Testaussuunnitelman merkitys ketterässä ohjelmistokehityksessä

Outi Ruusunen

Haaga-Helia ammattikorkeakoulu
Tradenomi
Opinnäytetyö
2022

Tiivistelmä

Tekijä(t)
Outi Ruusunen

Tutkinto
Tradenomi

Raportin/Opinnäytetyön nimi
Testaussuunnitelman merkitys ketterässä ohjelmistokehityksessä

Sivu- ja liitesivumäärä
31 + 8

Useat erilaiset ketterän ohjelmistokehityksen viitekehykset tarjoavat ohjenuoria ja suosituksia ohjelmistokehityksen toteutukseen. Näissä viitekehyksissä testaus mielletään kiinteäksi osaksi ohjelmistokehitystä, eikä testauksenhallintaan liittyviä seikkoja ole siten juuri erikseen huomioitu. Ohjelmistotestauksen kansainväliset standardit ohjaavat testauksenhallintaa organisaatiotasolta aina alemmille testaustasoille saakka, mutta testauksenhallinta ei tavallisesti näyttäydy sellaisenaan ketterässä ohjelmistokehityksessä, vaan on sulautunut ketterän organisaation toimintatapoihin.

Opinnäytetyön ohjaavana ajatuksena on, että ketterin menetelmin toteutettavassa ohjelmistokehityksessä testaussuunnitelman dokumentointi on hyvin kevyttä ja vaihtelevaa, tai se jätetään jopa kokonaan laatimatta. Tämä voi johtaa pahimmillaan jopa siihen, että ohjelmistokehitysprosessi kokonaisuudessaan on tehoton ja aikaa (ja rahaa) hukataan tarpeettomasti. Testaussuunnitelman laatiminen voisi auttaa selkiyttämään rooleja ja vastuita, lisäisi läpinäkyvyyttä eri tasoilla tehdyistä testaustehtävistä ja parantaisi myös parhaassa tapauksessa tuotteen laatua, kun riskit ja muut mahdolliset ongelmakohdat on huomioitu niin organisaatiotasolla kuin alemmilla testaustasoillakin.

Opinnäytetyön tietoperustassa selvitetään yleisellä tasolla, mitä tässä opinnäytetyössä tarkoitetaan ketterällä ohjelmistokehityksellä ja avataan testauksen toteutukseen liittyviä keskeisiä käsitteitä, kuten testaustasot, testauksenhallinta ja testaussuunnitelma. Työssä myös vertaillaan testaussuunnitelman laatimiseen liittyviä erityispiirteitä eri ohjelmistokehitysmalleissa.

Työn empiirisessä osassa kartoitetaan, millä eri tasoilla testaussuunnitelmia tehdään yrityksissä ja organisaatioissa, joissa toimitaan ketterän ohjelmistokehityksen viitekehyksessä. Opinnäytetyössä pyritään selvittämään myös, kuinka testauksen suunnittelua toteutetaan erilaisissa ketterän ohjelmistokehityksen viitekehyksissä ja toimintamalleissa.

Opinnäytetyön tuloksena tuotettiin ketterään ohjelmistokehitykseen soveltuvan ohjelmistokehitysprosessin malli, jossa testaus ja testauksenhallinta näyttelee nykyistä suurempaa roolia.

Asiasanat
ketterä ohjelmistokehitys, testaus, testaussuunnitelma, testauksenhallinta, dokumentaatio

Sisällys

1 Johdanto

Testauksen menetelmät ovat vuosien saatossa kehittyneet ja myös eri testaustehtävien automatisaatiota on kehitetty kasvavalla volyymilla. Testaus on yksi laadunhallinnan tärkeimmistä elementeistä ohjelmistokehityksessä. Testaukseen on syytä panostaa jo siitäkin syystä, että sen avulla voidaan vähentää kustannuksia, kun virheitä ei päädy tuotantoon saakka. Nopeassa kehityssyklissä jatkuvan palvelutuotannon turvaamiseksi myös testauksen on pysyttävä mukana kehityksessä. Tästä syystä testaus tulee integroida osaksi koko ohjelmistokehityksen elinkaarta. Ketterissä toimintamalleissa palautetta kehitettävästä tuotteesta voidaan saada joustavasti nopeiden kehityssyklien avulla.

Ketterän ohjelmistotuotannon toimintamalleissa testaus nähdään yhtenä osakokonaisuutena, joka tehdään muiden työtehtävien rinnalla. Malleissa ei kuitenkaan käsitellä useinkaan kovin perinpohjaisesti sitä, miten ohjelmistotestausta tulisi tehdä, vaan miten se liittyy muihin ohjelmistoprosessin työvaiheisiin. (Kasurinen 2013.) Testauksen integrointi ohjelmistokehityksen varhaisiin vaiheisiin edellyttää organisaatioilta strategista suunnittelua sekä laadunhallinnan mieltämistä kiinteäksi osaksi ohjelmistokehitystä.

Työskenneltyäni asiantuntijana testauksen ja testauksen kehittämisen tehtävissä, olen pannut merkille, että testaussuunnitelmien sisällöissä ja dokumentoinnissa ylipäänsä on huomattavan paljon vaihtelevia käytäntöjä. Tästä syystä halusin opinnäytetyössäni selvittää, millaisia testaussuunnitelmia laaditaan ketterin menetelmin ohjelmistoja kehittävissä yrityksissä ja organisaatioissa. Työssä selvitetään, millä eri testaustasoilla testaussuunnitelmia laaditaan, miten ne dokumentoidaan ja mikä merkitys testaussuunnitelmalla on testauksenhallinnan näkökulmasta. Tämä tieto kerätään kyselytutkimuksena, jossa selvitetään testaussuunnitelman merkitystä testauksenhallinnan työvälineenä. Tutkimusinstrumenttina käytetään puolistrukturoitua kyselyä, joka on suunnattu ketterissä ohjelmistokehitysprojekteissa työskenteleville IT-alan ammattilaisille.

Työssä ei analysoida testauspolitiikan tai -strategian toteutumista organisaatioissa, vaikka ne ovatkin merkittävä osa testauksenhallintaa. Työn tarkoitus ei myöskään ole vertailla eri testausmenetelmiä ja -tekniikoita keskenään, vaan pyrkiä selvittämään, kuinka ketteriin ohjelmistokehitysmalleihin perustuvissa ohjelmistoprojekteissa testaussuunnitelmia laaditaan ja koetaanko testaussuunnitelmista saatavan tiedon olevan riittävällä tasolla nykyisissä toimintamalleissa. Kyselytutkimuksen tuloksena odotan saavani vahvistusta olettamukselle siitä, että testaussuunnitelmia ei juurikaan laadita, etenkään ylätasolla. Toivon opinnäytetyöni tarjoavan uusia näkökulmia testauksenhallinnan ja ohjelmistokehitysprosessien tueksi.

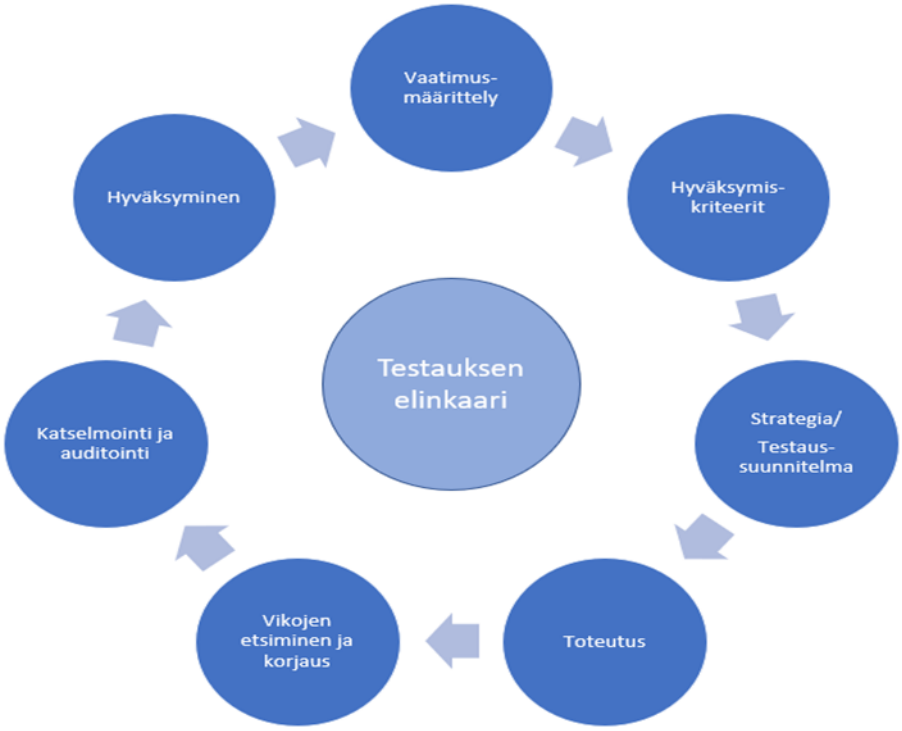
2 Tietoperusta

Kansainvälisten ISO/IEC/IEEE 29119 -sarjan standardit määrittelevät yhteisesti sovitut normit ohjelmistotestauksen ohjaamiseen, hallintaan ja toteuttamiseen missä tahansa organisaatiossa, projektissa tai testaustoiminnassa. Sen eri osa-alueissa käsitellään testauksen käsitteitä ja määritelmiä, testausprosesseja, testien suunnittelua ja testauksen menetelmiä ja tekniikoita, testausdokumentaatiota, avainsanaohjattua (automaatio)testausta sekä ketterien projektien ja tekoälypohjaisten järjestelmien testausta varten. (IEEE 2013.)

International Software Testing Qualifications Board (FiSTB), on ohjelmistotestauksen asiantuntijoiden sertifioinnista vastaava yhteisö. Sen tuottamat sertifiointisisällöt pohjautuvat edellä mainittuihin alan yleisesti hyväksyttyihin standardeihin. Tässä luvussa käsitellään testausta erityisesti näiden sertifiointisisältöjen ja FiSTB:n suomalaisen aliorganisaation Finnish Software Testing Boardin (2015a) suomeksi kääntämän, vakiintuneen toimialasanaston näkökulmasta. Lisäksi tässä luvussa kuvataan, mitä tarkoitetaan ohjelmistokehityksen ja testauksen dokumentaatiolla sekä esitellään muutamia yleisimmin käytössä olevia ketterän ohjelmistokehityksen elinkaarimalleja. Luvussa pyritään myös avaamaan sitä, miten testaussuunnitelma laaditaan erilaisissa ketterän ohjelmistokehityksen malleissa.

2.1 Ohjelmistotestaus

Ohjelmistotestauksella tarkoitetaan prosessia, johon kuuluu lukuisia eri tehtäviä, kuten testauksen suunnittelu, testien analysointi, suunnittelu ja toteutus, testauksen suoritus, testauksen edistymisen ja tuloksien raportointi sekä testauksen kohteen laadun arviointi. Sen avulla voidaan arvioida ohjelmiston laatua sekä pienentää tuotantokäytössä tapahtuvien häiriöiden riskiä. Joissain tapauksissa ohjelmistotestausta tarvitaan myös sopimuksellisten tai lakeihin pohjautuvien tai tietyn teollisuudenalan standardeihin liittyvien vaatimusten täyttämiseksi. (FiSTB 2018, 12-13.) Vaikka testaamiseen kuuluvatkin kaikki edellä mainitut osa-alueet, on mekaaninen testaustyö kehitysvaiheen aikana keskeisessä asemassa (Kasurinen 2013, 39). Kuvassa 1 on kuvattu testauksen eri vaiheet aina määrittelyjen identifioimisesta järjestelmän hyväksymiseen asti.



Kuva 1. Ohjelmistotestauksen elinkaari (mukaillen Khania ym. 2016)

2.2 Testauksen tasot

Testaustasoilla tarkoitetaan testaustehtävien joukkoa, joita organisoidaan ja hallitaan yhdessä. Testaustasot liittyvät aina muihin ohjelmistokehityksen elinkaaren tehtäviin, joten testausta tehdään kaikissa vaiheissa. (FiSTB 2018, 27.)

Yksikkötestauksella tarkoitetaan yksinään testattaviin komponentteihin keskittyvää testausta. Sen tavoitteena on ohjelmiston riskien pienentäminen, järjestelmän määritysten mukaisen toiminnallisen ja ei-toiminnallisen käyttäytymisen todentaminen, komponenttien vikojen löytäminen sekä vikojen etenemisen estäminen ylemmille testaustasoille. Yksikkötestauksen pohjamateriaalina voidaan käyttää yksityiskohtaisia suunnitelmia, koodia, tietomallia sekä komponenttien määrittelyä. (FiSTB 2018, 27-28.)

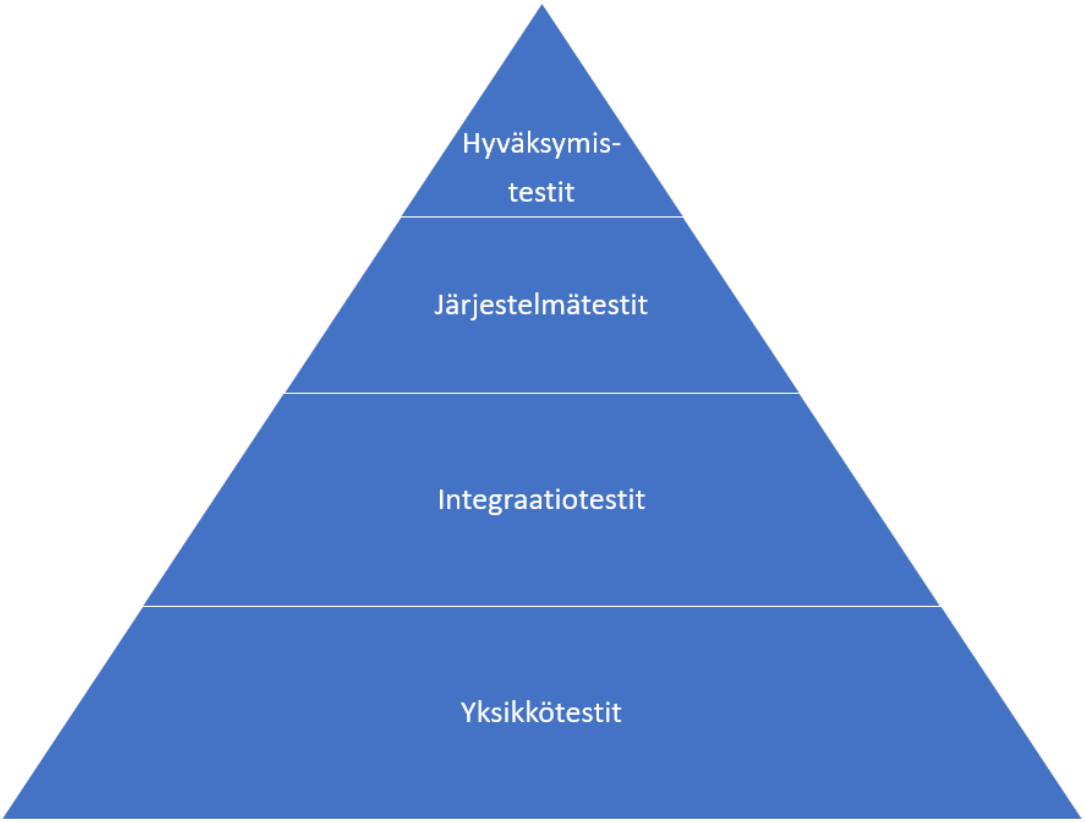
Integrointitestaus seuraa yksikkötestausta. Sen tärkeimpänä tavoitteena on varmistaa järjestelmän eri osien toimivuus kokonaisuutena. Integrointitestauksessa todistetusti toimi-vaan kokonaisuuteen siis lisätään uusia osia ja tarkistetaan, että kokonaisuus toimii edelleen. (Kasurinen 2013, 42.) Sen avulla pyritään kasvattamaan luottamusta rajapintojen laatuun, pienentää riskejä tiedon välittymisessä eri komponenttien tai järjestelmien välillä sekä sen todentaminen, että järjestelmä

vastaa toiminnalliselta ja ei-toiminnalliselta käyttäytymiseltään suunniteltua ja määriteltyä. (FiSTB 2018, 27-28.)

Järjestelmätestauksessa keskitytään koko järjestelmän päästä päähän tapahtuvaan toiminnalliseen ja ei-toiminnalliseen käyttäytymiseen. (FiSTB 2018, 31-32.) Järjestelmätestauksen tarkoitus Kasurisen (2013, 45) mukaan on testata järjestelmää toiminnallisena kokonaisuutena testiympäristössä. Tämän erottaa hyväksymistestauksesta se, että järjestelmätestauksessa myös yksittäiset komponentit ovat tarkastelun kohteena. Tyypillisesti tässä vaiheessa järjestelmälle ei kuitenkaan enää aiota tehdä merkittäviä muutoksia. (Kasurinen 2013, 45.) Joidenkin lähteiden mukaan järjestelmätestauksesta käytetään myös termiä ”kokonaistestaus”, vaikkakaan FiSTB:n sertifiointisisällössä kokonaistestausta ei käsitteenä mainita muutoin kuin testaussanastossa, jossa termillä viitataan testaussuunnitelmaan, joka kattaa useita testaustasoja.

Hyväksymistestaus keskittyy järjestelmätestauksen tavoin tyypillisesti koko järjestelmän tai tuotteen ominaisuuksiin ja käyttäytymiseen. Sen tavoitteena on lisätä järjestelmään laatuun kohdistuvaa luottamusta sekä varmistaa, että järjestelmä on valmis ja toimii odotetusti ja määrittelyjen mukaisesti (FiSTB 2018, 32-33.). Hyväksymistestauksen tarkoituksena on antaa asiakkaalle mahdollisuus testata tuote kohdeympäristössä vielä ennen sen hyväksy-mistä käyttöönottoon, jolloin ohjelmisto siirtyy laiteteknisesti asiakkaan omaisuudeksi tai ainakin kehityksenaikainen huolto- ja korjausvelvoite päättyy (Kasurinen 2013, 45).

Ketterässä ohjelmistoprojektissa tehdään tyypillisesti eniten testausta yksikkötestien tasolla ja vähennetään testien määrää siirryttäessä kehityksessä ylemmille tasoille. Tällaisella laadunvarmistuksen mallilla pyritään siihen, että viat pyritään poistamaan mahdollisimman aikaisessa ohjelmistokehityksen elinkaaren vaiheessa. Testaustehtäviä pyritään ketterässä ohjelmistokehityksessä automatisoimaan kaikilla testaustasoilla. (FiSTB 2015.) Kuvassa 1 on havainnollistettu, miten eri testaustehtävät vähenevät, mitä ylemmille tasoille kehitystyössä edetään.



Kuva 2. Testaustasojen pyramidimalli (FiSTB 2018)

2.3 Ohjelmistokehityksen elinkaarimallit

Ohjelmistokehityksen elinkaarimalli määrittelee, minkälaisia vaiheita ohjelmistokehitysprosessiin kuuluu. Tyypillisesti elinkaarimalli seuraavista vaiheista: vaatimusmäärittely, suunnittelu ja analyysi, kehitys, testaus ja ylläpito. Näiden toteutusjärjestys vaihtelee elinkaarimallin mukaan. (FiSTB 2018.) Hyvin tunnettuja elinkaarimalleja ovat mm. vesiputousmalli, V-malli, spiraalimalli sekä erilaiset iteratiiviset ja inkrementaaliset, eli ketterät kehitysmallit.

Sopiva ohjelmistokehityksen elinkaarimalli tulee valita ja muokata projektin tavoitteiden, kehitettävänä olevan tuotteen tyypin, liiketoiminnan prioriteettien (esim. markkinoillesaantiaika) sekä tunnistettujen tuote- ja projektiriskien mukaan (Lampinen 2019, ??). Ohjelmisto-kehityksen elinkaarimalleja voidaan myös yhdistellä keskenään. V-mallia voidaan käyttää esimerkiksi taustajärjestelmien ja niiden integroinnin kehittämiseen ja testaamiseen, kun taas ketteriä, inkrementaalisia ja iteratiivisia kehitysmalleja käytetään tyypillisesti web-sovellusten kehittämisessä ja erityisesti niiden käyttöliittymien ja toiminnallisuuksien testaamisessa. Ketterässä ohjelmistokehityksessä käytetään yleisesti ohjelmiston prototyyppejä, jotka voidaan laatia varhaisessa vaiheessa. Tällöin inkrementaaliseen kehitysmalliin siirrytään, kun kokeiluvaihe on valmis. (FiSTB 2018.)

2.3.1 Vesiputousmalli

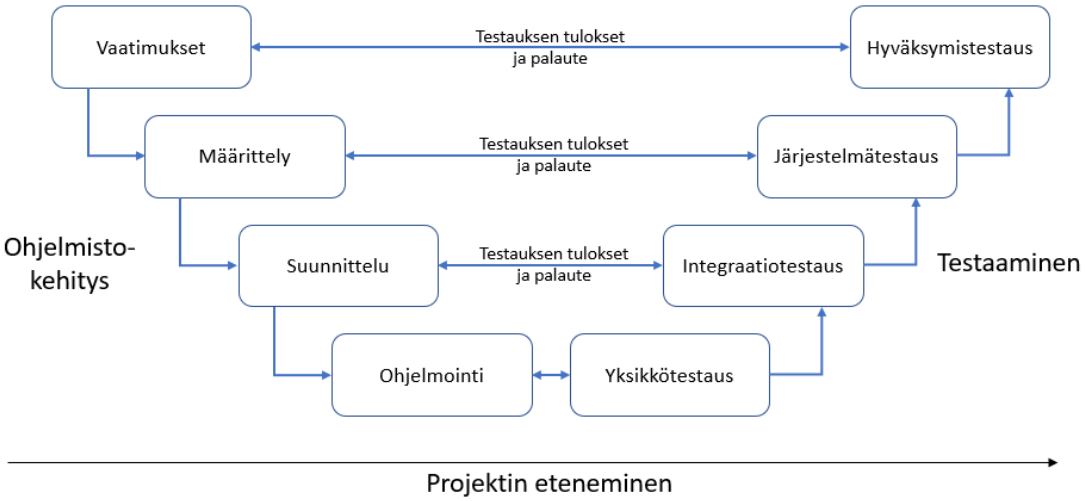
Vesiputousmallilla tarkoitetaan tapaa toteuttaa ohjelmistokehitystä siten, että kehitystehtävät, kuten vaatimusten analysointi, suunnittelu, koodaus ja testaus, tehdään valmiiksi yksi toisensa jälkeen. Näin kehitysprosessin uusi vaihe voi alkaa vasta, kun edeltävän vaiheen tuotokset on testattu ja todettu valmiiksi. Tästä syystä vesiputousmallia kuvataan peräkkäismalliksi, koska prosessi etenee lineaarisesti, tehtävien perättäisenä sarjana. Vaiheet eivät tavallisesti mene päällekkäin, mutta palautetta kehityksenaikaisista tuotoksista tulisi saada mahdollisimman aikaisessa vaiheessa seuraavan vaiheen käynnistyttyä. (FiSTB 2018, 25.) Tällaista ohjelmistokehityksen tapaa pidetään vanhentuneena ja osin ongelmallisena, sillä eri työvaiheiden valmistumisen odottelusta aiheutuu turhaa viivettä ja hukkaa.

2.3.2 V-malli

V-mallissa ohjelmistokehitys etenee jotakuinkin vesiputousmallin tavoin, mutta testausta pyritään tekemään kaikissa ohjelmistokehityksen vaiheissa. Tässä mallissa ohjelmointityön tarkastuksessa käytetään yksikkötestejä. Suunnitelmien toteutuminen puolestaan validoidaan integrointitestein. Määrittelyjen paikkansapitävyys todennetaan järjestelmätestauksella, ja järjestelmän vaatimukset hyväksymistestauksella. (Kasurinen 2013, 13.)

Perinteisessä V-mallissa, kuten monessa muussakin ohjelmistotuotannon peräkkäismallissa, heikkoutena on se, että testaus alkaa liian myöhään. Esimerkiksi vaatimusmäärittely tai järjestelmäarkkitehtuurin laatiminen tapahtuvat paljon ennen testauksen aloittamista, vaikka mielekkäämpää voisi olla testauksen kannalta miettiä sitä, miten vaatimuksien täyttyminen voidaan todentaa, tai miten järjestelmiä voisi rakentaa siten, että järjestelmän osia päästään testaamaan ennen koko paketin valmistumista. (Kasurinen 2013, 13.)

V-malli koostuu kuitenkin kutakin toteutusvaihetta vastaavista testaustasoista, mikä omalta osaltaan tukee aikaista testausta. Tässä mallissa kuhunkin testaustasoon liittyvien testien suoritus etenee järjestyksessä peräkkäin, mutta joissain tilanteissa tapahtuu päällekkäisyyksiä. (FiSTB 2018, 25.) Kuvassa 3 kuvataan V-mallin mukaisesti, miten eri ohjelmistokehityksen vaiheissa tehdyllä testaustehtävillä validoidaan kunkin ohjelmistokehitysvaiheen tuotoksia.



Kuva 3. V-malli (mukaillen Kasurista 2013)

2.3.3 Spiraalimalli

Spiraalimallissa luodaan tavallisesti kokeellisia tuoteversioita, eli prototyyppejä, joista osa ehkä toteutetaan laajastikin uudelleen tai jopa hylätään myöhemmän kehitystyön aikana. (FiSTB 2018, 26.) Spiraalimallissa edetään suunnittelusta lopputuotosta kohti lisäten jokaisella iteraatiokierroksella kehityksen tasoja. Ensimmäisellä kehityskierroksella pyritään ensisijaisesti vaatimusten tarkentamiseen ja validointiin ennen kuin varsinainen kehitystyö aloitetaan. Seuraavalla kehityskierroksella tarkoituksena on kehittää tuotteen muotoilua edelleen. Spiraalin uloimmalla kehityskierroksella laaditaan testaus ja integrointisuunnitelma ja viimeistellään kehitys testaus mukaan lukien tuotteen asennusta varten. (Harju 2021, 18.)

2.3.4 Iteratiiviset ja inkrementaaliset mallit

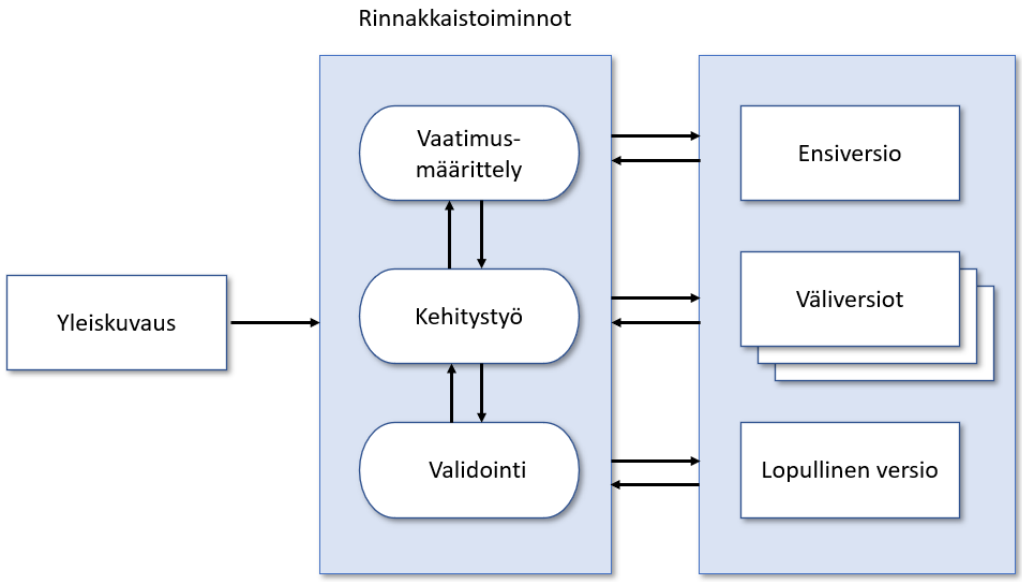
Iteratiivisessa ohjelmistokehityksessä usean yhteisesti sovitun mittaisen iteraation, eli kehityskierroksen aikana määritellään, suunnitellaan, toteutetaan ja testataan joukko ominaisuuksia. Iteraatiot voivat sisältää muutoksia projektin tehtäväkokonaisuuteen tai muutoksia aiemmissa iteraatioissa toteutettuihin ominaisuuksiin. Jokaisen iteraation tarkoituksena on tuottaa toimiva ohjelmisto, jossa kokonaisuuteen lisätään ominaisuuksia vähitellen, kunnes, kunnes viimeinenkin ohjelmiston osa on toimitettu tai toteutus lopetetaan. (FiSTB 2018, 25.)

Iteratiivisista kehitysmalleista tunnetuimpia ovat Rational Unified Proscess (RUP) -malli sekä spiraalimalli. RUP-mallissa iteraatiokierrokset ovat yleensä suhteellisen pitkiä, jopa kaksi tai kolme kuukautta, ja ominaisuuksien tuoteversiot ovat suuria, kaksi tai kolme toisiinsa liittyvien

ominaisuuksien ryhmää (FiSTB 2018, 25). Tästä syystä RUP-mallia pidetään myös itsenäisenä, laajaa lopputuotetta kohti etenevänä miniprojektina. (Kasurinen 2013.)

Inkrementaalisilla kehitysmalleilla tarkoitetaan niitä ohjelmistokehityksen tapoja, joissa tuotteen määrittely, kehitys ja validaatio kulkevat rinnakkain sen sijaan, että ne olisivat toisistaan erillisiä toimintoja, kuten vesiputousmallissa. Jo ensimmäisestä kehitetystä sovellusversiosta kerätään käyttäjiltä palautetta ja heiltä saadun palautteen perusteella sovellusta kehitetään edelleen. Versioita ohjelmistosta voi tulla useita, kunnes lopulta päädytään lopulliseen versioon. Inkrementaalinen lähestymistapa muistuttaa tavallista ongelmanratkaisuprosessia, jossa vaihe vaiheelta ja askel askeleelta edetään kohti ratkaisua. Vesiputous-malliin verrattuna inkrementaalisten mallien etuina ovat kustannustehokkuus, jatkuva asiakaspalautteen saaminen sekä mahdollisuus nopeaan käyttöönottoon. (Sommerville 2016, 49-51.) Inkrementaalisissa kehitysmalleissa ohjelmistokehitys on nopeaa ja syklistä. Testausta tässä prosessimallissa tehdään koko ohjelmiston kehityksen ajan. (Lappalainen 2019, 10.)

Inkrementaalisessa mallissa ohjelmistokehityksen eri vaiheet tehdään osissa, mikä tarkoittaa sitä, että ohjelmiston ominaisuudet laajenevat pala kerrallaan. Näiden palasten koko vaihtelee ja palaset voivat olla niinkin pieniä kuin käyttöliittymän ikkunan yksittäinen muutos tai uusi kyselyn vastausvaihtoehto ja toisaalta niinkin suuria kuin uusi kokonainen järjestelmä. (FiSTB 2018, 25.) Kuvassa 4 havainnollistetaan, miten eri ohjelmistokehityksen vaiheet kulkevat pikemminkin rinnakkain, ei peräkkäin.



Kuva 4. Inkrementaalinen ohjelmistokehitysmalli (mukaillen Sommervilleä 2016, 50)

2.4 Ketterät ohjelmistokehitysmenetelmät

Ketterä kehitys -ilmaisua käytetään kuvaamaan useita erilaisia ohjelmistokehitysmenetelmiä, joille yhteistä on iteratiivisuus ja inkrementaalisuus (Lehtonen 2014, 2). Ketterässä ohjelmistokehityksessä koko kehitystiimi kantaa vastuuta lopputuotoksen laadusta. Tällöin testaajat, kehittäjät ja liiketoiminnan edustajat työskentelevät läheisesti jokaisen kehitysprosessin vaiheessa varmistaen, että kaikilla on yhteinen käsitys testausstrategiasta ja testausautomaation lähestymistavoista, ja että liiketoiminta saa riittävästi tukea hyväksymistestien kirjoittamiseen. (FiSTBa 2015, 10.) Yleisvastuu testausprosessista ja testaustehtävien menestyksekkäästä johtamisesta kuuluu kuitenkin alan standardien mukaan testauspäällikön tehtäviin. Roolia voi hoitaa ammattimainen testauspäällikkö, projektipäällikkö, kehityspäällikkö tai laatupäällikkö. (FiSTB 2018, 58.)

Ketterissä elinkaarimalleissa testitasot usein limittyvät, koska koodiin tai vaatimusmäärittelyihin voi tulla muutoksia missä vaiheessa tahansa. Tyypillisesti kehittäjien suorittamien yksikkötestien lisäksi ketterissä elinkaarimalleissa tehdään ominaisuuksien todentamistestausta, jossa käyttäjätarinaa testataan hyväksymiskriteerejä vasten sekä ominaisuuksien kelpuutustestausta, jossa kehitystiimi usein liiketoiminnan edustajien kanssa yhteistyössä varmistavat, että ominaisuus on valmis käyttöönottoon. Näiden rinnalla tehdään jatkuvaa, usein automatisoitua regressiotestausta sen varmistamiseksi, että mikään aiemmin kehitetyistä ominaisuuksista ei ole rikkoutunut. (FiSTB 2015a, 21.) Finnish Software Testing Boardin Ketterän testaajan sertifikaattisisällön (2015a, 9) mukaan testaussuunnittelu alkaa kaikilla testaustasoilla välittömästi testausprosessin alkaessa ja jatkuu testauksen päätöstehtävien valmistumiseen saakka.

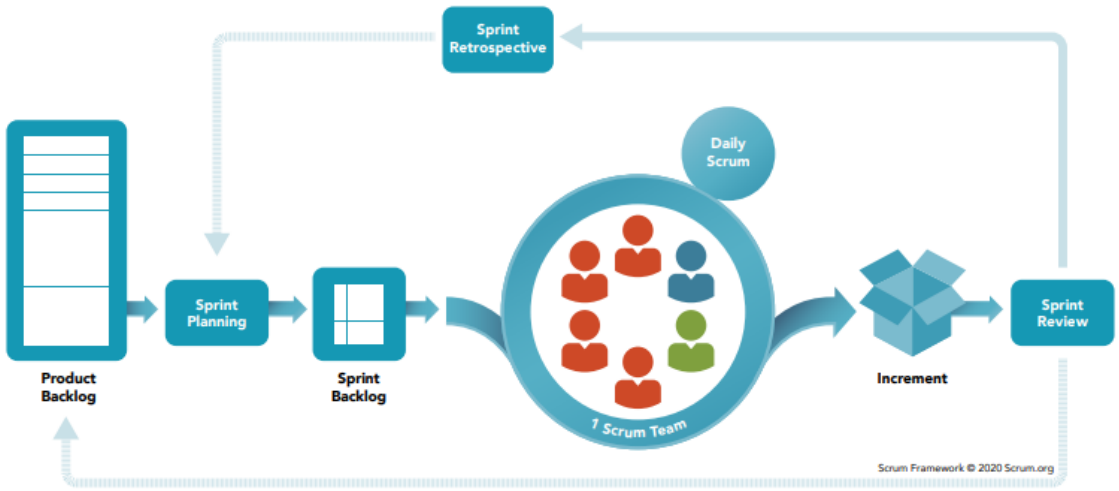
Ketteriin ohjelmistokehitysmalleihin kuuluu useita hieman toisistaan poikkeavia menetelmiä (Lappalainen 2019, 10). Näistä esimerkkejä ovat Scrum- ja Kanban-mallit, Extreme Programming (XP) sekä Extreme Programming (XP) -malliin pohjautuvat testivetoinen ja hyväksymistestivetoinen kehitys. (FiSTB 2015a.) Kyseisissä malleissa otetaan vaihtelevasti kantaa siihen, kuinka testausta tulee toteuttaa.

2.4.1 Scrum

Scrum on ketterän johtamisen viitekehys, joka sisältää lukuisia käytäntöjä ja välineitä ohjelmistokehitystiimin työn ohjaamiseen. Scrum jakaa projektin iteraatioihin (sprintti), joiden aikana kehitystyötä tehdään kehitystiimeissä. Jokaisen sprintin tavoitteena on tuottaa julkaistava tuoteversio (inkrementti). Tuotteen kehitysjonoa, eli Product Backlogia hallinnoi tuoteomistaja, joka myös kehittää backlogia jokaisessa sprintissä. Sprintin sisäistä kehitysjonoa (Sprint Backlog) puolestaan hallinnoi kehitystiimi, joka valitsee kussakin sprintissä toteutettavat tehtävät. (FiSTB 2015, 11-12.)

Scrum-menetelmään liitetään usein myös valmiin määritelmät eli Definition of Ready ja Definition of Done. Definition of Ready määrittää sen, milloin kukin yksittäinen kehitystehtävä on riittävästi määritelty, jotta se voidaan viedä ohjelmistokehitystiimin kehitettäväksi. Definition of Done puolestaan kertoo sen, milloin tällaisen kehitystehtävän voidaan katsota olevan julkaisuvalmis. Scrumin peruspilareihin kuuluvat läpinäkyvyys (transparency), mukauttaminen (adaptation) ja tarkastelu (inspection), joita pyritään edistämään päivittäispalavereilla, jalostamalla kehitysjonoa tarvittaessa, katselmoimalla julkaisuvalmiita inkrementtejä sekä kehitysprosessin kehittämiseen tähtäävillä kulunutta sprinttiä taaksepäin katsovilla retrospektiivisillä palavereilla. (Schwarber, Sutherland 2020.)

Vaikka testaus onkin usein osa ketterän kehitystiimin työskentelyä, malli ei erikseen ota kantaa siihen, miten testausta tulisi toteuttaa esim. Scrum-projektissa. Se ei myöskään ota kantaa muihin ohjelmistoprojektin käyttämiin kehitystekniikoihin. (FiSTB 2015, 12.) Kuvassa 5 havainnollistetaan Scrum-viitekehyksen mukaisen ohjelmistokehityksen elinkaari. Mallissa kaikki järjestelmän kehitettävät ominaisuudet on määritelty Product Backlogille, eli tuotteen kehitysjonoon. Kehitystiimi kokoontuu Sprint Planning -tilaisuuteen keskustelemaan siitä, mitä kehitystehtävistä valitaan sprintin aikana toteutettavaksi ja arvioidaan kunkin kehitystehtävän työmäärä sekä kuinka paljon kehitystiimi pystyy sprintin aikana tuottamaan valmiita inkrementtejä. Usein ennen Sprint Planningia järjestetään lisäksi kehitysjonoa tarkastelevia ja määrittelyitä tarkentavia Backlog Refinement-tilaisuuksia, joka auttaa kehitystiimiä valitsemaan kehitystehtäviä tehtävälistalleen. Kehitystiimi ottaa kehitykseen valituista tehtävistä tehtäviä sitä mukaa, kun edelliset tehtävät valmistuvat. Katselmointitilaisuus, eli Sprint Review järjestetään sprintin päätteeksi. Se antaa tuoteomistajalle ja muille sidosryhmille mahdollisuuden antaa palautetta inkrementistä. Näitä iteraatioita voidaan toistaa useita kertoja ennen kuin tuote, esim. uusi ohjelmisto on valmis julkaistavaksi tuotantokäyttöön. (Scrum.org 2021.)

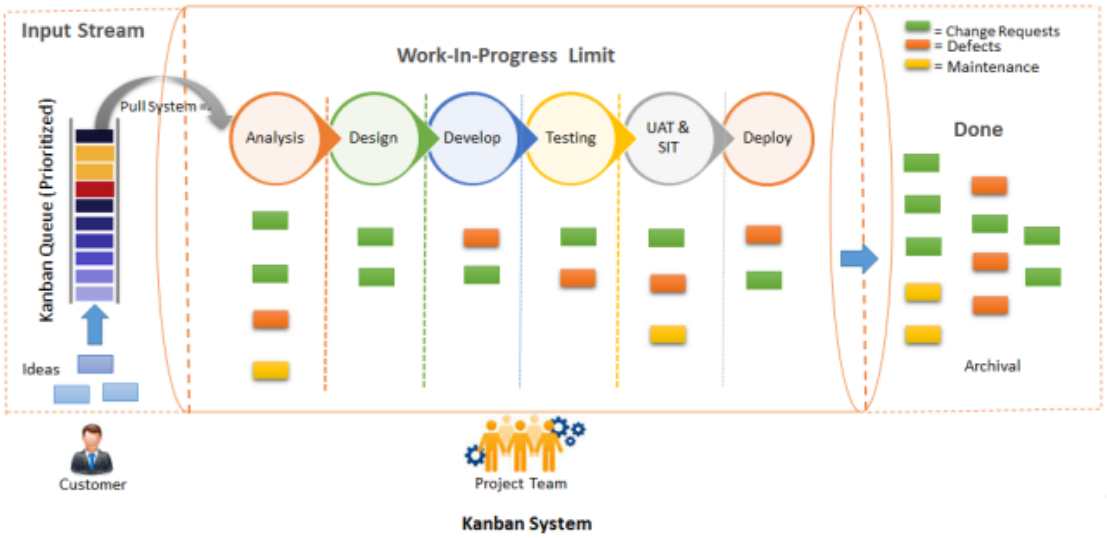


Kuva 5. Scrum-viitekehys (Scrum.org 2021)

2.4.2 Kanban

Kanban-viitekehyksen yleisenä tavoitteena on optimoida kehitysjono tuotettavan arvon perusteella. Sen perusperiaatteita ovat työnkulun visualisointi, käynnissä olevan työn määrän kontrollointi, työnkulun hallinta ja mittaaminen sekä käytänteiden tekeminen näkyviksi ja jatkuvaan kehittämiseen. Tämä hallinnoitava arvoketju kuvataan sähköisellä tai tarralapuilla seinälle toteutettavalla Kanban-taululla (Koski 2012, 29-30).

Kuvassa 6 taulun sarakkeilla kuvataan kehitystehtävien työnkulkua vasemmalta oikealle edeten työstövalmiista tehtävästä valmiiseen. Kehitystehtäviä ovat esimerkiksi kehitys ja testaus. Yksittäisiä kehitystehtäviä siirretään sarakkeesta seuraavaan tehtävien edistymisen mukaan. Kanban-mallissa rinnakkaisten aktiivisten tehtävien määrä on tiukasti rajattu, mikä tarkoittaa sitä, että yhdessä sarakkeessa/työvaiheessa oleville tehtäville on asetettava maksimiarvot. Kanban-mallissa tehtävävirta pyritään optimoimaan minimoimalla tehtävien läpi-menoaika. Uusia tehtäviä lisätään Kanban-taululle heti, kun taululla on tilaa edellisten tehtävien valmistuttua, joten aikarajoittaminen synkronointimenetelmänä on valinnaista toisin kuin Scrum-mallissa. (FiSTB 2015.) Kanban-mallissa testauksen ja hyväksymistestauksen tehtävät seuraavat kehitystyötä ja vasta kun inkrementti on käynyt läpi kaikki vaiheet, se voidaan merkitä valmiiksi (Koski 2012, 23). Myös Kanban-mallissa on käytössä Definition of Done, joka määrittää sen, milloin inkrementti on valmis. Toisinaan hyväksymistestausta ei kuitenkaan sisällytetä Definition of Doneen, jolloin asiakkaan hyväksymistestaus tehdään vasta inkrementin valmistumisen jälkeen.

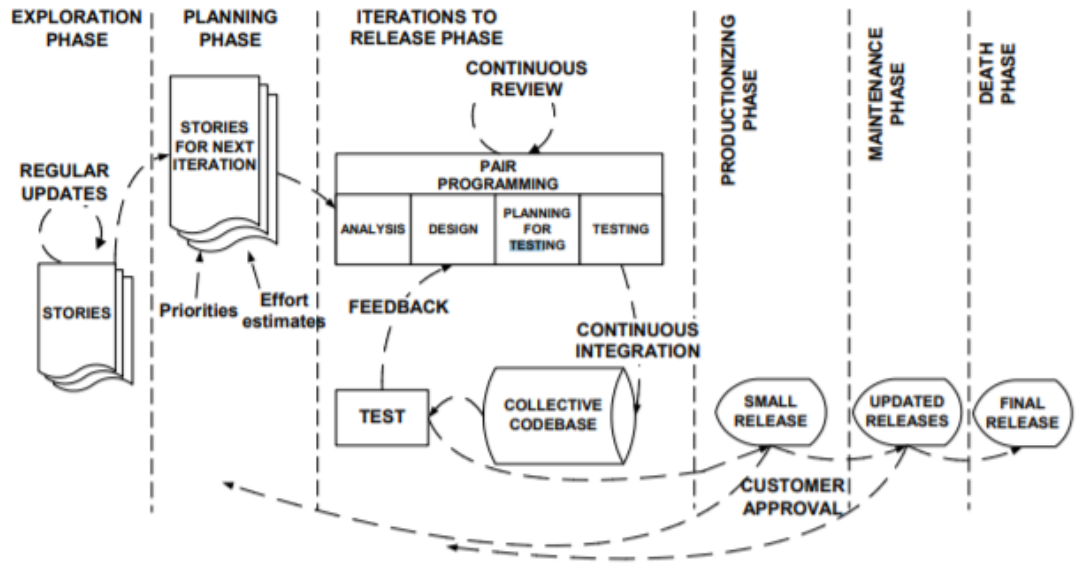


Kuva 6. Kanban-viitekehys (Agilegnostic)

2.4.3 Extreme Programming

Extreme Programming- eli XP-mallin keskiössä on kehitystiimin tiivis yhteistyö ja kommunikaatio. Mallissa pyritään luomaan informatiivinen ja innostava, työskentelyä tukeva työtila. Mallille tyypillisiä piirteitä ovat pariohjelmointi sekä nopeatahtinen julkaisusykli. Ohjelmointia toteutetaan usein testilähtöisesti ja sitä suunnitellaan vaiheittain. (FiSTB 2015.)

Kuvassa 7 on havainnollistettu XP-mallin mukainen kehityssykli. Sen ensimmäisessä vaiheessa määritellään ensimmäiseen julkaistavaan versioon vaaditut toiminnot loppukäyttäjän tai asiakkaan näkökulmasta kirjoitettujen käyttötapausten (use case) avulla, joita XP:ssä kutsutaan tarinoiksi (story). Toimittaja katselmoi käyttötapaukset ja kun ne ovat riittävän yksityiskohtaisella tasolla, ohjelmoija suunnittelee ja kirjoittaa käyttötapaukselle vastaavan testitapauksen, minkä jälkeen vasta kirjoitetaan varsinainen ohjelmakoodi. Käyttötapaukseen tarvittavan koodin valmistuttua se integroidaan muuhun ohjelmakoodiin ja testataan automatisoidun testausjärjestelmän avulla. Näin siis ajetaan välittömästi kaikki testausjärjestelmän sisältämät testit. Jos testit läpäistään onnistuneesti, ohjelmointia jatketaan seuraavalla käyttötapauksella. Muussa tapauksessa virhe korjataan ja vasta kun virheitä ei enää tule voidaan edetä seuraavaan käyttötapaukseen. Koodin moduulistestauksen jälkeen asiakkaan tehtävänä on vielä kirjoittaa käyttötapausta vastaava toiminnallinen testi, jolla varmistetaan, että ohjelmistolle asetetut vaatimukset ovat täyttyneet. Kun kaikki versioon tarkoitetut käyttötapaukset on testattu viimeisenkin vaiheen jälkeen onnistuneesti, ohjelmisto voidaan asentaa asiakkaalle ja aloittaa seuraava kehityskierros. (Lindberg 2003, 17-20.)



Kuva 7. XP-elinkaarimalli (Abrahamsson ym., 2016)

2.4.4 Testivetoinen kehitys ja hyväksymistestivetoinen kehitys

Test Driven Development (TDD), eli testivetoinen kehitys on saavuttanut suosionsa XP:n eli Extreme Programmingin kautta (FiSTBb 2015). Testivetoinen kehitys ei nimestään huolimatta ole testausmenetelmä. Se on ohjelmistojen kehitysmenetelmä, jossa ohjelmoija kirjoittaa sekä testikuvausta että ohjelmakoodia rinnakkain. Esimerkiksi ensin voidaan kirjoittaa yksikkötesti ja vasta tämän jälkeen testin läpäisevä koodi. Kun uusi toiminnallisuus halutaan lisätä koodiin, kirjoitetaan ensin testi, joka luonnollisesti antaa virheen ohjelmaa ajettaessa. Tämän jälkeen kirjoitetaan itse koodi toiminnallisuudelle ja testataan se uudestaan. (Perälä 2018, 13.)

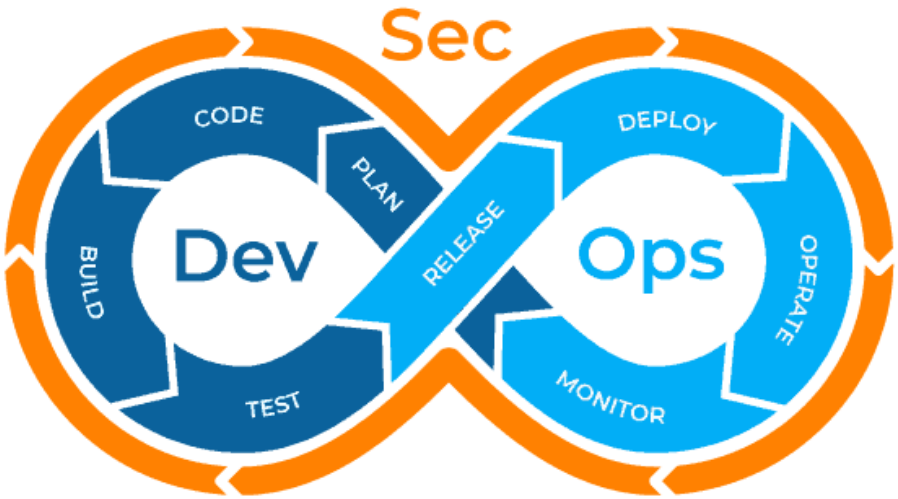
Tällainen tavallaan käänteinen kehitysmenetelmä pakottaa ohjelmoijan miettimään ohjelmoitavaa asiaa monelta eri kannalta. On käytännössä todettu, että testivetoisella menetelmällä koodatussa ohjelmassa on vähemmän virheitä ja valmis koodi on luettavampaa. Testivetoisen kehityksen oletettu paremmuus piilee nopeissa sykleissä, jolloin ohjelmiston virheet voidaan todeta nopeasti ja työskentely on mukautuvaa ja sujuvaa. Testivetoisen menetelmän edut ovat kuitenkin kiistanalaisia. Menetelmän oppiminen on vaativaa ja hidasta, eikä ohjelmoijilla ole usein riittävästi testaajan taitoja. Myös tuottavuus voi kärsiä, koska testivetoinen kehitys on hitaampi tapa koodata. (Lappalainen 2019, 11.)

Kuten testiohjatussa kehityksessä, myös ns. hyväksymistestiohjatussa kehityksessä luodaan testejä ennen koodin kirjoittamista. Kyseisten testien tulee edustaa ohjelmiston odotettua käyttäytymistä. Hyväksymistestiohjatussa kehityksessä tiimi luo yhden tai useamman hyväksymistason testin kullekin ominaisuudelle ennen kuin ominaisuutta aletaan työstää ohjelmoimalla. Tyypillisesti tällaiset testit määritellään käyttäjätarinoiden luomisen aikana ja testeistä keskustellaan tilaisuuksissa, joissa tiimi työskentelee läheisesti liiketoiminnan edustajien kanssa tuotteen kehitysjonolla olevan käyttäjätarinan ymmärtämiseksi. (Hendrickson, 2008.)

2.4.5 DevOps ja DevSecOps

DevOps -lyhenne tulee sanoista Development (Dev) ja Operations (Ops). DevOps-termillä tarkoitetaan ketterää toimintamallia, jossa korostuu ohjelmistokehittäjien ja IT-asiantuntijoiden kommunikaatio ja yhteistyö sen sijaan, että se olisi tietty ohjelmistokehitys- tai projektinhallintatyökalu. Mallin kantavana ideana on ohjelmiston testauksen ja julkaisun automatisointi siten, että ohjelmiston matka kehittäjiltä loppukäyttäjille olisi mahdollisimman suoraviivainen. (Juvonen 2018.) DevOpsissa kehittäjillä tarkoitetaan järjestelmäkehittäjiä, testaajia, laadunvalvojia sekä ylläpitäjiä, kun taas operoijilla tarkoitetaan sellaisia resursseja, jotka vastaavat järjestelmän, tuotteen tai palvelun tuotannosta. (Heikkinen 2018, 27).

Tämän viitekehyksen ytimessä on ohjelmistotuotannon ja -kehityksen jatkuvuuden turvaaminen kiivaassa kehityssyklissä, kun ohjelmistokoodin jatkuva integrointi vie tuotantoon jopa satoja uusia järjestelmäversioita päivässä. DevOps-kehittämisprosessiin voidaan integroida myös tietoturva (Sec), sillä nopeassa kehityssyklissä on tärkeää varmistaa ohjelmistojen tietoturva yhtä nopealla tahdilla. (Koskinen 2020, 8-11.) Tällöin viitekehystä kutsutaan DevSecOps:ksi. Kuvassa 8 havainnollistetaan tämän mallin eri vaiheita.



Kuva 8. DevSecOps-viitekehys (Plutora 2022)

DevSecOps-mallin vaiheita ovat suunnittelu, kehitys, rakentaminen, testaus, julkaisu, toimitus, käyttöönotto, käyttö ja valvonta. Kuten muissakin ketterissä malleissa, nämä vaiheet limittyvät keskenään ja eri vaiheita toteutetaan osin päällekkäin toistensa kanssa. DevSecOps-mallissa testaukseen ja turvallisuuteen liittyviä tehtäviä voidaan siirtää aikaiseen ohjelmistokehityksen vaiheeseen automatisoitujen yksikkö-, integraatio- ja turvallisuustestien ansiosta. Suunnitteluvaiheessa DevOps-mallissa määrittelydokumentaatio toimii pohjana järjestelmän muotoilulle (design), jonka tuotoksena on järjestelmäarkkitehtuurin ja palvelun toiminnallisen muotoilun ohella testaussuunnitelma ja testauksen työkalut. Testaussuunnitelman pohjalta toteutettaviin kehitysvaiheen tehtäviin kuuluvat puolestaan yksityiskohtaisten testiproseduurien, testidatan, testiskriptien sekä testausskenaarioiden kehittäminen valitulla testauksen työkalulla. Testaustyökalut tukevat jatkuvaa testausta ohjelmistokehityksen koko elinkaaren ajan. (Department of Defence 2019, 33, 40.)

DevOps on käsitteenä hyvin moniulotteinen, mistä syystä sen sisältö jää usein epäselväksi. Koska toimintamallia on mahdollista toteuttaa monella eri tavalla, on myös sen käyttöönotto organisaatioissa vaikeaa. (Heikkinen 2018, 8.)

2.5 Dokumentaatio ketterässä ohjelmistokehityksessä

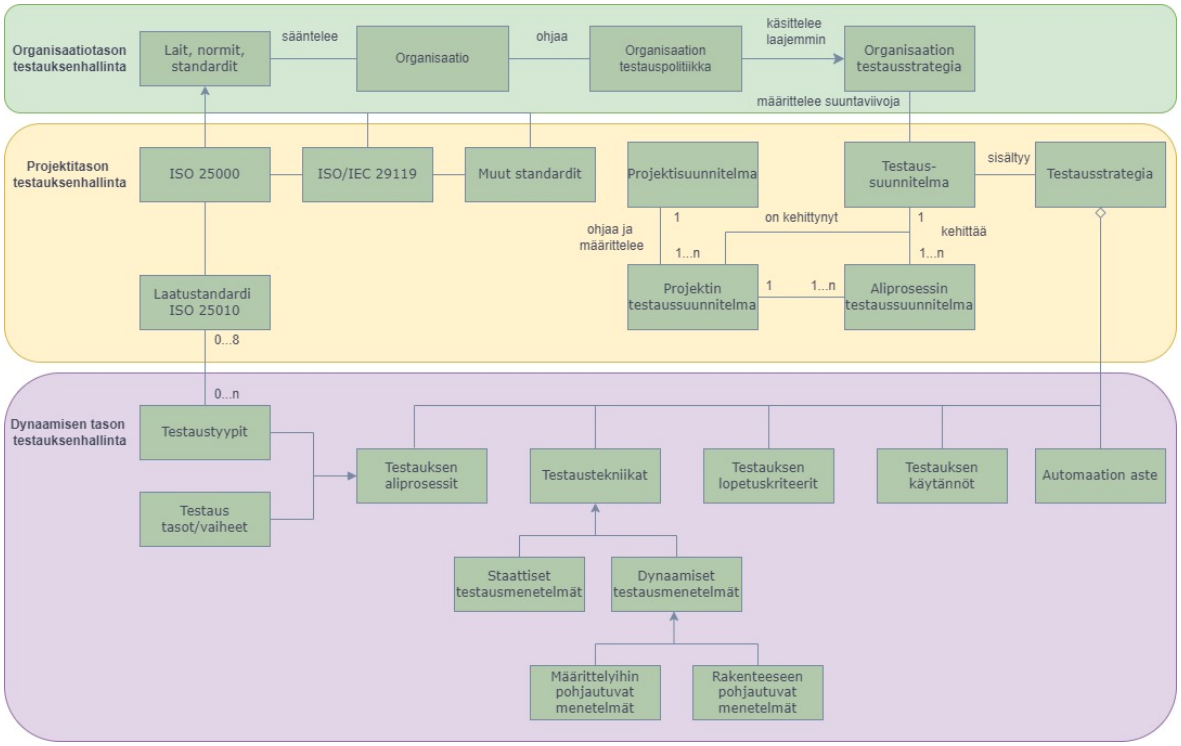
Dokumentaation merkitys ketterässä ohjelmistokehityksessä on vähentynyt verrattuna pe-rinteisiin järjestelmäkehitysmenetelmiin, koska kattavan dokumentaation tuottamisen uskotaan vievän aikaa järjestelmän kehittämiseltä ja testaamiselta. Dokumentaatio korvataan kehittäjien, ja heidän sidosryhmiensä välisellä, tiheällä vuorovaikutuksella. Vähäiseen dokumentaatioon voi kuitenkin liittyä useita haasteita. Usein haasteet liittyvät vaatimusmäärittelyn ja dokumentaation yhteistoimivuuteen. Jos esimerkiksi vaatimusmäärittely ei ole riittävän kattava tai se on liian monitulkintainen, voi kehittäjillä on eri näkemyksiä vaatimuksista. Tästä voi seurata ongelmia esimerkiksi ohjelmakoodiin tehtävistä muutoksista, jos muutosta ei ole dokumentoitu tai perusteltu miksi muutos on toteutettu tietyllä tavalla. Näin kehitystii-min sisällä voi herätä epätietoisuutta ja ristiriitoja vaatimusten näkökulmasta ja tärkeydestä. Dokumentaatioon liittyviä haasteita voidaan pyrkiä ehkäisemään lisäämällä tehtävälistan läpinäkyvyyttä ja kiinnittämällä huomiota tehtävälistan ajantasaisuuteen projektin osallisille. Jos esimerkiksi vaatimuksiin tehdään muutoksia niin, että kaikki eivät ole paikalla päättämässä muutoksesta, muutos tulisi kirjata välittömästi koko tiimin näkyville tehtävälistaan. (Laukkarinen 2019, 9, 19.)

Vaikka ketterän projektin yleinen käytäntö onkin välttää tuottamasta suurta määrää dokumentaatiota tehokkuuden kasvattamiseksi, tulee kuitenkin huomioida tasapaino dokumentoinnin vähentämisen ja riittävän dokumentoinnin tuottamisen välillä. Dokumentointi liiketoimintaan, testaukseen, toteutukseen ja ylläpitoon liittyvien tehtävien tukemiseksi on edelleen tärkeää, minkä vuoksi suunnittelussa onkin huomioitava se, mitä työn tuloksia tarvitaan sekä taso, jolla tulokset dokumentoidaan. Onkin huomattava, että on syytä välttää vain sellaista dokumentaatiota, joka ei tuota lisäarvoa asiakkaalle. Tiukasti säädellyissä, turvallisuuskriittisissä, hajautetuissa, tai hyvin monimutkaisissa projekteissa tai tuotteissa tarvitaan lisäksi formaalimpaa dokumentointia, jossa tiimit muuntavat esimerkiksi käyttäjätarinoita ja hyväksymiskriteereitä muodollisiksi vaatimusmäärittelyiksi. Auditoijia, säädöksiä ja muita vaatimuksia vakuuttamaan voidaan laatia myös vertikaalisia ja horisontaalisia jäljitettävyysraportteja. (FiSTB 2015, 20-21.)

2.6 Ohjelmistoprojektin testauksenhallinta

Suomalaiseksi kansalliseksi standardiksi vahvistetun kansainvälisen standardin ISO/IEC/IEEE 29119-1:2013 mukaan kaikkea ohjelmistotestausta sääntelee viime kädessä lait, normit ja standardit. Kasurisen (2013, 81) mukaan sellaisessa organisaatiossa, joka noudattaa ISO/IEC 29119 -ohjelmistotestausstandardin periaatteita, on testausdokumenteista käytössä testauspolitiikka, testausstrategia, testisuunnitelma ja testausraportit.

Kuvassa 9 on havainnollistettu testauksenhallinnan eri tasoja. Testauksenhallintaa kansainvälisten standardien mukaan tulisi harjoittaa niin organisaatiotasolla, projektitasolla kuin dynaamisen testauksen prosessienkin tasolla. Organisaatiotasolla laadittu testauspolitiikka ilmaisee organisaation johdon odotuksia ja lähestymistapoja ohjelmistojen testaukseen. Se myös tarjoaa suuntaviivoja organisaation testausstrategian toteuttamiseksi ja testausprosessien kehittämiseksi. Organisaation testausstrategia puolestaan linjaa testauksenhallinnan prosessien sekä dynaamisen testauksen prosessien vaatimuksia ja rajoituksia ja määrittelee, millainen organisaation testausprosessin tulee olla ja miten testaus tulee toteuttaa. Projektitasolla testauksenhallinnan prosesseilla ohjataan organisaation käytäntöjä testaussuunnitelman laatimiseksi eri dynaamisen testauksen tasoilla. Eri tasojen alitestausprosesseja tulisi siten hallinnoida ja toteuttaa samalla tavalla kuin testausprosessia yleisellä tasolla. Testauksenhallinnan prosessien pitäisi olla myös toteutettavissa kaikilla testauksen tasoilla. (IEEE 2021.)



Kuva 9. Testauksenhallinnan käsitekartta (mukaillen IEEE 2013)

2.7 Testaussuunnitelma testauksen tuotoksena

Testauksen tuotoksia syntyy kaikissa ohjelmistoprojektin vaiheissa. Tällaisilla tuotoksilla tarkoitetaan automatisoitujen testien lisäksi esimerkiksi erilaisia dokumentteja, kuten testausstrategioita ja -suunnitelmia, laaturiskiluetteloita, manuaalisia testejä, vikaraportteja ja testilokeja. Koska ketterässä projektissa tapahtuu tyypillisesti muutoksia koko projektin ajan, suositaan myös testauksen osalta hyvin kevyttä työn tulosten dokumentointia. (FiSTB 2015, 20-21).

Kehitys- ja ylläpitoprojektien testaustoimenpiteitä linjataan testaussuunnitelmassa. Tässä dokumentissa määriteltäviä asioita ovat testauksen työkalut, testausmenetelmät ja testauksessa tarvittavat resurssit. Testaussuunnitelmassa otetaan tyypillisesti kantaa myös testitapausten suunnitteluun ja sopivien testitapausten valintaan. Dokumentissa voidaan viitata organisaation testauspolitiikkaan sekä siihen, mitä testausstrategiaa testauksessa käytetään. Lisäksi suunnitelmassa kuvataan ohjelmistokehityksen elinkaarimaali ja menetelmät sen toteuttamiseksi. Testaussuunnitelman tulee sisältää lisäksi oleelliset riskit ja testauksen tavoitteet. Sen tulee ottaa myös kantaa testauksen laajuuteen, ja siinä tulee huomioida mahdolliset rajoitteet arvioiden myös testauksen kriittisimpiä osa-alueita. Testaussuunnitelmaa on päivitettävä läpi koko tuotteen elinkaaren, sillä mitä pidemmälle projekti edistyy, sitä enemmän on saatavilla tietoa ja testaussuunnitelmaan voidaan sisällyttää aina vain tarkempia yksityiskohtia. (FiSTB 2018.)

Riskien tunnistamista edesauttaa testaustehtävistä saatava palaute, jonka tulisi olla pohjana testauksen suunnittelulle. Suunnittelu voidaan dokumentoida kokonaistestaussuunnitelmaan ja erillisiin tasokohtaisiin testaussuunnitelmiin, kuten järjestelmätestaus- ja hyväksymistestaussuunnitelmat, tai erillisiin testaustyyppikohtaisiin suunnitelmiin, kuten käytettävyystestaus- ja suorituskykytestaussuunnitelmiin. (Kasurinen 2013.) FiSTB-sertifikaattisisällön mukaan kokonaistestaussuunnitelmalla tarkoitetaan testauksen pääsuunnitelmaa, eli ns. Master test plania, joka kattaa useampia testitasoja. (FiSTB 2015.)

Ketterissä ohjelmistoprojekteissa testaussuunnitelma on usein lyhyt, yhden tai kahden sivun mittainen dokumentti, ja on osa testausstrategiaa. Testausstrategia sisältää yleensä kuvauksen siitä, miten ohjelmistoa on tarkoitus testata, jokaiselle sprintille räätälöidyn testaussuunnitelman, yksityiskohtaisesti määritellyt testitapaukset, ideoita tutkivan testauksen toteuttamiseen, testauksen lokitiedot, joissa on erotettu odotettu tulos sekä regressiotestauksen tarkistuslistan. Vaihtoehtoisesti testaussuunnitelman dokumentointiin ketterässä ohjelmistokehityksessä voidaan käyttää pelkkiä tarkistuslistoja ja tutkivaa testausta. (IEEE 2021).

Taulukossa 1 on vertailtu, kuinka testaussuunnittelu on otettu huomioon erilaisissa ketterän ohjelmistokehityksen viitekehyksissä ja malleissa. Siinä kerrotaan myös, mikä työvaihe laukaisee testaussuunnittelun kussakin ohjelmistokehitysmallissa.

Taulukko 1. Testaussuunnittelu ketterän ohjelmistokehityksen elinkaarimalleissa

	Testaussuunnitelma	Syöte	Laatija/-t	Tuotos
Scrum	Ei	-	-	
Kanban	Kyllä	Työnkulun ja sen sisältämien tehtävien määrittely	Kehitystiimi	Ylätason kehitystehtävä, esim. ”testaussuunnitelma”
XP	Kyllä	Analyysi ja design	Kehitystiimi	Testitapaukset
TDD ja ATDD	Kyllä	Käyttäjätarinoiden määrittely/suunnittelupalaveri	Kehitystiimi	Käyttäjätarina, joka sisältää testitapauksen/testiskenaarion
DevOps ja DevSec Ops	Kyllä	Järjestelmän designin suunnittelu määrittelydokumenttien pohjalta	Kehitystiimi	Testiproseduuri, skenaariot, testiskriptit, testiaineisto

3 Kyselytutkimuksen toteutus

Opinnäytetyöni empiirisen osan tavoitteena oli tutkia, kuinka ketterillä menetelmillä ohjatuissa ohjelmistoprojekteissa työskentelevät ammattilaiset suhtautuvat testaussuunnitelman laatimiseen. Tutkimus toteutettiin puolistrukturoidulla kysymyslomakkeella, jonka teemat ja kysymykset oli laadittu Tietoperusta-luvussa käsiteltyjen aihealueiden pohjalta. Kysely toteutettiin Google Formsilla ja vastausten muodostama aineisto analysoitiin Google Formsin kuvaajien ja kysymyskohtaisen datan avulla.

Kyselyn ensimmäiset viisi kysymystä keskittyivät taustoittamaan vastaajan roolia ketterän ohjelmistokehityksen parissa sekä sitä, millaisessa organisaatiossa vastaaja työskentelee. Olennaisinta oli, että taustatietojen avulla voitiin pyrkiä kartoittamaan sitä, mikä on vastaajan suhde testaussuunnitteluun. Koska julkaisusykli vaihtelee organisaatioissa usein sen perusteella, mikä ohjelmistokehityksen elinkaarimalli on valittu, pyrin taustoittavilla kysymyksillä selvittämään lisäksi sitä, mitä ketterän ohjelmistokehityksen viitekehyksen periaatteita vastaajien organisaatioissa/yrityksissä noudatetaan.

Jätin tietoisesti tarkastelun ulkopuolelle sukupuolen mahdollisen vaikutuksen kyselyn tuloksiin, sillä tässä kontekstissa vastaajan sukupuolella ei ole merkitystä testaussuunnitelman hyödyntämisessä käytännön työssä. Taustatietona halusin kartoittaa kuitenkin vastaajien ikää, sillä halusin tutkia, onko testaussuunnitelman laatimiseen liittyen sukupolvien välisiä eroavaisuuksia. Tästä syystä ikäluokat on jaettu kolmeen kategoriaan X-, Y- ja Z-sukupolvien mukaan. Koska nykyisin työtä tehdään varsin paikkariippumattomasti monissa IT-alan yrityksissä, en kokenut mielekkääksi taustoittaa vastaajien maantieteellistä sijaintia, sillä opinnäytetyöni tarkoituksena ei ole tutkia testauksenhallinnan toteuttamisen mahdollisia maantieteellisiä eroavaisuuksia.

Opinnäytetyöni tietoperustassa laatimani taulukon mukaan kaikissa ketterän kehittämisen viitekehyksissä ja toimintamalleissa kehitystiimi on vastuussa testaussuunnitelman laatimisesta. Tästä syystä kyselyllä selvitettiin sitä, kuka organisaatioissa vastaa minkäkin tason testaussuunnitelman laatimisesta. Roolin kartoittamisella pyrin muodostamaan käsityksen siitä, mikä merkitys testaussuunnitelman laatimisella koettiin olevan ohjelmistokehitysprosessin tukena. Merkityksellisyyttä mittaavissa kysymyksissä on käytetty vastausvaihtoehtoina lineaarista asteikkoa, jossa korkein numeroarvo kuvaa vastaajan erittäin merkityksellistä ja pienin numeroarvo vähiten merkityksellistä kokemusta kysymyksessä esitettyä aihetta kohtaan. Vaikka en onnistunut nimeämään arvoja Google Formsissa näiden ääripäiden arvojen välille, muodostin vastausvaihtoehdot seuraavalla periaatteella:

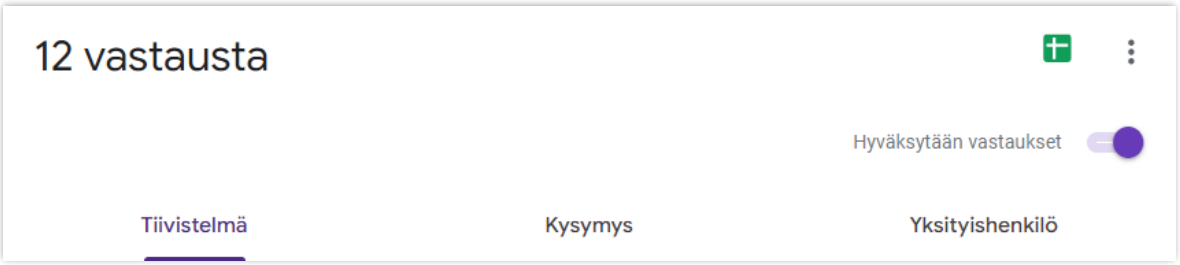
1. ei lainkaan merkityksellinen
2. vain vähän merkityksellinen
3. jonkin verran merkityksellinen
4. merkityksellinen
5. erittäin merkityksellinen

Kyselyn vastaajat pyrin hankkimaan omien verkostojeni kautta sosiaalisen median alustoilta. Ensisijaisena jakelukanavanani oli LinkedIn, jossa kaiken kaikkiaan 163 kontaktini joukossa on lukuisia laadunvarmistuksen ja -hallinnan asiantuntijoita ja ylipäänsä ketterissä ohjelmistoprojekteissa niin yksityisellä kuin julkisellakin sektorilla työskenteleviä henkilöitä. Jaoin kyselyn linkin myös Facebookin Monimuoto TIKOlaiset /Haaga-Helia -ryhmässä, jossa ryhmän yli 300 jäsenen joukossa on varmasti niin tuoreita alan asiantuntijoita, tulevaisuuden lupauksia kuin myös jo pidemmän aikaa sitten valmistuneita IT-alan ammattilaisia. Koska kyselylomake toteutettiin verkkokyselynä, laitoin julkaisuni yhteyteen myös pyynnön jakaa linkkiä edelleen henkilöille, jotka saattaisivat sopia tutkimukseni kohderyhmään. Toi-voin tällä tavoin saavani riittävästi vastauksia voidakseni tehdä luotettavasti johtopäätöksiä testaussuunnitelman merkityksestä laadunvarmistuksen ja -hallinnan työkaluna ketterässä ohjelmistoprojektissa.

Kysely julkaistiin alun perin 3.11.2022, ja sen oli tarkoitus olla auki kaksi viikkoa 16.11. mennessä vastauksia oli kuitenkin tullut vain seitsemän, joten julkaisin linkin vielä uudemman kerran LinkedIn:ssä ja laitoin tällä kertaa julkaisun julkiseksi. Tämän myötä kyselyyn tuli vielä viisi vastausta lisää. Kysely suljettiin 28.11.2022, ja sen lopullisten vastausten määrä oli 12.

4 Tulokset

Tarkastelin saamiani vastauksia määrällisestä näkökulmasta, mutta myös laadullisen sisällönanalyysin menetelmin siten, että vertailin kysymyskohtaisia vastauksia kunkin vastaajan muihin kysymyksiin antamien vastausten perusteella. Koska vastauksia oli sen verran vähän, tulosten vertailu oli helppoa Google Formsin omien toiminnallisuuksien ansiosta. Tiivistelmä-osiossa vastausten jakauma oli esitetty pylväs- ja sektoridiagrammein. Kysymys-osiossa puolestaan näytettiin tarkemmin kunkin vastauksen lukumäärät sekä Yksityishenkilö-osiossa pääsi tarkastelemaan kunkin yksittäisen vastaajan ilmoittamia tietoja yksityiskohtaisemmin. Tuloksista oli myös mahdollista luoda Google Sheets/Excel-dokumentti, jonka avulla yksityiskohtaiset tiedot olivat vieläkin helpommin luettavissa. (Kuva 10.)

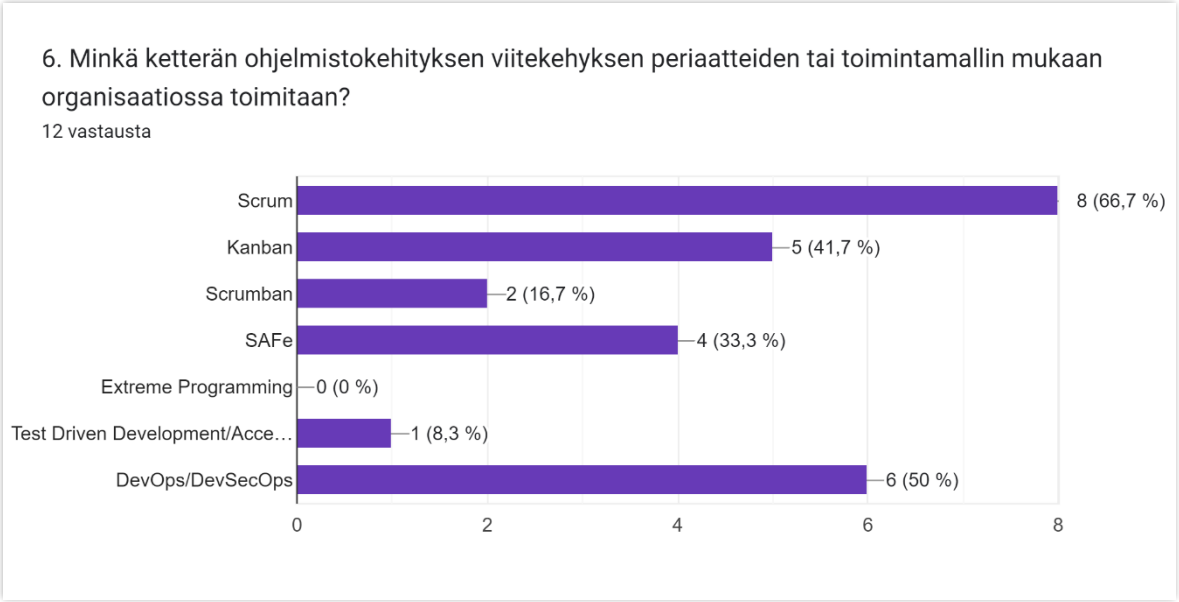


Kuva 10. Google Forms -datan analyysimenetelmät

Alhaisen vastaajamäärän vuoksi jätin raportoimatta kyselyyn vastanneiden taustatiedot vastaajien ja heidän edustamiensa yritysten anonymiteetin säilyttämiseksi. Yleisellä tasolla voitaneen kuitenkin todeta, että vastaajien joukossa oli useissa eri rooleissa, ketterässä ohjelmistokehityksessä toimivia henkilöitä, joiden tehtävänkuvaan suoraan tai välillisesti laadunvarmistus, -hallinta ja/tai testaus olennaisesti liittyy. Ikäkysymykseen en myöskään tässä analyysissa pureutunut syvällisemmin edellä mainitusta syystä, enkä siihen, olivatko vastaajat julkisen, yksityisen vai kolmannen sektorin suurista, keskisuurista tai pienistä organisaatioista.

4.1 Ohjelmistokehityksessä käytetyt ketterän ohjelmistokehityksen viitekehykset ja toimintamallit

Suurin osa vastaajista (66,7 %) kertoi organisaatiossaan toimittavan Scrum-viitekehyksen mukaisesti. Myös DevOps/DevSecOps oli valtaosassa näistä tapauksista mainittu. Yhdessä vastauksessa kerrottiin, että ohjelmistokehityksessä oli käytössä kaikki kyselyssä mainitut viitekehykset tai toimintamallit lukuun ottamatta Extreme Programmingia tai Test Driven/Acceptance Test Driven Developmentia. XP ja TDD:kin esiintyivät kaiken kaikkiaan vastauksissa selvänä vähemmistönä. (Kuva 11.)

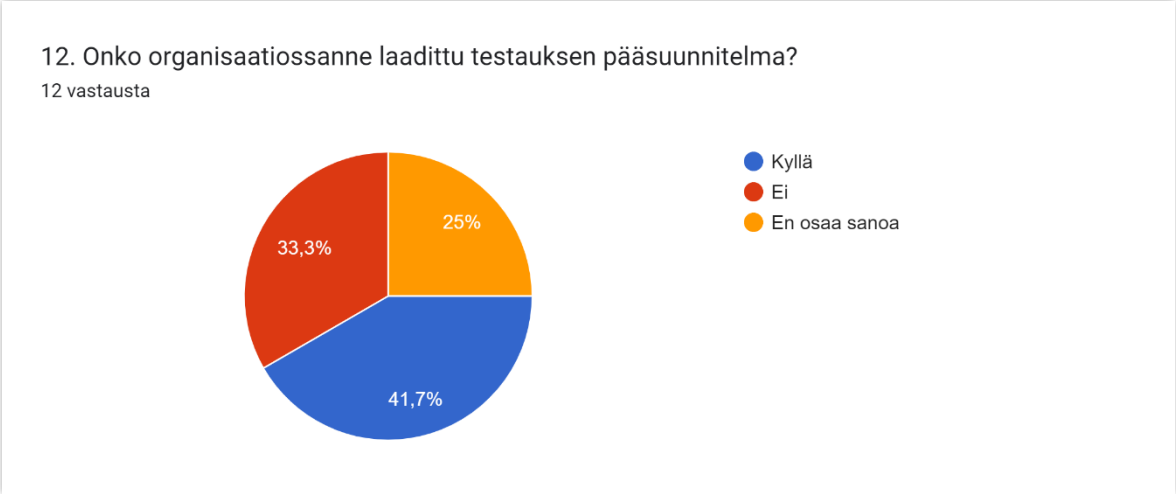


Kuva 11. Ohjelmistokehityksessä käytetyt ketterän ohjelmistokehityksen viitekehykset ja toimintamalli

Julkaisusykli vaihteli eri organisaatioiden välillä, mutta tyypillisin vastaus oli jatkuva julkaisu/julkaisu aina kun jotain valmistuu.

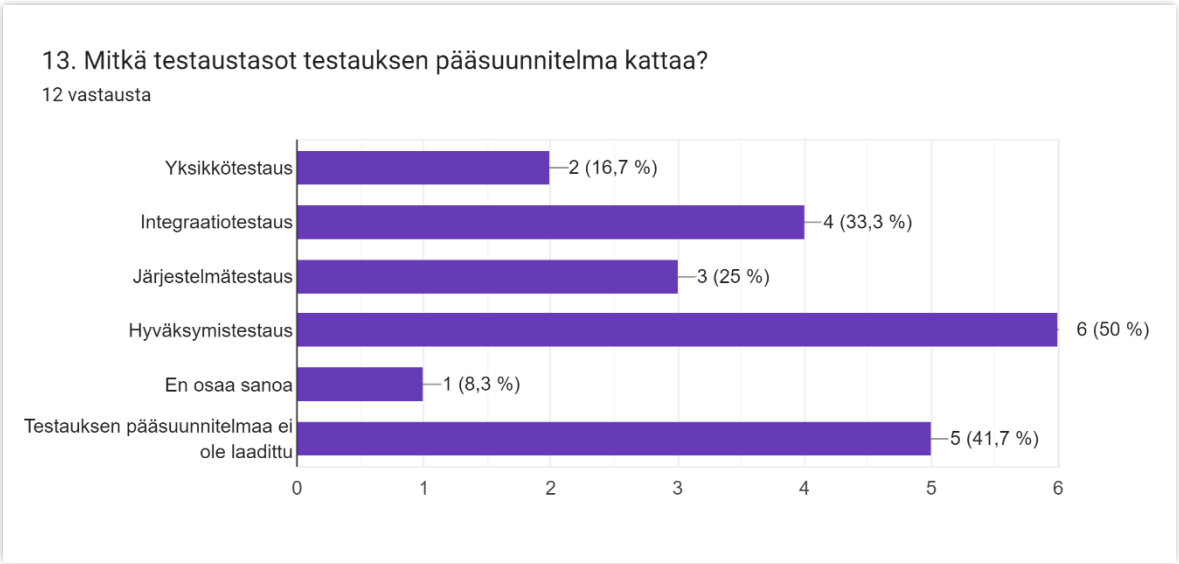
4.2 Testausstandardit ja ylätason dokumentaatio

Selvä vähemmistö kertoi noudattavansa organisaatiossaan ohjelmistotestauksen kansainvälisten standardien mukaisia testauskäytäntöjä. Vain kolmannes (33,3 %) vastaajista kertoi, että organisaatiossaan on laadittu testauspolitiikka. Hieman useampi, eli puolet vastaajista kertoi, että ylätason testausstrategia on kuitenkin laadittu. Kysyttäessä onko organisaatiossa laadittu testauksen pääsuunnitelma (ns. Master test plan) yli puolet vastaajista kertoi, että päätestaussuunnitelmaa ei ole laadittu tai ei osannut sanoa, onko sellaista laadittu (Kuva 12).



Kuva 12. Testauksen pääsuunnitelman laatiminen

Tarkennettaessa kysymystä sillä, mitä testaustasoja päätestaussuunnitelma kattaa, viisi vastaajaa kertoi, että testauksen pääsuunnitelmaa ei ollut laadittu (Kuva 13). Vain kahdessa vastauksessa pääsuunnitelman kerrottiin kattavan kaikki testaustasot, eli yksikkö-, integraatio-, järjestelmä- ja hyväksymistestauksen. Hyväksymistestaus oli kuvattuna valtaosassa testauksen pääsuunnitelmista, kun puolestaan yksikkötestaus kaikkein harvimmin.



Kuva 13. Pääsuunnitelman kattamat testaustasot

Kysyttäessä, kuka vastaa pääsuunnitelman laatimisesta, hajontaa vastauksissa oli varsin paljon aina Scrum Masterista kehitystiimiin, mutta vastausten joukossa oli myös epätietoisuutta pääsuunnitelman laatijasta.

4.3 Tasokohtaisten testaussuunnitelmien laatiminen ja sisältö

Tasokohtaisia testaussuunnitelmia laadittiin kaikkien vastaajien organisaatioissa. Vain kolmessa tapauksessa tasokohtaiset suunnitelmat laadittiin kaikille testaustasoille. Kuvassa 13. on visualisoitu kaikki eri sisällöt, joiden kerrottiin esiintyneen tasokohtaisissa testaussuunnitelmissa. Tarkentavaan kysymykseen koskien sitä, miksi tasokohtainen testaussuunnitelma on jätetty laatimatta, oli vastausten perusteella tulkittavissa, että yleisimmin tasokohtaisten suunnitelmien laatimiseen liittyy usein erityistä tarveharkintaa. Myös kehityshankkeen koolla oli tässä merkitystä. Suurimmassa osassa tapauksista tasokohtaisia testaussuunnitelmia laati testauspäällikkö tai testauspäällikön rooliin rinnastettava henkilö.

Kuva 14. Tasokohtaisten testaussuunnitelmien sisältö

Selvä enemmistö ilmoitti testaussuunnitelmien sisältävän testauksen työkalut, testausmenetelmät sekä testiympäristöt. Testiaineistoon, skenaarioihin, testausstrategiaan, testianalyysin tuloksiin ja testauksen resursseihin kiinnitettiin sen sijaan kaikkein vähiten huomiota tasokohtaisissa testaussuunnitelmissa. Vastaukset olivat linjassaan tasokohtaisten testaussuunnitelmien laatimista koskevan kysymyksen kanssa, sillä yhdessäkään vastauksessa ei ilmaistu, että tasokohtaista testaussuunnitelmaa ei olisi laadittu lainkaan.

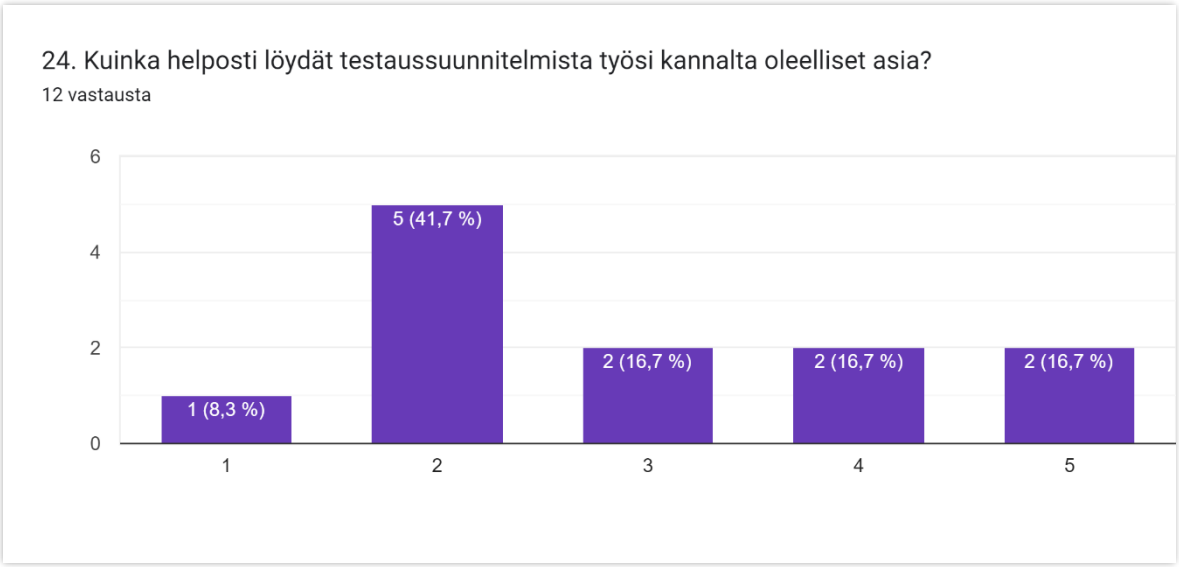
Selvä enemmistö oli käyttänyt testaussuunnitelmien dokumentointiin jotakin tehtävienhallintasovellusta, kuten Jiraa tai Trelloa, mutta myös verkkosivut, kuten Sharepoint ja Confluence mainittiin lähes yhtä monessa tapauksessa. Wordiakin käytettiin dokumentointiin, mutta selvästi harvemmin. Testaussuunnitelmia kerrottiin päivitettävän aktiivisesti suurimmassa osassa tapauksista.

4.4 Testaussuunnitelman merkitys ketterässä ohjelmistokehityksessä

Kysymykset 23–28 oli laadittu kartoittamaan vastaajien ajatuksia testaussuunnitelman laa-timisen merkityksellisyydestä. Valtaosan vastaajista mukaan testaussuunnitelmat eivät tue työskentelyä hyvin, mutta ei myöskään huonosti (Kuva 14).

Kuva 15. Testaussuunnitelma työskentelyn tukena

Selkeä enemmistö koki, ettei löydä erityisen hyvin oman työn kannalta olennaisia asioita (Kuva 15).

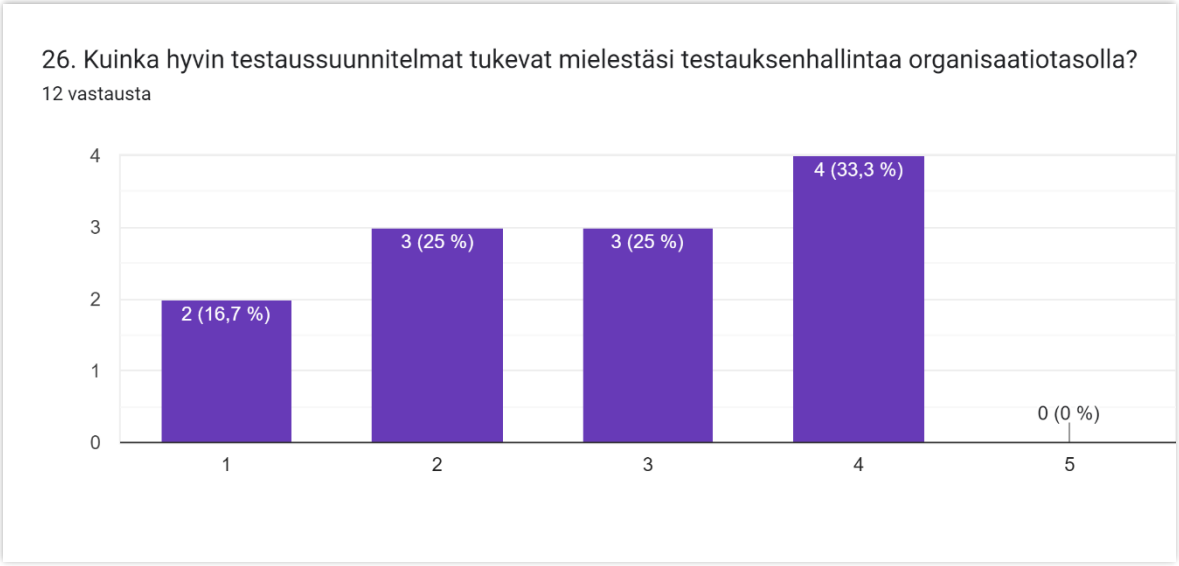


Kuva 16. Oman työn kannalta olennaisten asioiden löytäminen testaussuunnitelmasta

Tulosten perusteella oltiin hieman keskimääräistä tyytyväisempiä siihen, miten hyvin testaus on suunniteltu toteutettavan (Kuva 16).

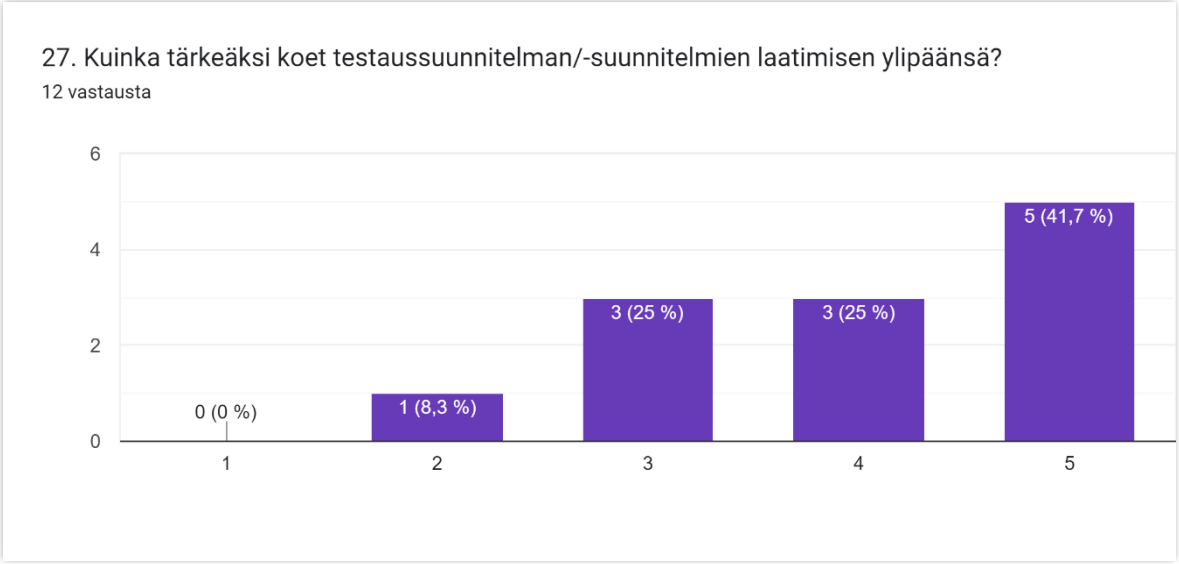
Kuva 17. Tiedonsaanti testauksen toteutuksesta

Testaussuunnitelmien koettiin tukevan testauksenhallintaa organisaatiotasolla kuitenkin hieman keskimääräistä huonommin (Kuva 17).



Kuva 18. Testaussuunnitelmien merkitys testauksenhallinnan tukena

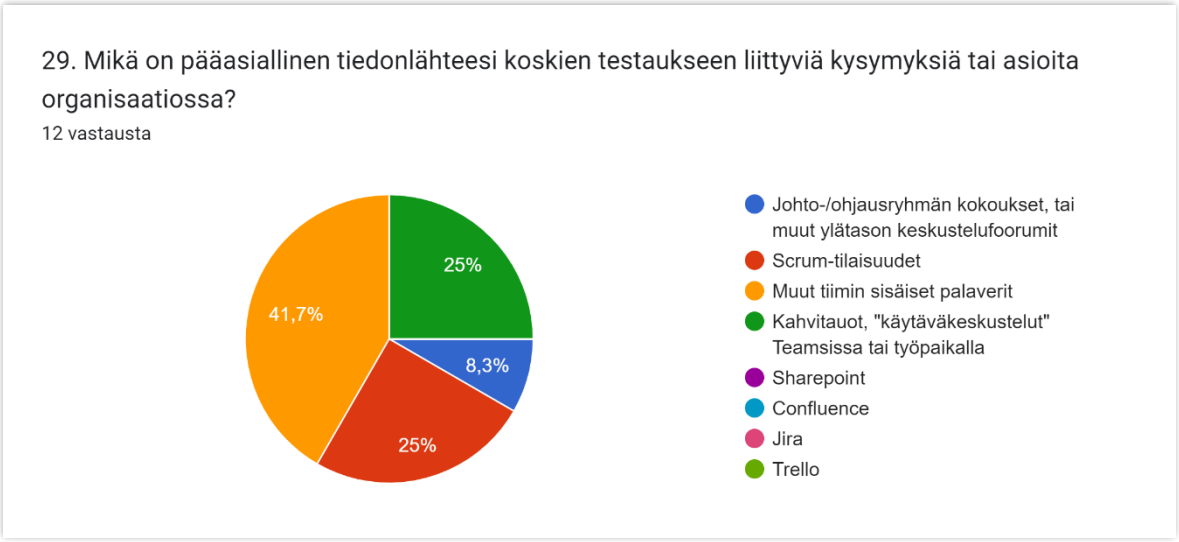
Testaussuunnitelma/-t koettiin yleisellä tasolla silti varsin merkityksellisiksi (Kuva 18), vaikka ne jätettiinkin monella tasolla laatimatta, tai laatiminen ei ollut säännönmukaista tai sisällöl-tään kaikilta osin standardien mukaista.



Kuva 19. Testaussuunnitelmien merkitys ylipäänsä

4.5 Tiedonlähteet testaukseen liittyvissä asioissa

Selvä enemmistö koki saavansa tietoa testaukseen liittyvistä asioista pääasiassa joko Scrum-tilaisuuksissa tai muissa tiimin sisäisissä palavereissa, mutta myös kahvitauot, käytäväkeskustelut ym. näyttelivät varsin suurta roolia tiedonvälityksen kanavana.



Kuva 20. Tiedonlähteet

Samaiset tiedonlähteet toistuivat myös toissijaisista tiedonlähteistä kysyttäessä, mutta myös ALM-työkalut, kuten Jira ja Confluence mainittiin useissa vastauksissa.

4.6 Yhteenveto

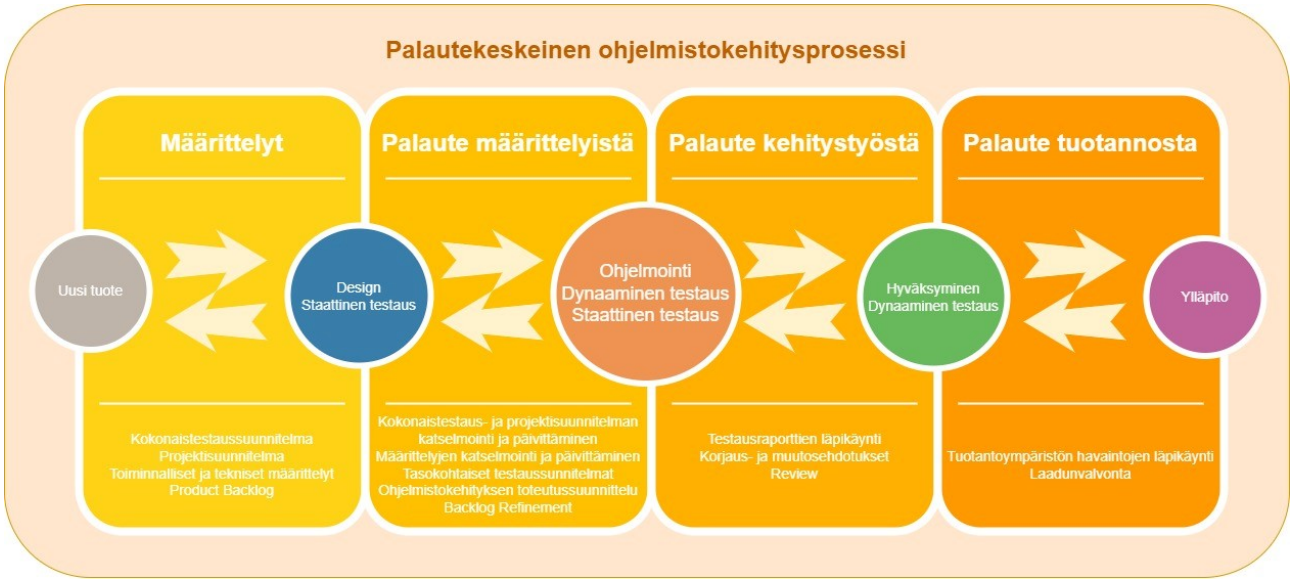
Valideja johtopäätöksiä kyselyn tulosten perusteella on vaikeaa tai jopa mahdoton tehdä, sillä vastauksia kyselyyn tuli lopulta niin vähän. Voidaan kuitenkin spekuloida sillä, onko sillä ollut vaikutusta testaussuunnitelman koettuun merkitykseen, että organisaatiotasolla testauksenhallinta vaikuttaa olleen heikommalla tasolla valtaosassa tapauksista ylätason dokumentaation puuttuessa. Se, että selkeä enemmistö ei kokenut löytävänsä erityisen hyvin oman työn kannalta oleellisia asioita testaussuunnitelmista ei kuitenkaan välttämättä kerro siitä, että testaussuunnitelmat olisivat epätarkkoja tai puutteellisia, tai että ne olisi huonosti dokumentoitu. Kysymyksen asettelun ollessa sellainen, joka olettaa, että testaussuunnitelmasta löytyy kaikkien työskentelyyn olennaisesti liittyvää tietoa, on mahdollista, että joidenkin vastaajien työhön testauksen tehtävillä ja prosessilla ei ole suoraa yhteyttä. Merkillepantavaa on kuitenkin erityisesti se, että ne henkilöt, jotka toimivat sellaisessa roolissa, joiden voisi olettaa olevan jollain lailla vastuussa ohjelmistokehitysprosessin sujuvuudesta, olivat pääosin sitä mieltä, että testaussuunnitelmista ei löydy oman työn kannalta oleellisia asioita kovinkaan helposti.

Testaussuunnitelman sisällöt vaihtelivat huomattavan paljon ja vaikka esimerkiksi standardien mukaan pääsuunnitelmassa tulisi kuvata kaikki testaustasot, oli vain harvassa tapauksista laadittu standardien mukainen päätestaussuunnitelma. Tavat dokumentoida testaussuunnitelmia vaihtelivat myös varsin paljon, mutta koska testaussuunnitelmien dokumentointia koskevat kysymykset oli asetettu siten, että avoimella vastauksella oli mahdollista halutessaan tarkentaa dokumentointiin liittyviä seikkoja, ei tähän juurikaan tullut vastauksia, joten jäi lopulta varsin epäselväksi, miten dokumentaatiota organisaatioissa käytännössä toteutetaan, vaikka joitakin sovelluksia olikin monivalintakysymyksissä mainittu.

Kaiken kaikkiaan kyselyn tulokset vahvistivat käsitystä siitä, että testaussuunnitelman laatimiseen ja sisältöön liittyvät käytännöt vaihtelevat huomattavan paljon, eikä ohjelmistotestauksen standardien suosittamia käytäntöjä useinkaan noudateta, eikä niiden olemassaoloakaan aina tiedosteta.

5 Kehitysehdotus

Jotta opinnäytetyöni tuloksia voisi hyödyntää tulevaisuudessa, olisi varmasti hyödyllistä tarkentaa uudella tutkimuksella nyt saatuja tuloksia siitä näkökulmasta, mikä merkitys testaussuunnitelman laatimisella (tai laatimatta jättämisellä) on ohjelmistokehitysprosessin sujuvuuteen ylipäänsä, sillä tässä kyselyssä ei esimerkiksi kartoitettu sitä, mikä vaikutus testaussuunnitelman laatimisella oli ohjelmistokehitysprosessin sujuvuuteen tai tehokkuuteen, valmiin ohjelmistotuotteen laatuun tai muihinkaan seikkoihin, joilla yleisesti prosessin sujuvuutta tai laatua voidaan mitata. Siitä huolimatta koin merkitykselliseksi ehdottaa tässä uutta ohjelmistokehitysprosessin mallia (kuva 21), sillä opinnäytetyöprosessi ja työni testauksen tehtävissä ovat saaneet minut tarkastelemaan kriittisesti erilaisia ohjelmistokehitysprosesseja. Edelleenkään en ole nimittäin vakuuttunut siitä, että yksikään tunnettu ketterän ohjelmistokehityksen malli tunnistaa testauksen (myös liiketoiminnan tekemän testauksen) merkityksen ensisijaisen arvokkaana palautekanavana, eikä sen merkitystä riittävästi näissä malleissa korosteta. Kehittelemäni malli perustuu siksi ennen kaikkea testauksesta saatavan palautteen antamiseen ja vastaanottamiseen sekä tiedonkulkuun liiketoiminnan, kehitystiimin ja sidosryhmien välillä erityisesti testauksen tuotosten pohjalta. Vaikka testauksen sertifikaattisisällöissäkin todetaan, että liiketoiminnan edustajien olisi hyvä olla mukana testaamassa järjestelmän ominaisuuksia, näyttäytyy liiketoiminnan edustajien rooli kuitenkin ohjelmistokehitysprosesseissa varsin vähäisenä. Tässä mallissa liiketoiminta kytkeytyy vahvasti ohjelmistokehitysprosessiin aina määrittelyistä ja tuotantokäyttöön hyväksymiseen saakka, vaikka varsinainen kehitystiimi vastaakin ohjelmistokehityksen asiantuntemuksesta teknisen määrittelyn ja varsinaisen ohjelmointityön ja teknisen testauksen osalta.



Kuva 21. Palautekeskeinen ohjelmistokehitysprosessi

Malli mukailee Scrum-mallin mukaista ohjelmistokehitysprosessia, ja esimerkiksi Scrum-tilaisuudet ja roolit ovat kutakuinkin samat, mutta korostaa testaussuunnitelmien ja -raportoinnin merkitystä palautteenannon välineenä sekä liiketoiminnan roolia kaikissa ohjelmistokehitysprosessin vaiheissa. Kun kehitetään esimerkiksi uutta tuotetta, määrittelyt tuotetaan yhteistyössä esimerkiksi liiketoiminnan asiantuntijoiden kanssa, jotka myös vastaavat siitä, millä kriteereillä tuote voidaan hyväksyä tuotantokäyttöön. Kokonaistestaus- ja mahdollisen projektisuunnitelman laatimisesta puolestaan tulisi vastata se taho, jolla on eniten asiantuntemusta ja ymmärrystä tuotteen kehitykseen liittyvistä riskeistä niin liiketoiminnan kuin ohjelmistokehityksenkin näkökulmasta. Näin ollen testaus- ja projektisuunnitelmien katselmointi ja niistä saatava palaute on ensisijaisen tärkeää.

Siinä vaiheessa, kun varsinaista ohjelmistokoodia ei vielä ole kirjoitettu, on kaiken kaikkiaan erityisen tärkeää tehdä staattista testausta, eli esimerkiksi dokumentaation, kuten määrittelyjen ja suunnitelmien katselmointia. Tämä tapahtuu luonnollisimmin testaussuunnittelun kautta, kun testaussuunnittelussa tulee määritellä testattavat tilanteet käyttötapauksiin ja kokonaistestaussuunnitelmaan peilaten. Kun katselmoinnin tulokset dokumentoidaan ja kommunikoidaan kehitystiimille, muotoutuu varsinainen toteutussuunnitelma ja alempien testaustasojen suunnitelmat. Varsinaisessa kehitysvaiheessa tehdään ohjelmoinnin rinnalla jatkuvaa dynaamista ja staattista testausta ja niiden tulokset raportoidaan katselmointitilaisuuksissa. Jos tuotos edellyttää vielä kaikkien testausvaiheiden ja niistä saadun palautteen jälkeenkin vielä hyväksymistestauksen, se tulee sisällyttää kehitysprosessiin.

Ylläpito voi käynnistää tarvittaessa uuden prosessin, jos havaitaan tarve tehdä parannuksia tuotokseen sen jälkeen, kun tuote on jo julkaistu käyttöön. Tuotantovaiheessa tuleekin seurata huolella esimerkiksi asiakas- ja käyttäjäpalautetta sekä selkeitä tuotantokäytössä ilmeneviä virheitä. Tärkeää on liiketoiminnan toimijoiden roolien ja vastuun tunnistaminen prosessissa sekä testauksen mieltäminen ensisijaisen tärkeäksi palautteenannon kanavaksi ohjelmistokehityksen sujuvuuden ja tuotteen laadun näkökulmasta.

Malli ei ota kantaa siihen, kuinka kevyttä tai raskasta dokumentaatiota suunnitelmista tulee tuottaa. Pääasia on se, että on sovittu yhteisesti, mikä dokumentaation taso on riittävä ja kaikilla avainhenkilöillä on tieto siitä, mistä dokumentaatio on löydettävissä. Strategiassa olisi kuitenkin hyvä olla määriteltynä organisaation kaikkien käynnissä olevien projektien tiekartta, jossa puolestaan kokonaistestaussuunnitelma on laadittu projektitasolle. Tämän voisi toteuttaa helposti esim. Jira-työkaluilla hyödyntämällä eri tason tehtävätyyppejä. Muiden tasojen testaussuunnitelmat tai tehtävät olisi niin ikään dokumentoitava samaan järjestelmään, jolloin riippuvuudet eri projektien ja toimintojen välillä tehdään näkyväksi.

6 Pohdinta

Kuluneet pari vuotta ovat olleet valitettavan haastavia opinnäytetyöni edistämisen kannalta. Kun aloitin vuonna 2020 uudessa IT-alan työpaikassa, puhkesi COVID-19 pandemia ja uuteen työyhteisöön integroituminen ja uusien asioiden opettelu vei voimavaroja opinnäytetyön tekemiseltä, vaikka alkuperäistä ideaa tulikin jo tuolloin hahmoteltua. Vaihdoin lopulta työpaikkaa vuoden 2021 alkupuolella ja jälleen olin uuden opettelun edessä. Uuden toimenkuvani liittyessä edelleen testauksen tehtäviin, alkoi opinnäytetyön lopullinen idea hahmottua työkokemuksenkin karttuessa. Halusin silti tehdä opinnäytetyöni ilman varsinaista toimeksiantajaa, sillä halusin kartoittaa erilaisten organisaation toimintatapoja laatia testaussuunnitelmia ketterässä ohjelmistokehityksessä. Edellä mainituista syistä lukuisten lähteiden perusteellista läpikäyntiä sekä kyselyn toteutusta ja tulosten analysointia ei tullut varsinaisesti toteutettua menetelmäkirjallisuuteen pohjaten.

Koska ketterään kehitykseen kuuluu jatkuva muutos, mielestäni organisaatiotasolla tulisi kuitenkin kiinnittää huomiota testauksenhallintaan ja mukauttaa käytettävät testausmenetelmät, -työkalut ja käytännöt kulloinkin käytössä olevan ohjelmistokehityksen mallin mukaan siten, että ne tukevat ohjelmistokehityksen kaikkia vaiheita. Koska testauksen työkalut ja menetelmät kehittyvät jatkuvasti, varsinkin organisaatioissa, joissa päätökset hankinnoista tehdään organisaatiotasolla, on tärkeää, että testauksenhallintaa myös säännöllisesti tarkastellaan ja arvioidaan kriittisesti organisaatiotasolla, jotta käytössä olevista työkaluista, menetelmistä ja käytännöistä saadaan maksimaalinen hyöty irti, ja jotta testausprosesseja voidaan kehittää niin testaajien työn näkökulmasta kuin tuotteen laadunhallinnankin näkökulmasta entistä tehokkaammaksi. Testaussuunnitelmat niin organisaatiotasolla kuin alemmilla testaustasoillakin auttaa tekemään näkyväksi ne käytännöt, työkalut ja menetelmät, jotka testauksessa on käytössä. Vain tarkastelemalla testausta kokonaisuutena voidaan nähdä mahdollisia pullonkauloja tai kehityskohteita prosesseissa.

Vaikka ketterässä ohjelmistokehityksessä koko kehitystiimi kantaa vastuuta lopputuotoksen laadusta, on mielestäni kuitenkin tärkeää, että laatua seurataan johdonmukaisesti myös ylätasolla. Viime kädessä kehitettävien tuotteiden laatu on liiketoiminnan etu. Siksi sitä mielestäni sovi sälyttää vain kehitystiimin vastuulle varsinkaan, jos kyseessä on sellainen tuote, jonka kehittämiseen liittyy useita eri järjestelmiä ja tiimejä. Ketterissä ohjelmistokehitysmalleissa ei juuri löydy mainintoja testauspäällikön, projektipäällikön, laatupäällikön tai kehityspäällikön rooleista, eikä sellainen rooli kuulu kehitystiimin kokoonpanoon, vaikka jollain kehitystiimin jäsenistä olisikin suurempi vastuu testauksen ja laadunhallinnan osa-alueesta. Erityisesti sellaisissa organisaatioissa, joissa toiminta koostuu useiden eri tiimien tekemästä tuotekehityksestä, lienee ylätason rooli paikallaan ja testaussuunnitelman laatiminen enemmän tarpeen kuin pienemmissä organisaatioissa, joissa prosessien hallittavuus on kaiken kaikkiaan helpompaa.

Lähteet

Abrahamson, P., Salo, O., Ronkainen, J., Warsta, J. 2019. Agile Software Development Methods: Review and Analysis. VTT:n julkaisuja. Otamedia Oy. Espoo. Luettavissa: https://www.vttresearch.com/sites/default/files/pdf/publications/2002/P478.pdf. Luettu 14.5.2022.

Agilegnostic 2015. Luettavissa: https://agilegnostic.wordpress.com/2015/09/13/lean-kanban-methodology-to-application-support-and-maintenance/. Luettu 14.5.2022.

FiSTB 2015a. ISTQB:n testaussanasto v. 2.3 Suomi – Englanti. Luettavissa: http://fistb.t.tivia.fi/sites/fistb/files/liitteet/FiSTB_sanasto_2015-04-30%202.3%20FI-ENG.pdf. Luettu 12.2.2022

FiSTB 2015b. Perustason sertifikaattisisällön laajennus. Ketterä testaaja. Luettavissa: http://fistb.t.tivia.fi/sites/fistb/files/liitteet/FND-Agile-Syllabus_FI.pdf. Luettu 12.2.2022

FiSTB 2018. Sertifioitu testaaja. Perustason sertifikaattisisältö. Luettavissa: http://fistb.t.tivia.fi/sites/fistb/files/liitteet/CTFL%202018%20Sertifikaattisisa%CC%88lto%CC%88%2020181010-1%20Valmis.pdf. Luettu 12.2.2022

Harju, Sanna 2021. Vaatimusmäärittelyn merkitys palveluna ostetun tietojärjestelmän kehittämisessä. YAMK opinnäytetyö. Haaga-Helia ammattikorkeakoulu. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/500176/Harju_Sanna.pdf?sequence=2&isAllowed=y. Luettu 12.2.2022.

Heikkinen, Tiina 2018. DevOps ja sen yhteensovittaminen IT-palvelutuotantoon Luettavissa: https://jyx.jyu.fi/bitstream/handle/123456789/59867/URN%3aNBN%3afi%3ajyu-201810184446.pdf?sequence=1&isAllowed=y. Luettu 14.5.2022.

Hendrickson, Elisabeth 2008. Luettavissa: http://1iv3y5147rw9384bl74cgkcm-wpengine.netdna-ssl.com/wp-content/uploads/2011/04/atddexample.pdf Luettu 14.5.2022.

IEEE 2013. International Standard - Software and systems engineering —Software testing —Part 1: Concepts and definitions. Luettavissa: https://standards.ieee.org/ieee/29119-1/5308/. Luettu 12.2.2022

IEEE 2021. Software and systems engineering – Software testing – Part 2: Test processes. Luettavissa: https://standards.ieee.org/standard/29119-2-2021.html. Luettu 14.5.2022.

Juvonen, Rami 2018. Ohjelmistoprojektin sudenkuopat ja miten ne vältetään. E-kirja. BoD - Books on Demand. Helsinki. Luettu 14.5.2022.

Kasurinen, Jussi-Pekka 2013. Ohjelmistotestauksen käsikirja. E-kirja. Docendo. Jyväskylä.

Khan, R., Srivastava, A., Pandey, D. 2016. Agile approach for Software Testing process. Luettavissa: https://www.researchgate.net/publication/315914633_Agile_approach_for_Software_Testing_process. Luettu 12.2.2022.

Koski, Timo 2012. Kanban: periaatteita ja kokemuksia. Tietojenkäsittelytieteiden laitos. Jyväskylän yliopisto. Luettavissa: https://jyx.jyu.fi/bitstream/handle/123456789/40566/Timo%20Koski.pdf;jsessionid=A403206E1D0D4093D13F9667466EBC71?sequence=1. Luettu 11.9.2022

Koskinen, Anna 2020. Building Security Into the Core of DevOps. Informaatioteknologian tiedekunta. Jyväskylän yliopisto. Luettavissa: https://jyx.jyu.fi/bitstream/handle/123456789/67345/URN:NBN:fi:jyu-202001171290.pdf?sequence=1. Luettu 11.9.2022.

Lahtinen, Anssi 2020. Testausautomaatiostrategian luominen DevOps-ympäristössä: Tietohallinnon näkökulma. Informaatioteknologian tiedekunta. Jyväskylän yliopisto. Luettavissa: https://jyx.jyu.fi/bitstream/handle/123456789/69926/URN:NBN:fi:jyu-202006154170.pdf?sequence=1. Luettu 11.9.2022.

Lampinen, Henriikka 2019. Teknisen dokumentaation haasteet ketterässä järjestelmäkehityksessä. Informaatioteknologian tiedekunta. Jyväskylän yliopisto. Luettavissa: https://jyx.jyu.fi/bitstream/handle/123456789/66687/URN:NBN:fi:jyu-201912105155.pdf?sequence=1.

Lappalainen, Tuire 2019. Testaus osana ohjelmistokehitystä. Informaatioteknologian tiedekunta. Jyväskylän yliopisto. Luettavissa: https://jyx.jyu.fi/bitstream/handle/123456789/66829/URN:NBN:fi:jyu-201912165321.pdf?sequence=1. 11.9.2022.

Laukkarinen, Emmi 2018. Scrumin haasteet tietojärjestelmäkehitysprojekteissa. Informaatioteknologian tiedekunta. Jyväskylän yliopisto. Luettavissa: https://jyx.jyu.fi/bitstream/handle/123456789/62442/URN:NBN:fi:jyu-201901111160.pdf?sequence=1. Luettu 14.5.2022.

Lehtonen, Teijo 2014. Sulautettujen järjestelmien ketterä käsikirja. https://www.utupub.fi/bitstream/handle/10024/99142/Sulautettujen_jarjestelmien_kettera_kasikirja_Painos1.pdf?sequence=2&isAllowed=y Luettu 14.5.2022.

Lindberg, Harri 2003. Extreme Programming. Tietojenkäsittelytieteen laitos. Tampereen yliopisto. Luettavissa: https://trepo.tuni.fi/bitstream/handle/10024/91382/Lindberg_Harri.pdf;jsessionid=900F5689314CB48629672C232E4E7E16?sequence=1. Luettu 11.9.2022.

Perälä, Juuso 2018. Testivetoisen ohjelmistokehityksen edut ja haitat ohjelmistoprojekteissa. Informaatioteknologian tiedekunta. Jyväskylän yliopisto. Jyväskylä. Luettavissa: https://jyx.jyu.fi/bitstream/handle/123456789/58147/URN%3aNBN%3afi%3ajyu-201805282814.pdf?sequence=1&isAllowed=y. Luettu 12.2.2022.

Plutora 2022. DevSecOps: A Complete Guide to What, Why, and How. Luettavissa: https://www.plutora.com/blog/devsecops-guide. Päivitetty 4.5.2022. Luettu 14.5.2022.

Schwarber, K., Sutherland, J. 2020. Scrum-opas. Scrumin määritelmä ja pelisäännöt. Viitattu 16.1.2022. Luettavissa: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Finnish.pdf. Luettu 14.5.2022.

Scrum.org 2021. Luettavissa: https://scrumorg-website-prod.s3.amazonaws.com/drupal/2021-01/Scrumorg-Scrum-Framework-tabloid.pdf. Luettu 12.2.2022.

Sommerville, Ian 2016. Software engineering. E-Kirja. Pearson Education. 10. painos.

Liitteet

Liite 1. Kyselylomakkeen saate ja kysymykset

Testaussuunnitelma ketterässä ohjelmistokehityksessä

Työskenteletkö ketterän ohjelmistokehityksen parissa? Jos vastasit kyllä ja ohjelmistojen laadunhallinta, laadunvarmistus ja/tai testaus liittyy millään tavalla toimenkuvaasi, tämä kysely on juuri sinulle. Jos et miellä kuuluvasi kohderymään, mutta tunnet jonkun, joka voisi sopia profiiliin, voit vapaasti jakaa kyselyn linkin eteenpäin verkostossasi.

Tämä kysely on osa Haaga-Helia AMK:n IT-tradenomiopintojeni opinnäytetyötä ja toivoisin, että voisit käyttää pienen hetken vastataksesi oheiseen kyselyyn. Kaikki vastaukset käsitellään anonyymisti, eikä vastauksia voi yhdistää yksittäiseen henkilöön. Kyselyyn vastaamiseen menee noin 10 minuuttia.

Jos työskentelet konsulttina asiakaorganisaatiossa, vastaa kyselyyn niiden toimintamallien ja käytäntöjen mukaan, joita asiakasorgasnisaatiossa noudatetaan.

Kiitos etukäteen vastauksistasi!

Kyselyssä yhteyshenkilönä toimii kyselyn toteuttaja,

Outi Ruusunen
outiruusunen@gmail.com

*Pakollinen

1. 1. Mikä on nykyinen roolisi ohjelmistokehityksessä? *

2. 2. Kuinka kauan olet toiminut nykyisessä roolissasi? *

   Merkitse vain yksi soikio.

   - ◯ 0-1 vuotta
   - ◯ 2-4 vuotta
   - ◯ 5-10 vuotta
   - ◯ yli 10 vuotta

3. 3. Ikäsi *

   Merkitse vain yksi soikio.

   - ◯ 24 vuotta tai alle
   - ◯ 25-41 vuotta
   - ◯ 42-59 vuotta
   - ◯ 60 vuotta tai yli
   - ◯ En halua kertoa

4. 4. Minkä kokoisessa organisaatiossa työskentelet? *

   Merkitse vain yksi soikio.

   - ◯ Mikroyritys (enintään 10 hlöä)
   - ◯ Pienyritys (enintään 50 hlöä)
   - ◯ Keskisuuri yritys (enintään 250 hlöä)
   - ◯ Suuryritys (vähintään 250 hlöä)

5. 5. Minkä kokoisessa tiimissä työskentelet? *

Merkitse vain yksi soikio.

- ◯ 1-3 hlöä
- ◯ 4-9 hlöä
- ◯ 10 hlöä tai enemmän
- ◯ En osaa sanoa

6. 6. Minkä ketterän ohjelmistokehityksen viitekehyksen periaatteiden tai toimintamallin mukaan organisaatiossa toimitaan? *

Voit valita useita vaihtoehtoja. Lisää tarvittaessa vaihtoehto "Muu" ja kirjoita, mikä muu.

Valitse kaikki sopivat vaihtoehdot.

- ☐ Scrum
- ☐ Kanban
- ☐ Scrumban
- ☐ SAFe
- ☐ Extreme Programming
- ☐ Test Driven Development/Acceptance Test Driven Development
- ☐ DevOps/DevSecOps
- ☐ Muu: _____

7. 7. Työskenteletkö... *

Merkitse vain yksi soikio.

- ◯ Julkisella sektorilla
- ◯ Yksityisellä sektorilla
- ◯ Kolmannella sektorilla
- ◯ Muu: _____

8. 8. Kuinka pitkän julkaisusyklin mukaan työskentelette?

Merkitse vain yksi soikio.

- ◯ Jatkuva julkaisu/julkaisu aina kun jotain valmistuu
- ◯ 1 viikko + mahdolliset ad hoc-julkaisut
- ◯ 2 viikkoa + mahdolliset ad hoc-julkaisut
- ◯ Pidempi kuin 2 viikkoa
- ◯ Muu: _____

9. 9. Noudatetaanko organisaatiossanne ohjelmistotestauksen kansainvälisten ISO/IEC 29119 standardien mukaisia testauskäytäntöjä? *

Merkitse vain yksi soikio.

- ◯ Kyllä
- ◯ Ei
- ◯ En osaa sanoa

10. 10. Onko organisaatiossanne laadittu testauspolitiikka? *

Tässä testauspolitiikalla tarkoitetaan *korkean tason* dokumenttia, joka kuvaa periaatteet, lähestymistavat ja tärkeimmät tavoitteet, joita organisaatiolla on testaukseen liittyen. (Lähde: FiSTB)

Merkitse vain yksi soikio.

- ◯ Kyllä
- ◯ Ei
- ◯ En osaa sanoa
- ◯ Muu: _____

11. 11. Onko organisaatiossanne laadittu testausstrategia?

Tässä testausstrategialla tarkoitetaan *korkean tason* kuvausta käytettävistä testaustasoista ja testauksesta näillä tasoilla. Kuvaus voidaan tehdä organisaatio- tai projektitasolla (yhdelle tai useammalle projektille). (Lähde: FiSTB)

Merkitse vain yksi soikio.

- ◯ Kyllä
- ◯ Ei
- ◯ En osaa sanoa
- ◯ Muu: _____

12. 12. Onko organisaatiossanne laadittu testauksen pääsuunnitelma? *

Tässä organisaation testauksen pääsuunnitelmalla tarkoitetaan nk. projektikohtaista Master Test Plania, testauksen pääsuunnitelmaa tai kokonaistestaussuunnitelmaa, joka kattaa useampia testaustasoja, kuten yksikkö-, integraatio-, järjestelmä- ja hyväksymistestauksen. (Lähde: FiSTB)

Merkitse vain yksi soikio.

- ◯ Kyllä
- ◯ Ei
- ◯ En osaa sanoa
- ◯ Muu: _____

13. 13. Mitkä testaustasot testauksen pääsuunnitelma kattaa? *

Valitse kaikki sopivat vaihtoehdot.

- ☐ Yksikkötestaus
- ☐ Integraatiotestaus
- ☐ Järjestelmätestaus
- ☐ Hyväksymistestaus
- ☐ En osaa sanoa
- ☐ Testauksen pääsuunnitelmaa ei ole laadittu
- ☐ Muu: _____

14. 14. Jos testauksen pääsuunnitelmaa ei ole laadittu, kerro tässä miksi ei.

_____

_____

_____

_____

_____

15. 15. Kuka organisaatiossanne on vastuussa testauksen pääsuunnitelman laatimisesta? Kerro rooli organisaatiossa. *

Jos testaussuunnitelmaa ei ole laadittu, vastaa "-" ja siirry seuraavaan kysymykseen.

_____

16. 16. Mitä tasokohtaisia testaussuunnitelmia organisaatiossanne on laadittu? *

Voit valita useita vaihtoehtoja.

Valitse kaikki sopivat vaihtoehdot.

- [ ] Yksikkötestaus
- [ ] Integraatiotestaus
- [ ] Järjestelmätestaus
- [ ] Hyväksymistestaus
- [ ] En osaa sanoa
- [ ] Testaussuunnitelmaa ei ole laadittu
- [ ] Muu: _____

17. 17. Jos tasokohtaisia testaussuunnitelmia ei ole laadittu, kerro tässä miksi ei.

_____

_____

_____

_____

_____

18. 18. Kuka/ketkä organisaatiossanne on vastuussa tasokohtaisten testaussuunnitelmien laatimisesta? Kerro rooli organisaatiossa

Jos testaussuunnitelmaa ei ole laadittu, vastaa "-" ja siirry seuraavaan kysymykseen.

_____

_____

_____

_____

_____

19. 19. Mitä seuraavista asioista testaussuunnitelmissa on kuvattu? *

Voit valita useita vaihtoehtoja.

Valitse kaikki sopivat vaihtoehdot.

- [ ] Testauksessa käytettävä testausstrategia
- [ ] Testauksen työkalut
- [ ] Testausmenetelmät
- [ ] Testianalyysin tulokset
- [ ] Manuaalitestitapaukset
- [ ] Automaatiotestitapaukset
- [ ] Testiskriptit
- [ ] Testiaineisto
- [ ] Skenaariot
- [ ] Käyttäjätarinat
- [ ] Testiympäristöt
- [ ] Testauksen resurssit
- [ ] Tiedon siitä, miten testauksen tulokset raportoidaan
- [ ] En osaa sanoa
- [ ] Testaussuunnitelmaa ei ole laadittu
- [ ] Muu: _____

20. 20. Miten testaussuunnitelmat on dokumentoitu? *

Voit valita useita vaihtoehtoja. Tarkenna vastaustasi halutessasi seuraavassa avoimessa kysymyksessä.

Valitse kaikki sopivat vaihtoehdot.

- [ ] Word-dokumentti
- [ ] PowerPoint-esitys
- [ ] Excel-taulukko
- [ ] Teams-kanava
- [ ] Tehtävänhallintasovelluksessa (esim. Jira, Trello)
- [ ] Verkkosivulla (esim. Sharepoint, Confluence)
- [ ] Testaussuunnitelmia ei ole laadittu
- [ ] Muu: _____

21. 21. Tarkenna tässä halutessasi edellistä vastaustasi.

Kuvaile tarkemmin, miten testaussuunnitelmat on dokumentoitu.

_____

_____

_____

_____

_____

22. 22. Päivitetäänkö testaussuunnitelmia aktiivisesti? *

Merkitse vain yksi soikio.

- ◯ Kyllä
- ◯ Ei
- ◯ En osaa sanoa
- ◯ Testaussuunnitelmaa ei ole laadittu

23. 23. Kuinka hyvin testaussuunnitelmat mielestäsi tukevat työskentelyäsi? *

Merkitse vain yksi soikio.

	Ei lainkaan hyvin				Erittäin hyvin
	1	2	3	4	5

24. 24. Kuinka helposti löydät testaussuunnitelmista työsi kannalta oleelliset asia? *

Merkitse vain yksi soikio.

En lainkaan helposti

1 ○
2 ○
3 ○
4 ○
5 ○

Erittäin helposti

25. 25. Saatko mielestäsi riittävästi tietoa siitä, kuinka järjestelmän testaus on suunniteltu toteutettavan? *

Merkitse vain yksi soikio.

En lainkaan riittävästi

1 ○
2 ○
3 ○
4 ○
5 ○

Riittävästi

26. 26. Kuinka hyvin testaussuunnitelmat tukevat mielestäsi testauksenhallintaa organisaatiotasolla? *

Merkitse vain yksi soikio.

Ei lainkaan hyvin

1 ○
2 ○
3 ○
4 ○
5 ○

Erittäin hyvin

27. 27. Kuinka tärkeäksi koet testaussuunnitelman/-suunnitelmien laatimisen ylipäänsä? *

Merkitse vain yksi soikio.

	En lainkaan merkitykselliseksi
1	◯
2	◯
3	◯
4	◯
5	◯
	Erittäin merkitykselliseksi

28. 28. Perustele tässä halutessasi edellistä vastaustasi.

29. 29. Mikä on **pääasiallinen** tiedonlähteesi koskien testaukseen liittyviä kysymyksiä tai asioita organisaatiossa? *

Merkitse vain yksi soikio.

- ◯ Johto-/ohjausryhmän kokoukset, tai muut ylätason keskustelufoorumit
- ◯ Scrum-tilaisuudet
- ◯ Muut tiimin sisäiset palaverit
- ◯ Kahvitauot, "käytäväkeskustelut" Teamsissa tai työpaikalla
- ◯ Sharepoint
- ◯ Confluence
- ◯ Jira
- ◯ Trello
- ◯ Muu: _____

30. 30. Mitä muita tiedonlähteitä sinulla on koskien testaukseen liittyviä kysymyksiä tai asioita organisaatiossa?

Valitse kaikki sopivat vaihtoehdot.

- ☐ Johto-/ohjausryhmän kokoukset, tai muut ylätason keskustelufoorumit
- ☐ Scrum-tilaisuudet
- ☐ Muut tiimin sisäiset palaverit
- ☐ Kahvitauot, "käytäväkeskustelut" Teamsissa tai työpaikalla
- ☐ Sharepoint
- ☐ Confluence
- ☐ Jira
- ☐ Trello
- ☐ Muu: _____

31. Kiitos vastauksistasi! Jätä vielä tähän halutessasi palautetta tai kommentteja kyselystä.