiköiltä pyydettiin arvioimaan omaa laatuansa ISO/IEC 25010 -laatustandardin määrittelemän laatumallin mukaan. Vastauksista voidaan huomata, että OU:t suhtautuvat laatuunsa melko optimistisesti. ”Täysin erimieltä” sekä ”osittain eri mieltä” vastauksia oli erittäin vähän kuuden laatuominaisuuden kohdalla. Näitä laatuominaisuuksia olivat toiminnallisuus, luotettavuus, turvallisuus, käytettävyys, tehokkuus ja yhteensopivuus. Laatuominaisuuksista toiminnallisuus ja luotettavuus painottuivat selkeästi eniten molemmilla haastattelukierroksella. Voidaan huomata, että organisaatioyksiköt mieltävät toiminnallisuuden ja luotettavuuden toteutumisen tärkeäksi.

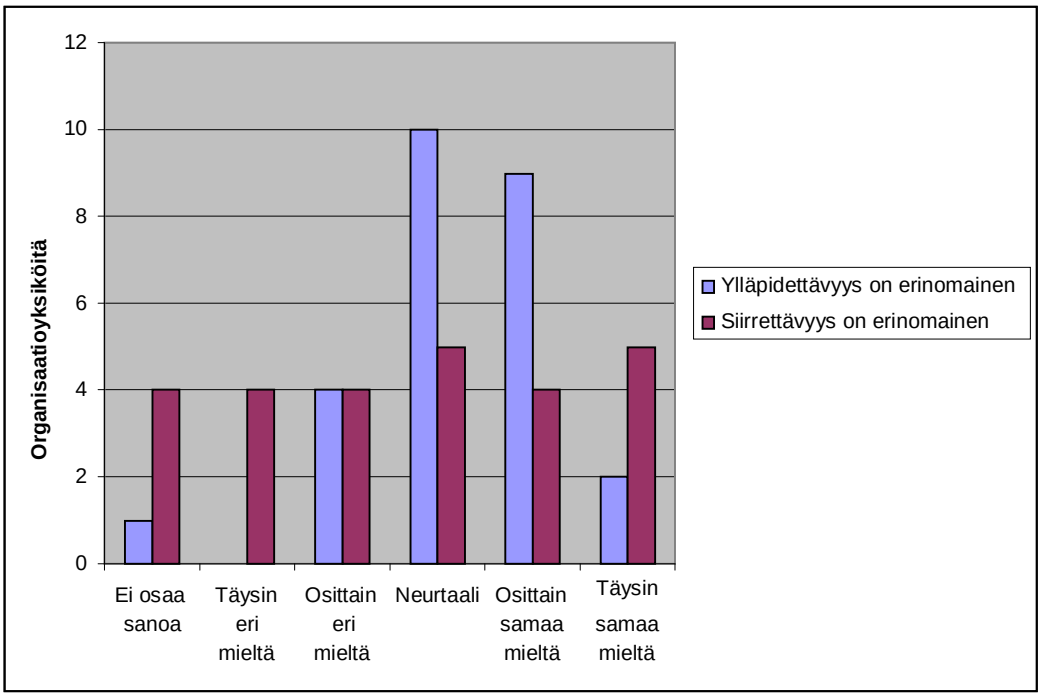


Taulukko 4: Tärkeimmät laatuominaisuudet - toiminnallisuus ja luotettavuus

23 haastatelluista organisaatioyksiköistä on osittain tai täysin samaa mieltä siitä, että heidän tuotteensa toiminnallisuus on erinomainen. 21 OU:ta puolestaan katsoo, että heidän tuotteensa luotettavuus on lähellä erinomaista.



Taulukko 5: Tehokkuus, käytettävyys, turvallisuus ja yhteensopivuus

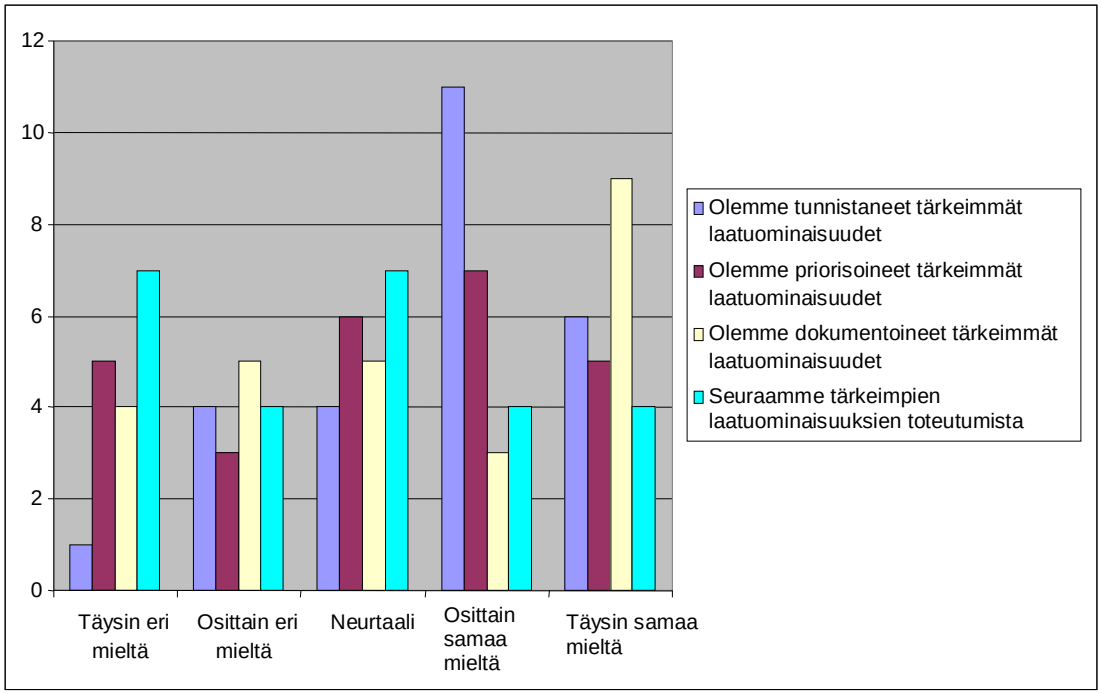


Taulukko 6: Ylläpidettävyys ja siirrettävyys

Ylläpidettävyys ja siirrettävyys aiheuttivat selvästi eniten hajontaa vastauksien jakautumisessa. Muutama OU jätti näistä toisen tekijän kokonaan arvioimatta. Siirrettävyys aiheutti mielipiteiden suurimman jakautumisen.

9.3 Laatuominaisuuksien huomioiminen toiminnassa

Toisella haastattelukierroksella organisaatioyksiköiltä kysyttiin onko OU tunnistanut ja priorisoinut tärkeimmät laatuominaisuudet, onko laatuominaisuudet dokumentoitu ja seurataanko tärkeimpien laatuominaisuuksien toteutumista mittauksilla. Alla on esitetty vastausten jakautuminen.



Taulukko 7: Laatuominaisuuksien priorisointi, dokumentointi ja seuranta

Laatustandardin soveltaminen vaatii OU:lta tärkeimpien laatuominaisuuksien tunnistamista ja priorisointia. Lisäksi laatuominaisuudet tulisi kommunikoida kaikille henkilöille, jotka osallistuvat ohjelmistokehitykseen. Laatuominaisuuksien toteutumista tulisi seurata mittaamalla. Yllä olevasta kaaviosta voidaan havaita, että merkittävä osa organisaatioyksiköistä ei huomioi laatuominaisuuksia toiminnassaan. Näille organisaatioyksiköille ISO/IEC 25010 -laatustandardi soveltuisi hyvin tukemaan ohjelmistokehitystä.

9.4 Laatumalli ja testaus

Organisaatioyksiköt voisivat hyödyntää laatumallia myös testauksen ohjaamiseen. Tärkeimmiksi nousevat laatuominaisuudet voisivat auttaa päätöksissä, mitä ei-toiminnallisen testauksen tyyppejä tulisi huomioida testausstrategiassa. Esimerkkinä voidaan ajatella, että kannettavalle laitteelle

suunnitellun sovelluksen tulisi täyttää laatuvaatimukset erityisen hyvin siirrettävyyden, käytettävyyden ja luotettavuuden osalta. Tällöin käyttämällä laatumallia ja priorisoimalla nämä tekijät, testausta voitaisiin painottaa kiinnittämällä erityistä huomiota esimerkiksi käytettävyystestiin ja muistivuototesteihin.

10 YHTEENVETO

Yritykset tarvitsevat vakiintuneita käytäntöjä tukemaan ohjelmistotuotannon eri prosesseja. Ohjelmistotuotannon laatu- ja testausstandardeja on olemassa jonkin verran, mutta on havaittu, että ne eivät ole riittävän kattavia, ja lisäksi niiden välillä on havaittu joitakin ristiriitoja. Tästä syystä on nähty tarve uusien standardien kehittämiselle. Testaus on ohjelmistokehityksen vaihe, joka aiheuttaa huomattavan osan projektin kustannuksista. Hyödyntämällä toiminnassaan testaus- ja laatustandardeja yritys voi tehostaa testausprosessiensa suorittamista ja siten alentaa projektin kustannuksia sekä parantaa lopputuotteen laatua. Testausstandardi tarjoaa yritykselle mahdollisuuden onnistuneen testauksen suorittamiseen. Laatimalla testauskäytännön ja strategian varmistutaan, että testauksessa kiinnitetään huomiota oikeisiin asioihin ja resurssit kohdennetaan oikein. Testauksen hallinnan avulla huomioidaan projektikohtaiset tarpeet ja vaatimukset testaukselle, jotta varsinainen testaustyö alimmalla tasolla voidaan suorittaa tehokkaasti ja onnistuneesti. Standardi luo yhtenäisen termistön ja kommunikoinnin projektin eri osapuolien välille ja mahdollistaa kaikkien osapuolien näkemysten huomioimisen ohjelmistotestauksessa.

Haastateltujen organisaatioyksiköiden valmius soveltaa ISO/IEC 29119 testaus- ja laatustandardia vaihtelee sen mukaan, miten pitkälle OU:n testausprosessit ovat kehittyneet. OU:n sisäinen suhtautuminen muutoksiin ja johtohenkilöstön taidot vaikuttavat osaltaan organisaatioyksikön kykyyn soveltaa standardeja. Testaukseen käytettävissä olevat resurssit, erityisesti päätoimisten testaajien määrä vaikuttavat myös merkittävästi siihen, kuinka laajassa mittakaavassa OU voi standardia soveltaa. Yritysten tulisi valita standardista itselleen sopivimmat osat, joita soveltaa toimintansa tukena. Haastatteluissa ilmeni, että jokaisella OU:lla on jotakin parannettavaa testauksen menettelytavoissaan huolimatta siitä, olivatko OU:n testausmenetelmät alkutekijöissään tai pitkälle kehittyneitä. Ensimmäisen haastattelukierroksen OU:t panostaisivat useammin testauksen strategiaan ja hallintaan.

ISO/IEC 25010: määrittelemän laatumallin ominaisuudet painottuvat eri tavoin haastateltujen organisaatioyksiköiden kohdalla. Laatuominaisuuksien painottumiseen, vaikuttavat toimiala, tuotteen luonne ja tuotteen kriittisyys. Lisäksi vaikuttavia tekijöitä olivat haastateltavan henkilön toimenkuva sekä käyttäjäryhmä, jota varten ohjelmistotuote kehitetään. Laatuominaisuuksista toiminnallisuus ja luotettavuus painottuivat selkeästi eniten kun taas siirrettävyyden kohdalla hajonta oli suurin. On tärkeää tunnistaa ja priorisoida ohjelmistotuotteen laatuominaisuudet, jotta

kehitysprosessin päätöksiä ja valintoja voidaan ohjata näiden ominaisuuksien mukaisesti. Tuloksista havaittiin että, moni OU ei ohjaa toimintaansa laatuominaisuuksien mukaan. Laatumallia voisi mahdollisesti käyttää myös testauksen suunnittelussa ohjaamaan päätöksiä siitä, mitä ei-toiminnallisia testauksen tyyppejä tehdään.

LIITTEET

Liite 1: Testauspolitiikan mahdollinen sisältö. [5]

ID	Otsikko	Esimerkki
1	Testauksen tavoitteet	Testauksen tarkoituksena on lieventää riskejä.
2	Testauksen laajuus	Kaikki asiakkaalle toimitettava ohjelmisto testataan.
3	Testausprosessi	Yritys noudattaa ISO 29119-2:n määrittelemää testausprosessia.
4	Itsenäisyyden aste	Testaamisen hoitaa ryhmä, joka on teknisesti, hallinnollisesti ja taloudellisesti kehitysorganisaatiosta erillinen.
5	Testausorganisaation rakenne	
6	Testaajien koulutus	Kaikkien testaajien täytyy osata ohjelmoida.
7	Testausprosessin parantaminen	Kaikki projektit käyttävät TPI:tä.
8	Standardit	Testausdokumentoinnissa noudatetaan ISO 29119 standardia.
9	Testauksen elinkaarimalli	Testaaminen noudattaa määrättyä elinkaarimallia, joka sovitetaan kehityksen elinkaarimalliin.
10	Organisaation muut käytännöt	Testaus noudattaa yrityksen laatupolitiikkaa.
11	Testauksen arvon mittaaminen	Yritys mittaa testauksen ROI:tä.
12	Testaajan eettisyys	Testaajat toimivat eettisesti määritetyn mallin mukaisesti.

Liite 2: Testausstrategian mahdollinen sisältö. [5]

ID	Kohdan nimi	Esimerkki
1	Testauksen tavoitteet	Tuki riskien lieventämiselle
2	Testauksen laajuus	Kaikki asiakkaalle toimitettava ohjelmisto testataan
3	Testauksen menetelmät	
4	Testausprosessi	• Sisältää testauksen seuranta- ja hallintaprosessit • Sisältää testauksen dokumentoinnin määrittelyn • Organisaatio seuraa ISO 29119-2:n määrittelemää testausprosessia • Organisaatio seuraa IEEE 2012:ssa määriteltyä staattista testausprosessia
5	Testaajan vastuut	Mitä tehtäviä testaajat suorittavat
6	Itsenäisyyden aste	Testauksen suorittaa testausryhmä, joka on teknisesti, hallinnollisesti ja taloudellisesti erillinen kehitysosastosta
7	Testiorganisaation rakenne	Voi sisältää testaustiimiin profiilin
8	Testaajan koulutus	Kaikkien testaajien on osattava ohjelmoida tai testaajilla täytyy olla tietyn tason koulutus
9	Testausprosessin parantaminen	
10	Standardit	Testauksen dokumentoinnissa noudatetaan ISO 29119 -standardia
11	Testauksen elinkaarimalli	Testauksella on tietty elinkaarimalli joka on yhteydessä kehityksen elinkaareen
12	Organisaation muut strategiat	
13	Testauksen arvon mittaaminen	Organisaatio seuraa sijoitetun pääoman tuottoa (ROI) testauksen osalta
14	Testauksen tyypit	Toiminnallinen testaus (lasilaatikkotestaus, mustalaatikkotestaus, ym.), ei-toiminnallinen testaus (luotettavuustestaus, käytettävyystestaus, ym.)
15	Testauksen tekniikat	
16	Testauksen tasot	
17	Pääsy- ja poistumiskriteerit	
18	Testauksen päättämisen kriteerit	
19	Uudelleentestaus ja regressiotestaus	
20	Testien valinta ja priorisointi	
21	Testauksen ympäristöt	
22	Testien uudelleenkäytettävyys	
23	Testauksen automaatio ja työkalut	• Lähestymistapa testauksen automatisointiin • Mitä testauksen työkaluja käytetään
24	Yleiset riskit	

Liite 3a: Testauksen hallinnan sisältämät tehtävät. [5]

1. Määritä testauksen tavoitteet; esimerkiksi mahdollisimman korkean luotettavuuden tason saavuttaminen käytettävissä olevien 3 päivän aikana.
2. Päätä, mitä testataan. Esim. testataan strategian mukaan ja lisäksi kaikki tietyn henkilön kirjoittamat komponentit.
3. Päätä, mitä testaustekniikoita käytetään. Mikäli testausstrategia on määritelty, voidaan joko noudattaa strategiaa vai poiketaanko strategiasta joiltakin osin. Esim. noudatetaan testausstrategiaa paitsi niiden luokkien osalta, jotka toteuttavat tilakaaviot, joiden kohdalla myös tilasiirtyminen testataan.
4. Määritä testauksen päättämiskriteerit.
5. Päätä mitä testauksen automaation lähestymistapaa käytetään. Esim. tässä projektissa asiakas tarjoaa kaikki testaustyökalut.
6. Päätä mitä testaustyökaluja käytetään. Esim. tätä projektia varten meidän täytyy rakentaa työkalu, joka simuloi laitteistoa, joka on tämän projektin kannalta
7. Määritä testauksen itsenäisyyden odotettu taso. Esimerkiksi tässä projektissa turvallisuustestaus toteutetaan tietyn konsultointiyrityksen toimesta.
8. Määritä testaajille henkilöstöprofiili (numerot, roolit, jne.). Esim. Määritä kuka tekee tietyt työt, mitä henkilöitä tarvitaan, ja mitkä asiat kuluvat ulkoiselle toimijalle.
9. Päätä, mitkä ominaisuudet järjestelmästä on määrä testata. Esim. Tietyt funktiot ja tietyt ei-toiminnalliset tekijät tässä projektissa vaativat erityishuomiota. Esimerkiksi turvallisuus testataan funktioiden X ja Y kohdalta sekä käytettävyys funktion Z kohdalla.
10. Päätä, mitä tehtäviä yksittäiset testaajat suorittavat. Esim. Testaaja A hoitaa turvallisuustestauksen ja testaaja B huolehtii kaikkien suunnitelmien arvioinnista.
11. Päätä vaadittava testausympäristön tuki.
12. Määritä virheiden hallintaprosessi ja sitä tukevat työkalut. Esim. käytetään IEEE 1044:n määrittämää poikkeusraportointiprosessia ja Bugzilla -työkalua virheiden raportointiin.
13. Päätä, mitkä testauksen tasot (esim. yksikkö, integraatio, järjestelmä, hyväksyntä) on määrä toteuttaa. Esim. Tässä projektissa kehittäjät huolehtivat muodollisesta ja dokumentoidusta yksikkötestauksesta.
14. Määritä mitä priorisoimisen keinoja käytetään päättämään, mitä testataan ja missä järjestyksessä. Esim. tässä projektissa asiakas haluaa käyttää heidän erityistä lähestymistapaa riskien analysointiin, joka on perustana meidän normaalille riskipohjaiselle testauksen lähestymistavalle.
15. Päätä, mitä sääteleviä standardeja pitää käyttää. Esim. projektitasolla tämä voi tarkoittaa määrittämistä, mitkä alijärjestelmät noudattavat tiettyjä standardeja.
16. Määritä testauksen vaiheiden aikataulutus (suhteessa kehityksen vaiheisiin). Esim. tämän projektin hyväksymistestaus on suunniteltu toteutettavaksi 3 kuukauden päästä joulusta 2006.
17. Määritä menetelmät testauksen etenemisen seurantaan. Tämä sisältää mittarit, joita seurataan ja analysoidaan, ja jotka on tälle projektille erityisiä.
18. Määritä regressiotestauksen ja uudelleentestauksen muodot, joita odotetaan. Esim. täydelliset regressiotestit suoritetaan vain, kun alijärjestelmään X on tehty muutoksia.
19. Määritä lähestymistapa testausprosessin parantamiselle.
20. Dokumentoi, mikä standardi määrittää käytettävän testausdokumentoinnin. Esim. lentoelektroniikkastandardia käytetään tähän turvallisuuskriittiseen järjestelmään;

Liite 3b: Projektin testaussuunnitelman sisältö. [5]

1	Projektin testausstrategia	• Riskit, joita testauksella lievennetään • Testauksen tyypit ja tasot, joita käytetään pienentämään kutakin riskiä • Testausympäristön vaatimukset • Testaustyökalujen vaatimukset • Projektin testauksen päättämisen kriteerit
2	Henkilöstöprofiili	• Yksittäisten testaajien tehtävät ja vastuut • Rekrytoinnin tarve • Koulutuksen tarve
3	Testaukseen liittyvä aineisto	• Testausdokumentaatio
4	Testauksen tehtävät	• Tehtävät, joita testauksen suorittaminen alusta loppuun vaatii
5	Testauksen aikataulu	• Arviot, kuinka kauan kunkin tehtävän suorittaminen
6	Poikkeavuudet organisaation testausstrategiasta	• Dokumentoidaan, mikäli niitä on
7	Testausprosessin parantaminen	• Lähestymistapa, jota käytetään testausprosessin parantamiskeksi

Liite 4: Tason testaussuunnitelman mahdollinen sisältö. [5]

1	Tavoitteet	Mittaus / laatu
2	Laajuus	• Taso tai tyyppi
3	Viittaukset	
4	Olettamukset ja rajoitukset	• Sisältäen säätelevät standardit
5	Testaukseen liittyvät riskit	• Tuote • Projekti
6	Projektin testausstrategia	• Tekniikat • Testauksen päättämiskriteerit • Automaatio ja työkalut • Ominaisuudet, jotka testataan • Ominaisuudet, joita ei testata • Arviot • Metodi edistymisen seurantaan o Mittaustiedot, joita kerätään • Regressiotestauksen lähestymistapa • Ristiviittaukset riskeihin • Keskeyttämisen ja jatkamisen kriteerit • Testausympäristöt • Testausdata • Itsenäisyyden taso
7	Aikataulu	
8	Henkilöstö	• Vastuut • Rekrytointi • Koulutus
9	Poikkeukset organisaation testausstrategiasta	

LÄHDELUETTELO

[1] Taipale, O., Smolander, K., Kälviäinen, K., *Cost Reduction and Quality Improvement in Software Testing*, Lappeenranta University of Technology

[2] Evans, I., *ISO 29119 Presentation*, 2008

[3] Haikala, I., ja Märijärvi, J., *Ohjelmistotuotanto*, Talentum, Helsinki, 2004.

[4] ISO, *International Software Quality Standard*, ISO/IEC 25010, 2008

[5] ISO, *International Software Testing Standard*, ISO/IEC 29119, 2008

[6] Juvonen, P., Laadullinen Tutkimus Metsäteollisuuden ohjelmistotoimittajista Kaakkois-Suomessa: Nykytilanne, Lappeenranta University of Technology, 2005

[7] Moore, J., *Software Engineering Standards: A User's Roadmap*, IEEE Computer Society, Los Alamitos, California, 1998.

[8] Myers, Glenford J., *The Art of Software Testing*, John Wiley & Sons, Inc. Hoboken, New Jersey, 2005

[9] Robillard, P., Krucheten, P., d'Astous P., *Software Engineering Process with the UPEDU*, Addison Wesley, 2003

[10] Taipale, O., *Masto Hypothesis*, Lappeenranta University of Technology, 2008

[11] Taipale, O., Karhu, K., Smolander K. *Observing Testing Schedule Over-runs from Knowledge Transfer Viewpoint*, Lappeenranta University of Technology

[12] ISO/IEC 15504-1 Information Technology – *Process Assessment – Part 1: Concepts and Vocabulary*, 2002