TESTAUKSEN KEHITTÄMINEN KETTERISSÄ OHJELMISTONKEHITYSPROJEKTEISSA

Lappeenrannan–Lahden teknillinen yliopisto LUT

Tuotantotalouden diplomityö

2023

Suvi Taussi

Tarkastajat: Dosentti Jouni Koivuniemi

Tutkijatohtori Lasse Metso

TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT

LUT Teknis-luonnontieteellinen

Tuotantotalous

Suvi Taussi

Testauksen kehittäminen ketterissä ohjelmistonkehitysprojekteissa

Tuotantotalouden diplomityö

2023

70 sivua, 15 kuvaa ja 5 taulukkoa

Dosentti Jouni Koivuniemi, tutkijatohtori Lasse Metso

Avainsanat: ketterät menetelmät, laadunvarmistus, testaus

Ketterin menetelmin toteutetut ohjelmistonkehitysprojektit tuovat oman haasteensa laadunvarmistukseen. Syklit ovat ripeitä ja asiakas tahtoo tuotosta nopealla aikataululla. Miten laadunvarmistus suoritetaan onnistuneesti ketterissä ohjelmistonkehitysprojekteissa? Mitkä ovat prosessin mahdolliset haasteet?

Tutkimuksen tarkoituksena oli selvittää, miten laadunvarmistusprosessi etenee ketterissä ohjelmistonkehitysprojekteissa sekä mitkä ovat mahdolliset prosessissa ilmenevät haasteet ja millä tavoin testausprosessia voisi tehostaa ja kehittää.

Työ toteutettiin toimeksiantajan pyynnöstä. Tutkimuksessa on käytetty laadullista eli kvalitatiivista tutkimusmenetelmää. Tutkimuksen aineiston keruu suoritettiin haastattelujen kautta. Haastateltavia oli kuusi kappaletta ja he kaikki työskentelivät eri asiakasprojekteissa toistaan erilaisissa työtehtävissä.

Tutkimuksessa selvisi, että laadunvarmistus ketterin menetelmin toteutettavissa ohjelmistonkehitysprojekteissa sisälsi erilaisia haasteita ja ongelmia. Suurimpia haasteita toimeksiantajan asiakasprojekteissa aiheuttivat tiimien siiloutuminen ja viestinnän takkuaminen. Toinen suurempi haaste projekteissa oli kokonaisuuden hallinta.

Diplomityön tuloksena valmistui kehitysehdotus siitä, miten toimeksiantajan kehitystiimi voisi parantaa toimintaansa asiakasprojektissa, jossa kommunikaatiokatkokset aiheuttivat eniten haasteita projektin etenemiselle. Kehitysehdotus painottuu tiedonjakamiseen tiimien kesken sekä haasteiden ja ongelmien aktiiviseen ratkaisemiseen.

ABSTRACT

Lappeenranta–Lahti University of Technology LUT

School of Engineering Science

Industrial Engineering and Management

Suvi Taussi

Improving testing in agile system development projects

Master’s thesis

2023

70 pages, 15 figures and 5 tables

Dosentti Jouni Koivuniemi, tutkijatohtori Lasse Metso

Keywords: agile methods, quality assurance, testing

Software development projects implemented using Ketter's methods bring their own challenges to quality assurance. The cycles are fast, and the customer wants a return on a fast schedule. How is quality assurance successfully performed in agile software development projects? What are the potential challenges of the process?

The purpose of the study was to find out how the quality assurance process progresses in agile software development projects, as well as what are the possible challenges in the process and how the testing process could be improved and developed.

The work was carried out at the request of the client. The research has used a qualitative research method. The collection of research material was carried out through interviews. There were six interviewees and they all worked on different customer projects in different jobs.

The research revealed that quality assurance in software development projects implemented using agile methods contained various challenges and problems. The biggest challenges in the client projects of the client were caused by the siloing of teams and communication problems. Another big challenge in the projects was overall management.

As a result of the diploma thesis, a development proposal was completed on how the client's development team could improve its operations in a customer project where communication breaks caused the most challenges for the project's progress. The development proposal focuses on sharing information between teams and actively solving challenges and problems.

Sisällysluettelo

Tiivistelmä

Abstract

1 Johdanto ... 4
1.1 Diplomityön tavoitteet ja tutkimuskysymykset ... 4
1.2 Diplomityön rajaukset ... 5
1.3 Teoreettinen viitekehys ja kirjallisuuskatsaus ... 5
1.4 Diplomityön rakenne ... 6
2 Ketterä ohjelmistokehitys ... 9
2.1 Ketterän ohjelmistokehityksen manifesti ... 9
2.2 Ketterän kehityksen periaatteet ... 10
2.3 Vesiputousmalli vs ketterät menetelmät ... 11
2.4 Ketterä testaus ... 15
3 Testauksen tasot ... 17
3.1 Yksikkötestaus ... 17
3.2 Integraatiotestaus ... 17
3.3 Järjestelmätestaus ... 18
3.4 Hyväksymistestaus ... 18
4 Testaus- ja kehitysmenetelmät ... 20
4.1 Staattinen testaus ... 20
4.1.1 Katselmoinnit ... 20
4.2 Testauksen lähestymistavat ja menetelmät ... 21
4.2.1 Testauksen kvartaalit ... 21
4.2.2 Riskilähtöinen testaus ... 22
4.2.3 Tutkiva testaus ... 23
4.2.4 Testiautomaatio ... 24
4.3 Kehityksen ja testauksen lähestymistavat ... 24
4.3.1 Test Driven Development ... 24
4.3.2 Behaviour Driven Development ... 25

4.3.3 Acceptance Test Driven Development 25
4.4 Ketterät menetelmät 26
4.4.1 Extreme Programming (XP) 26
4.4.2 Scrum 26
5 Erilaiset riskit ohjelmistoprojektien kehittämisessä 28
5.1 Aikataululiset riskit 28
5.2 Kustannuksiin liittyvät riskit 29
5.3 Operationaaliset riski 29
5.4 Tekniset riskit 30
5.5 Projektin ulkopuoliset riskit 31
6 Jatkuva testauksen kehittäminen 33
6.1 Testausstrategian määrittely ja testaussuunnitelman laatiminen 33
6.2 PDCA-malli 35
6.3 Kehitystiimin sisäiseen kommunikointiin panostaminen 36
6.4 Testausprosessin kehittämisen hyödyt 37
7 Tutkimuksen toteutus 39
7.1 Kohdeorganisaation esittely 39
7.2 Tutkimusprosessin kuvaus 39
7.3 Aineiston keruu ja haastateltavat 40
7.4 Tutkimustulosten analyysimenetelmä 42
7.5 Juurisyyanalyysi 43
8 Testausprosessin analysointi kohdeorganisaatiossa 44
8.1 Ketterien menetelmien toteutus 46
8.2 Testausprosessin ja kokonaisuuden hallinta 47
8.3 Kommunikaatio 50
8.4 Juurisyyanalyysin tulokset 51
9 Testausprosessin kehittäminen 54
9.1 Kollektiivien retrospektiivi malli 54
9.2 Ketterä valmentaja 58
10 Johtopäätökset 59
10.1 Tutkimuskysymyksiin vastaaminen 59
10.2 Tutkimuksen rajoitteet 62

10.3 Jatkotutkimusaiheet 63
Lähteet 64

1 Johdanto

Ketteryys ja ketterät menetelmät ovat olleet jo pitkään trendi ohjelmistokehitysalalla. Ohjelmistokehityksen alkuaikoina tiimit seurasivat vesiputousmallin ohjelmistokehitysmetodologiaa, jossa ohjelmistojen testaus suoritettiin kehitysputken loppupäässä, hieman ennen valmiin tuotteen toimitusta asiakkaalle. Vesiputousmallista metodologiaa on pidetty monin paikoin tehottomana prosessin raskauden vuoksi, vaikkakin menetelmä on edelleen monin paikoin käytössä. (Baumgartner et al 2021; Gregory & Crispin 2015, 22–25.)

Ketterän kehityksen aikakausi alkoi 2000-luvun alussa, jolloin ketteriä ohjelmistonkehitysmenetelmien raameja ryhdyttiin laatimaan. Ajatus testaajien sisällyttämisestä kehitysprosessiin alusta lähtien kehittyi. Vesiputousmallisen ohjelmistokehitysprojektin testauksesta poiketen, ketterän kehityksen testaus tapahtuu jokaisen kehityssprinttien aikana. Yksi ketterän testauksen suurimmista eduista on prosessin nopeus ja kustannusten aleneminen tuotteen julkaisuun asti. (Baumgartner et al 2021; Gregory & Crispin 2015, 22–25.)

Mitä ketterä testaus käytössä toteutetaan? Ja miksi laadunvarmistus on tärkeää? Ketterä ohjelmistokehitys mahdollistaa nopeat ja jatkuvat muutokset kehitysprosessin aikana. Ripeät syklit ja muuttuvat määritykset luovat omat haasteensa laadunvarmistukselle. Luoko ketterien menetelmien käyttö muita haasteita testausprosessiin? Ja miten testausprosessi ylipäätään etenee ketterin menetelmien toteutettavissa ohjelmistonkehitysprojekteissa?

Tämän diplomityön tarkoituksena on selvittää, miten toimeksiantajan asiakasyrityksissä testausprosessi etenee ketterin menetelmien toteutettavissa ohjelmistonkehitysprojekteissa sekä selvittää, minkälaisia ovat testausprosessin kohtaamat haasteet ja mahdolliset ongelmat.

1.1 Diplomityön tavoitteet ja tutkimuskysymykset

Diplomityö toteutettiin kohdeyrityksen toimeksiantona. Yksi oleellinen osa kohdeyrityksen palveluliiketoimintaa on laadunvarmistukseen liittyvien palvelujen tuottaminen asiakkaille. Kohdeyrityksen toiveena oli, että diplomityössä tutkittaisiin testausprosessia ja keskityttäisiin mahdollisiin ongelmakohtiin prosessissa.

Tutkimuksen tutkimuskysymykset ovat:

- mitä on ketterä ohjelmistonkehitys ja miten testaus toteutetaan ketterässä ohjelmistokehityksessä?
- mitkä ovat onnistuneen testauksen edellytykset ja haasteet?
- miten testauksen toteutusta voi kehittää?

Tarkoitus on, että diplomityön tuotoksena syntyy yrityksen käyttöön sopiva prosessimalli, jonka käyttöönoton myötä testausprosessia saisi tehostettua ja ongelmakohdat mahdollisesti poistettua.

1.2 Diplomityön rajaukset

Diplomityössä tarkastelen testausprosessin eri vaiheita painottaen ketterien menetelmien käyttöä ohjelmistokehitysprojekteissa. Diplomityön pääpaino on testauksessa, jonka vuoksi tässä työssä asioita ei käsitellä esimerkiksi koodauksen näkökulmasta.

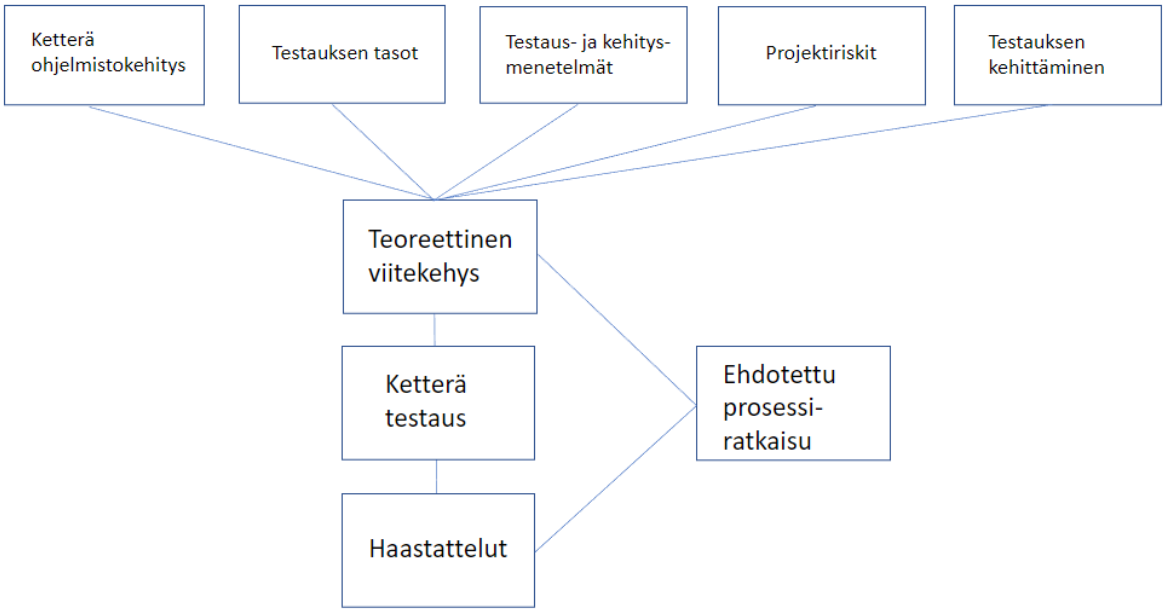
Diplomityön haastattelussa haastateltavat olivat toimeksiantajan työntekijöitä. Haastateltavat oli valittu toimeksiantajan laadunvarmistukseen keskittyvästä liiketoimintayksiköstä. Toimeksiantajan asiakkaita ei haastateltu tutkimusta varten.

Haastattelujen sisällöt ja haastatteluiden perusteella tehdyt analyysit liittyvät vain toimeksiantajaan eivätkä ole verrattavissa suoranaisesti muiden yritysten toimintaan tai toimintatapoihin.

1.3 Teoreettinen viitekehys ja kirjallisuuskatsaus

Tutkielman teoreettinen viitekehys ja teoreettinen tausta ovat nähtävissä kuvassa 1. Teoreettinen viitekehys rakentuu ohjelmistokehityksen testauksen ympärille, johon kuuluu muun muassa testauksen eri tasot ja testausmenetelmät. Tämän lisäksi teoria rakentuu ketterien

menetelmien ideologiasta, ohjelmistokehitysprojektien riskeistä sekä jatkuvan testauksen kehittämisen tavoista. Teoreettisen osuuden tavoitteena on perehtyä tutkimusaiheeseen ja tutustua potentiaalisiin lähestymistapoihin, joita voidaan hyödyntää uuden prosessin rakentamisessa.

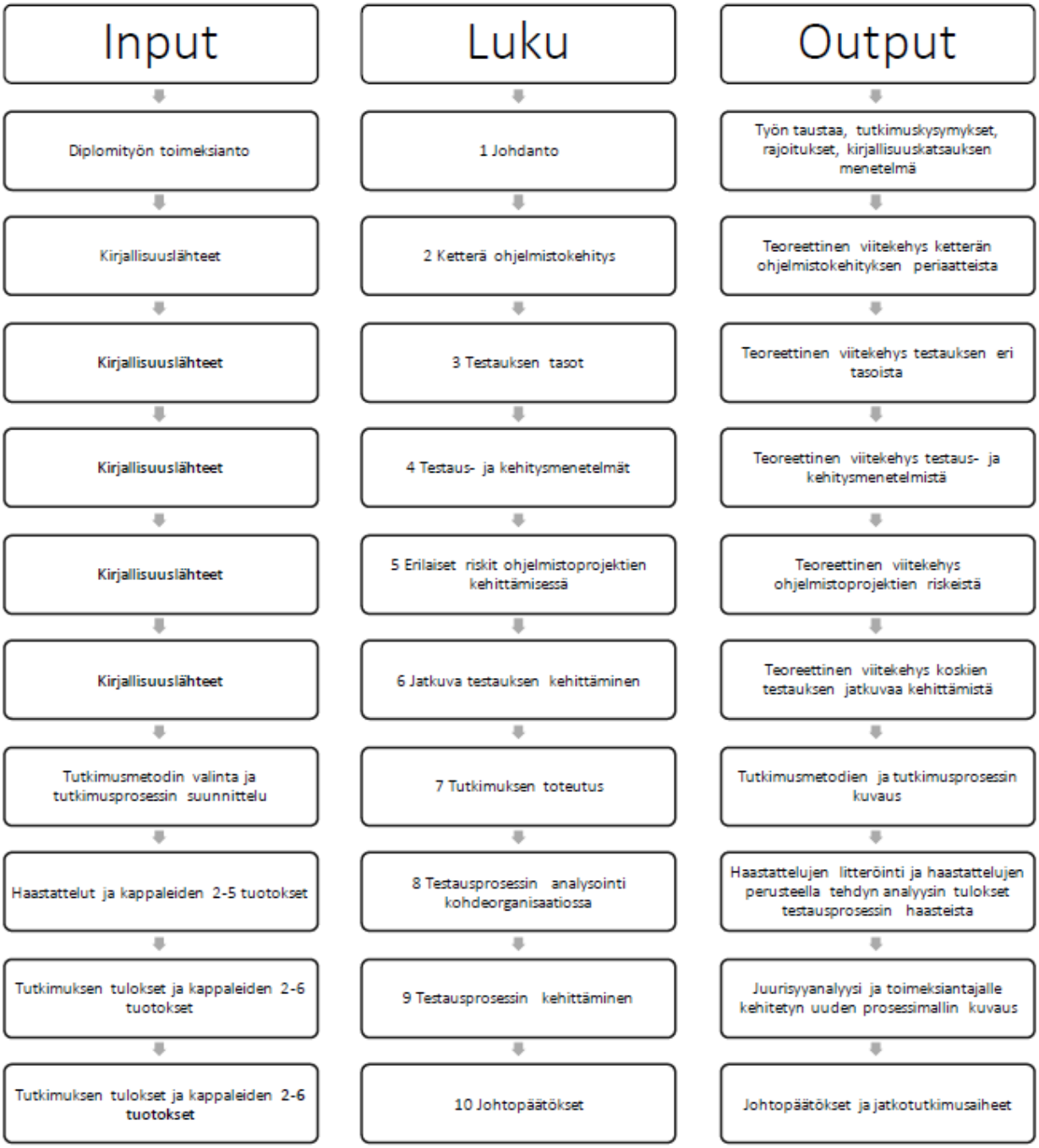


Kuva 1. Teoreettinen viitekehys

Teoreettinen viitekehys on koottu kuvailevan kirjallisuuskatsauksen narratiivisella orientaatiolla. Kuvaileva kirjallisuuskatsaus on yksi yleisimmin käytetty kirjallisuuskatsaustyyppi. Kyseisessä kirjallisuuskatsauksessa käytetyt aineistot ovat laajoja eikä aineiston valintaa rajaa tarkat metodiset säännöt. Myös tutkimuskysymykset voivat olla väljempiä. Narratiivinen orientaatio on metodisesti kevyin kirjallisuuskatsauksen muoto. Sen avulla voidaan antaa laaja kuva käsiteltävästä aiheesta ja samalla pyrkiä luomaan helppolukuinen lopputulos. (Salminen 2011.)

1.4 Diplomityön rakenne

Diplomityö koostuu kymmenestä pääluvusta. Kuvassa 2 on esitettynä työn sisältö jaoteltuna luvuittain input-output –kaavion avulla.



Kuva 2. Diplomityön sisältö input-output –kaaviona.

Ensimmäinen luku on johdanto, jossa esitellään diplomityön taustaa ja tavoitetta, teoreettista viitekehystä sekä diplomityön rakennetta. Työn teoreettinen osuus alkaa toisesta luvusta.

Toinen luku käsittelee ketterää ohjelmistokehitystä ja ketterän kehityksen periaatteita. Lisäksi luvussa verrataan vesiputousmallista ja ketterän kehityksen menetelmiä sekä kerrotaan millaista, on ketterä testaus.

Luvussa kolme käydään läpi testauksen eri tasot ja neljännessä luvussa esitellään erilaisia testaus- ja kehitysmenetelmiä. Viides luku käsittelee erilaisia ohjelmistonkehitysprojektien riskejä. Kuudennessa luvussa käydään läpi erilaisia tapoja toteuttaa jatkuva testauksen kehittäminen.

Seitsemännessä luvussa käydään läpi työn tutkimuksen toteutus, aineiston keruu sekä analyysimenetelmä.

Kahdeksas luku käsittelee tutkimuksen tuloksia. Yhdeksännessä luvussa esitellään tutkimuksen pohjalta luotu kollektiivien retrospektiivi malli. Kymmenes luku käydään läpi johtopäätökset.

2 Ketterä ohjelmistokehitys

Tässä luvussa käydään läpi ketterän ohjelmistokehityksen manifestin sisältö sekä ketterän kehityksen periaatteet. Lisäksi käsitellään ketterien menetelmien ja vesiputousmallin keskeisimmät erot sekä sen, millaista testaus on ketterän menetelmän projektissa.

2.1 Ketterän ohjelmistokehityksen manifesti

Ketterän ohjelmistonkehityksen manifestin laati joukko keveiden ohjelmistokehitysmenetelmien edustajia vuonna 2001. Joukko yhteisiä arvolauseita ja periaatteita tultiin tuntemaan ketterän ohjelmistokehityksen manifestina. (ISTQB 2015, 8–9.)

Itse manifesti sisältää neljä lausetta (ISTQB 2015, 8–9):

> "Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja
>
> Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota
>
> Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja
>
> Muutokseen reagoimista enemmän kuin pitäytymistä suunnitelmassa"

Ketterässä manifestissa painotetaan yhteistyötä ja kanssakäymistä. Tämän vuoksi manifesti esittää, että yllä olevien lauseiden vasemmalla puolella mainittuja asioita olisi syytä arvostaa enemmän kuin lauseiden oikealla puolella mainittuja asioita, vaikkakin ne ovat käsitteellisesti tärkeitä asioita. (Baumgartner, Klonk, Mastnak, Pichler, Seidl & Tanczos 2021; ISTQB 2015, 8–9.)

Ketterässä ohjelmistokehityksessä ihmiskeskeisyys on keskiössä. Kehitystiimien työskentely ympäristössä panostetaan jatkuvaan kommunikointiin ja kanssakäymiseen enemmän kuin työkaluihin ja prosesseihin. Sama periaate näkyy myös asiakasyhteistyössä; tiiviillä yhteistyöllä parannetaan mahdollisuuksia ymmärtää asiakkaan vaatimuksia ja tämän myötä

tuottaa haluttu ohjelmistojärjestelmä ja saavuttaa onnistunut projekti. (Baumgartner, Klonk, Mastnak, Pichler, Seidl & Tanczos 2021; ISTQB 2015, 8–9.)

Ketterissä menetelmissä pyritään usein vähentämään dokumentaation määrää, koska koetaan, että asiakkaan näkökulmasta toimiva lopputuotos on tarkkaa dokumentaatiota tärkeämpää. Myös ohjelmiston ja työtehtävien liian tarkka määrittely koetaan rajoittavana ja aikaa vievänä. Ketterin menetelmin tuotettu ohjelmisto on saatavilla aikaisemmin kehityksen elinkaaressa kuin perinteisessä vesiputousmallisessa projektissa, mikä jouduttaa tuotteen julkaisua markkinoille. Tästä johtuen ketterät ohjelmistokehityksen menetelmät ovat toimivia varsinkin nopeasti muuttuvissa liiketoimintaympäristöissä, joissa ohjelmiston sisäiset ratkaisut ovat epäselviä, tai joissa innovointi on tervetullutta ongelmiksi koetuilla alueilla. (Baumgartner, Klonk, Mastnak, Pichler, Seidl & Tanczos 2021; ISTQB 2015, 8–9.)

Muutokset ovat väistämättömiä erilaisissa ohjelmistoprojekteissa. Ketterässä ohjelmistokehityksessä painotetaan ohjelmistokehitysprojektin aikaisiin muutoksiin reagoimista joustavuuden ennemmin kuin tarkasti noudatettavan suunnitelman kautta. (Baumgartner, Klonk, Mastnak, Pichler, Seidl & Tanczos 2021; ISTQB 2015, 8–9.)

2.2 Ketterän kehityksen periaatteet

Ketterän kehityksen manifestissa on määritelty 12 noudatettavaa perusperiaatetta. Huttunen (2006, 15–16) on suomentanut diplomityössään ne seuraavasti:

“1. Tärkeintä on täyttää asiakkaan vaatimukset jatkuvilla ja riittävän aikaisilla ohjelmistotoimituksilla.

2. Muuttuvat vaatimusmäärittelyt hyväksytään ja otetaan vastaan tervetulleina, myös kehityksen loppuvaiheessa. Ketterät menetelmät valjastavat muutoksen asiakkaan kilpailueduksi.

3. Toimivia ohjelmaversioita toimitetaan säännöllisesti, muutaman viikon tai kuukauden välein, mieluummin useammin kuin harvemmin.

4. Liiketoiminnan ammattilaisten ja ohjelmistokehittäjien on työskenneltävä yhdessä päivittäin koko projektin ajan.

5. Projektit rakennetaan motivoituneiden yksilöiden ympärille. Heille annetaan ympäristö ja tuki, jota he tarvitsevat, sekä luotetaan siihen, että he saavat työn tehtyä.

6. Kaikkein tehokkain tapa tiedonvälityksessä kehitystiimille ja kehitystiimin sisällä on kasvokkain tapahtuva keskustelu.

7. Toimiva ohjelmisto on ensisijainen työn edistymisen mitta.

8. Ketterät menetelmät suosivat kestävää kehitystä. Rahoittajien, kehittäjien ja käyttäjien tulisi kyetä pitämään jatkuvasti yllä tasainen työtahti.

9. Jatkuva huomion kiinnittäminen tekniseen erinomaisuuteen, hyvään rakenteeseen sekä suunnitteluun lisää ketteryyttä.

10. Yksinkertaisuus – tekemättömän työn maksimoinnin taito – on olennaista.

11. Parhaat rakenteet, vaatimukset ja suunnitelmat nousevat itseorganisoituvista tiimeistä.

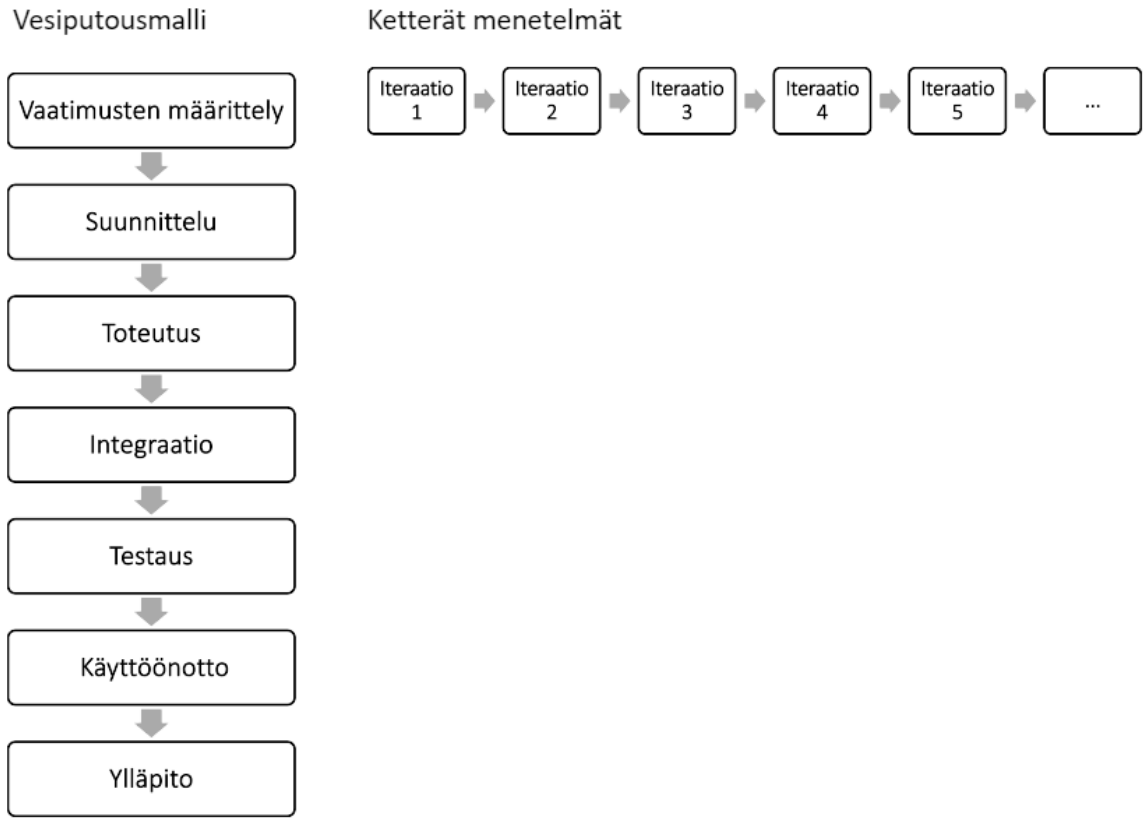
12. Tasaisin väliajoin tiimi arvioi, miten voisi tulla entistä tehokkaammaksi, ja kehittää toimintaansa sen mukaisesti. “

Ketteriä menetelmiä ja lähestymistapoja on olemassa useita. Jokainen niistä toteuttaa ketterän ohjelmistokehityksen manifestin periaatteita ja arvoja hiukan eri tavoin. (ISTQB 2015, 11–13.) Esimerkkinä ketteristä menetelmistä mainittakoon Scrum ja Extreme programming, joiden ideologiaa käydään tarkemmin läpi luvussa neljä, Testaus- ja kehittämismenetelmät.

2.3 Vesiputousmalli vs ketterät menetelmät

Ohjelmistokehityksen perinteisestä menetelmästä puhuttaessa tarkoitetaan yleensä vesiputousmallia. Vesiputousmallissa jokainen kehitysvaihe määritellään erikseen ja vaiheet suoritetaan järjestyksessä niin sanotusti ylhäältä alas (kuva 3). Käytännössä vesiputousmallin vaiheita saatetaan projektista riippuen suorittaa lineaarisesta etenemisestä poiketen myös päällekkäin, koska mallin seuraava vaihe voi paljastaa ongelmia mallin edellisestä vaiheesta

aiheuttaen näin muutoksia aiemmin tehtyihin suunnitelmiin ja määrityksiin. Ketterässä kehityksessä kehitys etenee ripeässä tahdissa lyhyt iteraatio kerrallaan. Yksi iteraatio voi sisältää niin määritystä, toteutusta kuin testausta. Iteraatio päättyy toiminnallisuuden julkaisemiseen. (Baumgartner et al 2021; Haikala ja Märijärvi, 2004.).



Kuva 3. Haikala ja Märijärvi (2004)

Ketterät ohjelmistokehityksen menetelmät yleistyivät 2000-luvulla, jonka myötä vesiputousmallin käyttö väheni. Roimanin (2017) mukaan asiakasvaatimusten jatkuva muuttuminen sekä asiakkaiden tahto nähdä nopeasti tuotoksia on vaikuttanut monessa tapauksessa vesiputousmallista ketterään menetelmään vaihtamisen. Hooryn ja Bottoffin (2022) mukaan molemmissa menetelmissä on kuitenkin omat hyvät ja huonot puolensa, jonka vuoksi menetelmän valinta olisi syytä ratkaista projektikohtaisesti.

Alla olevassa taulukossa on koottuna keskeisempiä eroja vesiputousmallin ja ketterien menetelmien välillä. Taulukko on koostettu Hoory ja Bottoffin (2022) tekemästä aineistosta, jossa vertaillaan vesiputousmallin ja ketterien menetelmien eroja.

Taulukko 1. Vesiputousmallin ja ketterien menetelmien oleellisimmat erot

	Vesiputousmalli	Ketterät menetelmät
Aikajana	Vesiputouksella on kiinteä aikajana; projektin alku ja loppu on kartoitettu alusta alkaen. Sopii projekteihin, joilla on tarkka määritelty lopputavoite.	Kiinteän aikajanan sijaan aikataulu mukautuu projektin edetessä.
Asiakasyhteistyö	Projektin kulku on hahmoteltu alusta alkaen, joten asiakaspalautteen sisällyttäminen ei ole jatkuva osa prosessia.	Asiakkaiden mukaan ottaminen projektin jokaisessa vaiheessa on oleellista. Asiakkaiden odotetaan osallistuvan ja antavan palautetta ohjelmistokehitystiimille projektin eri vaiheiden edetessä.
Joustavuus	Vesiputousmalli ei ole yhtä joustava kuin ketterät menetelmät, koska jokainen vaihe on saatava kokonaan päätökseen ennen kuin siirrytään seuraavaan vaiheeseen. Menetelmä on ihanteellinen projekteille, joissa on selkeä näkemys siitä, miten edetään alusta loppuun.	Ketterässä menetelmässä painotetaan joustavuutta. Muutokset ovat tervetulleita ja uutta tietoa voi sisällyttää myös projektin myöhemmässä vaiheessa.
Testaus	Suoritetaan omana prosessinaan toteutuksen ja integraation jälkeen. Projektin loppuvaiheessa toteutettu testaus voi olla aikataulullisesti haastavaa. Kriittisten virheiden löytyminen projektin loppuvaiheisessa testauksessa voi aiheuttaa suuria riskejä projektille.	Voi olla haasteellinen toteuttaa. Tapahtuu osissa koko projektin läpi. Mahdollisuus tiivistää yhteistyötä kehittäjien ja tuoteomistajan kanssa. Antaa testaajille mahdollisuuden olla mukana projektin alusta asti. Testiautomaation hyödyntäminen regressiotestauksessa korostuu.
Budjetti	Valmiiksi määritelty ja päätetty. Vesiputousmallin metodologiaa käyttävien hankkeiden budjetti on yleensä kiinteä.	Joustava. Mahdolliset suunnanmuutokset on huomioitava budjettia suunniteltaessa.

2.4 Ketterä testaus

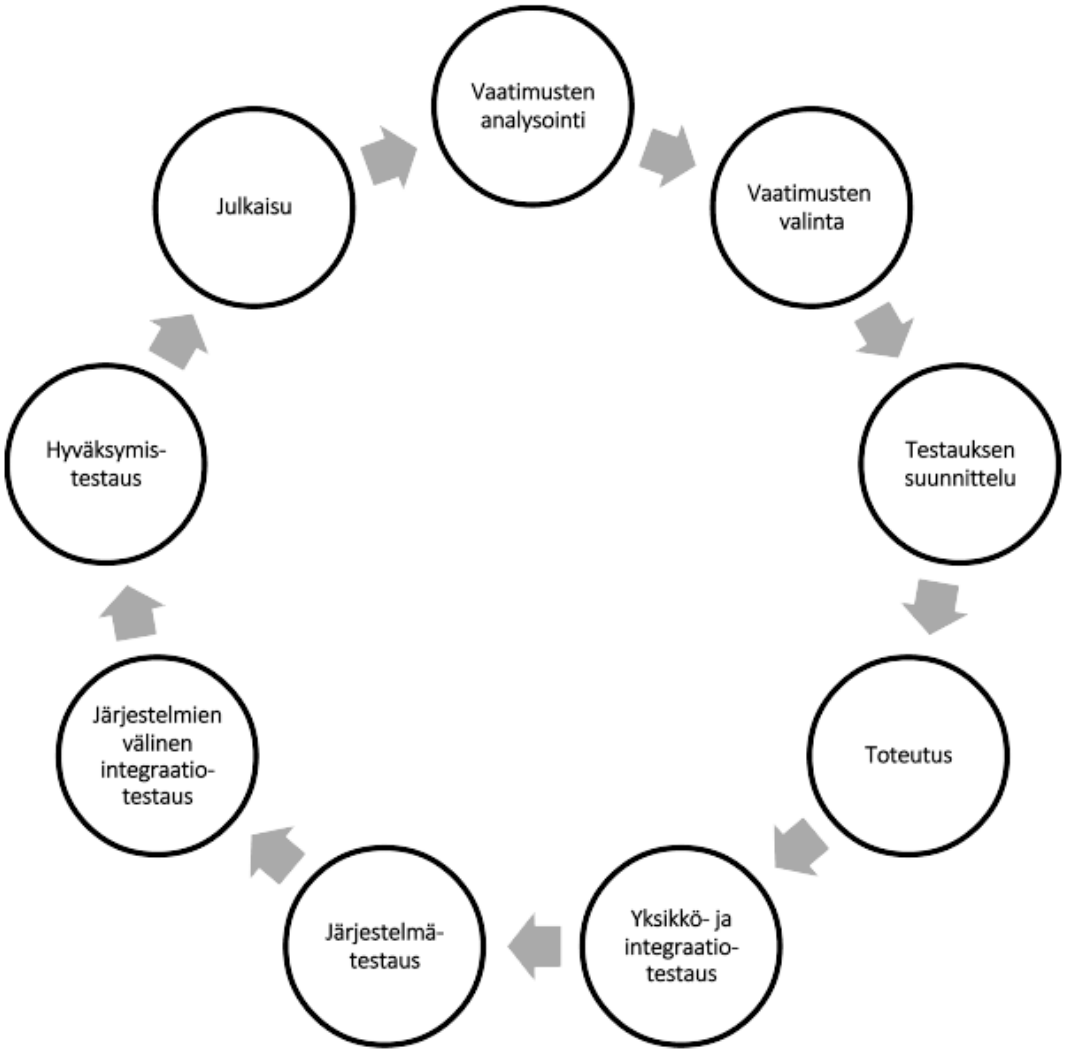
Vesiputousmallisessa projektissa testaus suoritetaan normaalisti projektin tai toimituserän loppuvaiheessa, kun taas ketterän kehityksen testaus suoritetaan osissa koko projektin ajan. Testauksen jatkuva suorittaminen mahdollistaa palautteen antamisen kehitystiimille jokaisen sprintin tai iteraation jälkeen. Tällä tavoin virheisiin ja korjauksiin voidaan reagoida nopeastikin verrattuna vesiputousmalliin, jossa testauksessa havaitut asiat tulevat ilmi vasta projektin lopussa. (Gregory & Crispin 2015, 51–56.)

Vesiputousmallissa vaatimukset on määritelty projektin alussa, ja muutospyyntöjä huomioimatta ne pysyvät muuttumattomina. Ketterissä ohjelmistokehitysprojekteissa vaatimukset voivat muuttua toteutuksen edetessä. Molemmat menetelmät luovat omat haasteensa testaukseen; vesiputousmallissa testauksen ja testitapausten suunnittelu on muuttumattomien määritelmien vuoksi helpompaa, mutta testauksen toteutus on aikaa vievää ja riskipitoisempaa, koska sen toteutus painottuu projektin loppuun. Ketterässä ohjelmistokehityksessä testauksen suunnittelu ja toteutus voi olla haasteellista mahdollisesti muuttuvien määritysten vuoksi, mutta testausta voidaan suorittaa jatkuvasti toteutuksen vierellä. (Gregory & Crispin 2015, 51–56.)

Vesiputousmallisessa projektissa testauksen hoitaa perinteisesti erillinen tiimi, jota johtaa testauspäällikkö. Erillisellä tiimillä pyritään muun muassa saavuttamaan riippumattomuutta ja puolueettomuutta. Testaustiimissä voi olla osana niin testauksen ammattilaisia kuin liiketoiminnan edustajia. Ketterässä testauksessa testaajat ovat osa kehitystiimiä ja erillistä testauspäällikköä ei ole. Testaajien oleminen osa kehitystiimiä lisää kehitystiimin sisäistä kommunikaatiota ja yhteistyötä. Testaajien ollessa mukana kehitystiimissä koko ohjelmistokehitysprosessin ajan, he voivat osallistua myös vaatimusten analysointiin ja työmäärien arviointeihin. (Gregory & Crispin 2015, 51–56.)

Koska ketterässä ohjelmistokehitysprojektissa testaus on jatkuvaa, tulisi prosessi tehdä joustavaksi ja muuttuvia vaatimuksia ja määrityksiä vastaanottavaksi. Ketterässä testauksessa, kuin myös ketterässä kehityksessä, korostetaan tehokkuutta. Tehokkuutta voidaan optimoida varaamalla testauksen suunnittelulle ja toteutukselle aikaa sekä analysoimalla testausprosessin etenemistä ja tekemällä tarvittavia parannuksia. (Perry 2006, 817–833.)

Kuvassa 4 on nähtävillä ketterän testauksen prosessi testaajan näkökulmasta Perryn (2006) malli mukaillen. Yhden iteraation testausprosessi alkaa vaatimusten analysoinnilla ja valinnalla. Tämän jälkeen suunnitellaan testaus toteutettavaan toiminnallisuuteen. Toteutuksen valmistumisen jälkeen suoritetaan manuaalista ja automatisoitua testausta yhdistelemällä yksikkö-, integraatio-, järjestelmä- ja hyväksymistestaus. Tämän jälkeen toteutus voidaan julkaista. Julkaisun jälkeen käynnistyy uusi iteraatio ja testausprosessi alkaa alusta.



Kuva 4. Ketterän kehityksen testausprosessi (Perry 2006).

3 Testauksen tasot

Tässä luvussa käydään läpi testauksen eri tasot. Testauksen tason yhteydessä kerrotaan mihin järjestelmän osaan tai toiminnallisuuteen kyseisen tason testaus kohdistuu ja minkä tyyppisiä virheitä kyseisellä tasolla tyypillisesti löytyy.

3.1 Yksikkötestaus

Yksikkötestauksen pääasialliset tavoitteet ovat riskien pienentäminen, virheiden löytyminen sekä niiden syntymisen estäminen sekä varmistus siitä, että järjestelmän toiminnallinen ja ei-toiminnallinen käyttäytyminen on vaatimusten ja määritysten mukaista. Lisäksi yksikkötestauksella kasvatetaan luottamusta komponenttien laatuun. Yksikkötestaus kohdistuu yleisimmin komponentteihin, koodiin ja tietorakenteisiin, luokkiin sekä tietokantamoduuleihin. (ISTQB 2018, 27; Watkins & Mills 2011, 48–54.)

Ketterässä kehityksessä kehitteillä oleva järjestelmä muuttuu jatkuvasti kehityksen myötä. Automatisoidulla regressiotestauksella voidaan varmistaa, että koodin muutokset eivät ole rikkoneet aiemmin toteutettua toiminnallisuutta. Tyypillisimmät yksikkötestauksessa löytyvät viat tai häiriöt liittyvät usein väärään toiminnallisuuteen, tietovirtaongelmiin tai virheelliseen logiikkaan ja koodiin. (ISTQB 2018, 27; Watkins & Mills 2011, 48–54.)

3.2 Integraatiotestaus

Integraatiotestauksessa keskitytään komponenttien väliseen vuorovaikutukseen ja pyritään löytämään virheitä sekä estää virheiden pääsy seuraaville testitasoille. Komponenttien integrointitestaus suoritetaan yksikkötestauksen jälkeen normaalisti automatisoidusti. Järjestelmäintegraatiotestauksessa keskitytään sen sijaan järjestelmien, mikropalveluiden ja pakettien rajapintoihin ja vuorovaikutuksiin. Järjestelmäintegraatiotestausta voidaan suorittaa samanaikaisesti järjestelmätestauksen kanssa tai sen jälkeen. (ISTQB 2018, 27; Watkins & Mills 2011, 54–61.)

Tyypillisimmät integraatiotestauksen kohteet ovat alijärjestelmät, infrastruktuuri, rajapinnat, tietokannat, ohjelmointirajapinnat sekä mikropalvelut. Komponentti-integraatiotestauksessa löytyviä vikoja tai häiriötä voivat olla virheelliset sekvenssit tai rajapintakutsujen ajoitus, rajapintojen epäyhteensopivuus sekä häiriöt komponenteissa. Järjestelmäintegraatiotestauksen tyypillisimmät viat liittyvät viestirakenteiden epäjohdonmukaisuuksiin järjestelmien välillä, rajapintojen epäyhteensopivuuteen sekä häiriöihin järjestelmien välisessä viestinnässä. (ISTQB 2018, 27; Watkins & Mills 2011, 54–61.)

3.3 Järjestelmätestaus

Järjestelmätestauksen tavoitteena on testata kokonaista järjestelmää tai ohjelmaa tuotannonkaltaisesti. Testauksessa käydään läpi vaatimusten mukaisia ei-toiminnallisia vaatimuksia sekä kokonaisia toimintaketjuja alusta loppuun. Järjestelmätestauksessa pyritään usein käyttämään tuotantoa vastaavaa testidataa, jotta testaus muistuttaisi mahdollisimman paljon tuotantoa. Myös testiympäristön tulisi olla tuotannonomainen. (Watkins & Mills 2011, 61–69.)

Kuten aiemmilla testaustasoilla, myös järjestelmätestauksen aikana tarkistetaan regressiotestauksen avulla, että muutokset eivät ole rikkoneet olemassa olevia toiminnallisuuksia. Järjestelmätestauksella voidaan myös varmistaa, että oikeudelliset tai muuhun säännöksiin perustuvat vaatimukset täyttyvät. (ISTQB 2018, 28–29.)

Järjestelmätestauksen kohteena on tyypillisesti muun muassa sovellukset, laitteisto, käyttöjärjestelmät sekä järjestelmän kokoonpano ja siihen liittyvä aineisto. Testauksen aikana löydettävät viat ja häiriöt voivat liittyä esimerkiksi virheellisiin laskutoimituksiin, odottamattomaan toiminnalliseen tai ei-toiminnalliseen käyttäytymiseen tai tietovuotoihin. (ISTQB 2018, 28–29.)

3.4 Hyväksymistestaus

Hyväksymistestauksen tarkoituksena on arvioida, onko järjestelmä valmis julkaisuun ja soveltuuko se loppukäyttäjien käytettäväksi. Hyväksymistestauksen suorittaakin yleensä loppukäyttäjän edustaja. Testaus tulisi suorittaa mahdollisimman tuotannonkaltaisessa ympäristössä ja testitapausten muistuttaa tuotannossa suoritettavia työtehtäviä.

Hyväksymistestauksen kohteena on testattavan järjestelmän lisäksi sen kokoonpano, liiketoimintaprosessit integroidussa järjestelmässä, lomakkeet, raportit sekä ylläpitoprosessit. (ISTQB 2018, 32; Watkins ja Mills, 2011, 75–83.)

Hyväksymistestauksen aikana ei olisi enää suotavaa löytyä virheitä ja mikäli niitä havaitaan, voidaan sitä joissakin tapauksissa pitää suurena projektiriskinä. Hyväksymistestauksessa löydettyjen pienempien, ei kriittisten virheiden korjaus, voidaan siirtää järjestelmän ylläpidon vastuulle. (ISTQB 2018, 32.)

Liiketoiminnallisesti vakavien virheiden löytymistä vasta hyväksymistestauksen aikana voidaan välttää ottamalla liiketoiminnan edustajia mukaan jo aiemmissa testaustasoissa. Tällöin esimerkiksi mahdolliset määrittelyjen väärinymmärrykset ja liiketoimintaprosessien sisältämät virheet tulevat aiemmin kehitystiimin tietoon. (ISTQB 2018, 32.)

Tyypillisimmät hyväksymistestauksen aikana todetut viat ja häiriöt liittyvät liiketoimintasääntöjen väärin toteuttamiseen tai käyttäjien vaatimusten toteuttamattomuuteen. Myös sopimuksellisten tai säädöksellisten vaatimusten täyttämättömyys sekä esimerkiksi tietoturvahaavoittuvuudet ja riittämätön suoritustehokkuus voivat olla hyväksymistestauksen aikana havaituista ongelmista. (ISTQB 2018, 32.)

4 Testaus- ja kehitysmenetelmät

Tässä luvussa tutustumaan erilaisiin testaus- ja kehitysmenetelmiin. Eri menetelmien kohdalla käydään läpi niiden perusperiaate, käyttötarkoitus ja hyödyt.

Luvussa esitellään erilaisia lähestymistapoja testauksen suorittamiseen sekä kehitysmenetelmiä, joissa testaaja ja ohjelmoija työskentelevät yhdessä. Lähestymistapoja voidaan käyttää rinnakkain, lyhyt kestoisesti tai koko projektin läpi. Poikkeuksena ketterät menetelmät, joista esitellään Extreme Programming ja Scrum.

4.1 Staattinen testaus

4.1.1 Katselmoinnit

Järjestelmän katselmoinneilla tarkoitetaan järjestelmän sisäisen ja ulkoisen laadun sekä standardien, määräysten ja suunnitelmien noudattamisen tarkistamista ilman, että koodia ajetaan tai muutetaan. Prosessissa on osallisena joko yrityksen sisäinen tiimi tai riippumaton tarkastaja. Katselmointien tarkoituksena on parantaa järjestelmän laatua sekä paljastaa heikkouksia ja puutteita. Katselmointi voi koskea niin määritysdokumentteja ja käyttötapauksia, testitapauksia kuin itse järjestelmän koodiakin. (Baumgartner et al 2021, 26; Mahfuz 2021, 52.)

Ketterässä ohjelmistokehityksessä katselmoinnit ovat usein osana testaus- ja kehitysprosessia. Scrum-menetelmään on sisäänrakennettu useampia katselmointeja, normaalisti ennen ja jälkeen iteraation. Scrummissa näitä tilaisuuksia kutsutaan sprint-demoksi ja retrospektiiviksi. (Baumgartner et al 2021, 26–28; Mahfuz 2021, 52.)

Oikein suoritetuilla katselmoinneilla voidaan ehkäistä virheiden päätymistä koodiin tai väärinymmärryksiä kehitystiimin sisällä. Katselmoinnilla voidaan säästää kustannuksissa varsinkin, jos epäkohtiin on puututtu ennen varsinaisen koodin kirjoittamista. Liiketoiminnan edustajia on suotavaa hyödyntää esimerkiksi käyttötapausten katselmoinnissa, jolloin liiketoimintaprosessien tuotannonkaltaisuus tulee varmistettua. (Baumgartner et al 2021, 26–28; Mahfuz 2021, 52.)

4.2 Testauksen lähestymistavat ja menetelmät

4.2.1 Testauksen kvartaalit

Testauksen kvartaali on testausmenetelmä, jossa ketterän ohjelmistokehityksen testivaiheita kuvataan eri näkökulmien mukaan. Kvartaalissa on nimensä mukaisesti neljä osuutta. Kuvassa 5 on havainnollistettu Gregoryn ja Crispin (2015) testauskvartaalia mukaillen jokaisen kvartaalin sisältö sekä siitä käytetty näkökulma.

Liiketoiminnallinen näkökulma

Tiimin tukeminen

Toiminnallinen testaus Esimerkit Tarinoiden testaus Suoritetaan manuaalisesti & automatisoidusti **K2**	Tutkiva testaus Skenaariot Käytettävyys Hyväksymistestaus Suoritetaan manuaalisesti **K3**
K1 Yksikkötestaus Järjestelmätestaus Suoritetaan automatisoidusti	**K4** Suorituskykytestaus Tietoturvatestaus Suoritetaan tehtävään tarkoitetuilla työkaluilla

Tuotteellinen näkökulma

Teknologinen näkökulma

Kuva 5. Testauskvartaalitaulukko Gregoryn ja Crispin (2015) taulukkoa mukaillen.

Testauskvartaalin ensimmäinen neljännes on testauksen näkökulmasta tiimiä tukeva ja siinä painotetaan teknologista näkökulmaa. Testit ovat yksikkö- ja integraatiotestejä. Testit ovat teknisesti suuntautuneita ja ne suoritetaan yleensä automatisoidusti. Testauksen tarkoituksena on varmistaa, että yksittäisten komponenttien toiminta on vaatimusten mukainen ja komponentit voidaan integroida järjestelmään. Testauksessa voidaan yhtä hyvin tarkistaa myös yksittäisten toiminnallisuuksien toimivuus. Testauskvartaalin ensimmäisellä

neljänneksellä testitapauksia tulisi olla määrällisesti eniten verrattuna muihin kvartaaleihin. (Gregory & Crispin 2015, 122–123.)

Toisella neljänneksellä testaus on edelleen tiimiä tukevaa, mutta mukaan otetaan liiketoiminnallinen näkökulma. Testauksen tarkoituksena on varmistaa, että kehitystiimi työstää liiketoiminnan vaatimuksia vastaavaa ohjelmistoa. Testitapaukset painottuvat liiketoimintaprosesseihin. Suurin osa testitapauksista oli hyvä automatisoida regressiotestauksen helpottamiseksi sekä nopean palautteen aikaansaamiseksi. (Gregory & Crispin 2015, 123–124.)

Testauskvartaalin kolmannessa neljänneksessä testaus on edelleen liiketoimintalähtöistä, mutta testitapauksissa painotetaan myös tuotteellista näkökulmaa. Tämä tarkoittaa, että testauksen tarkoituksena on varmistaa ohjelmiston soveltuvuus käyttäjän tarpeisiin. Testit suoritetaan manuaalisesti ja ne luokitellaan hyväksymistesteiksi. Kolmannessa kvartaalissa voidaan suorittaa myös käytettävyystestausta. (Gregory & Crispin 2015, 125.)

Kvartaalin viimeisessä neljänneksessä keskitytään tuotteeseen ja teknologiaan. Tähän osuuteen voi sisältyä esimerkiksi suorituskyky- ja tietoturvatestausta. Tämän kaltaisiin testauskokonaisuuksiin käytetään yleensä erillisiä, kyseistä testausta varten suunniteltua työkaluja ja testauksen suorittaa aihealueeseen perehtynyt testausasiantuntija. Testauskokonaisuutta helpottaa, mikäli ei-toiminnalliset vaatimukset on määritelty huolellisesti ja tarkalla tasolla. (Gregory & Crispin 2015, 126.)

4.2.2 Riskilähtöinen testaus

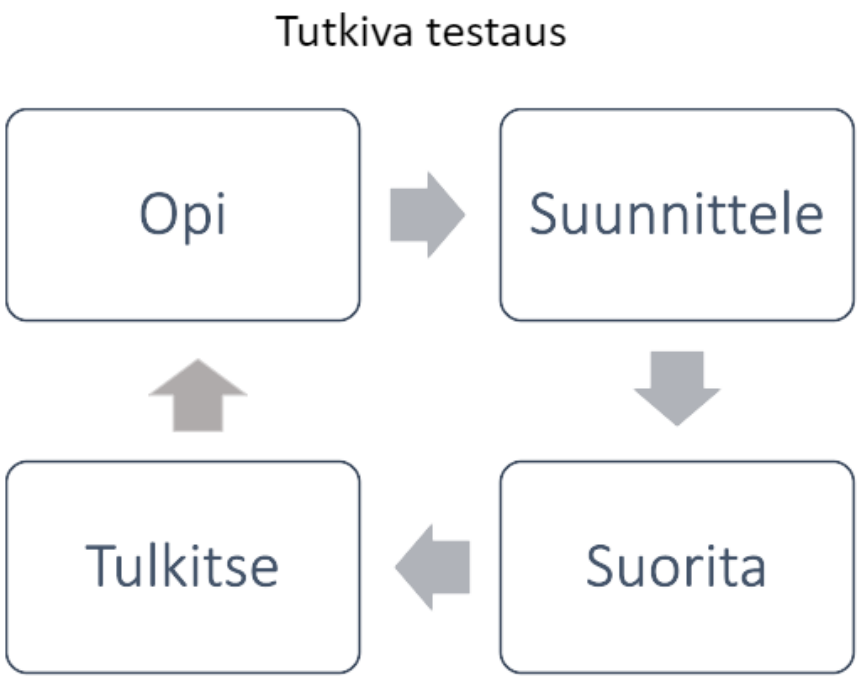
Riskilähtöinen testaus perustuu mahdollisten riskien tunnistamiseen ja niiden analysointiin. Riskien analysointi on tuoteomistajan vastuulla ja tuoteomistajan on jaettava tietoa riskeistä sekä toiminnallisuuden arvoista kehitystiimille. Analysoinnin perusteella päätetään testauksen kohde, testauksen laajuus sekä tarkemmat yksityiskohdat testauksen etenemisestä. Riskilähtöisen testauksen ideana on, että liiketoiminnallisesti kriittiset tai potentiaalisesti suurimmat vaarat aiheuttavat kokonaisuudet testataan ensin. (Baumgartner et al 2021, 126–129; Black 2014, 84–85.)

Tuoteriskejä voivat olla esimerkiksi virhe toiminnallisuudessa, toiminnallisuuden puuttuminen tai vaatimusten vastainen toiminta. Myös tietoturvassa, tiedon eheydessä tai

suorituskyvyssä voi olla puutteita. Laatimalla vaatimukset tarkasti ja ymmärrettävästi vähennetään tuoteriskejä. (Hambling 2015, 143–144.)

4.2.3 Tutkiva testaus

Tutkiva testaus on testauksen lähestymistapa, jossa painotetaan oppimista, suunnittelua ja suorittamista. Tutkivaa testausta voidaan suorittaa ryhmässä tai yhden testaajan voimin. Kuvassa 6 on kuvattuna Kinsbrunerin (2018) mallia mukaillen tutkivan testauksen prosessi. (Kinsbruner 2018, 117–121.)



Kuva 6. Tutkivan testauksen prosessi Kinsbrunerin mallia (2018) mukaillen.

Tutkiva testaus on manuaalisesti suoritettava lähestymistapa testaukseen, jossa testaajan tietämys ja kokemus korostuvat. Testaus on aikarajoitteista ja suunniteltua, vaikkakin tarkasti laadituista testitapauksista poikkeamista suositellaankin. Tutkiva testaus on tehokas tapa aloittaa testaus ja tutustua kehitteillä olevaan järjestelmään. (Baumgartner et al 2021, 129–130.)

Ominaista tutkivalle testaukselle on samanaikaisesti järjestelmän perusolemuksen oppiminen ja testauksen suunnittelu. Vaikka tutkivassa testauksessa voidaan poiketa valmiiksi laadituista testitapauksista, on tutkivan testauksen sisältö silti kirjattava ylös. Tähän voidaan käyttää apuna esimerkiksi toimintaa nauhoittavaa työkalua. (Baumgartner et al 2021, 129–130.)

4.2.4 Testiautomaatio

Testiautomaatiolla tarkoitetaan testitapausten ajamista automaattisesti testaustyökalua käyttäen. Testiautomaation käyttöönotto vaatii hyvää testausosaamista, mutta säästää resursseissa ja kustannuksissa. Joissakin tapauksissa automatisoidun testauksen käyttöönotolla on säästetty 80 % testauksen kustannuksista. Kustannusten vähenemisen lisäksi testiautomaation käyttöönoton hyötyjä ovat muun muassa lisääntynyt tarkkuus sekä nopeampi testauksen läpivienti. Testauksen automatisointi helpottaa myös usein toistettavan regressiotestauksen suorittamista. (Gharhai, 2020; Pohjalainen 2003.)

Käytännössä testitapauksen automatisointi tarkoittaa manuaalisen testin simulointia. Automatisoitaviksi testeiksi sopii hyvin usein samalaisena toistuvat asiat. Tällä tavoin testaajien resursseja voidaan painottaa esimerkiksi liiketoiminnallisesti kriittisempien osien manuaaliseen testaukseen. Yhtenä testiautomaation haasteena onkin prosessin muokkaaminen tehokkaaksi ja tuottavaksi. (Gharhai, 2020; Pohjalainen 2003.)

4.3 Kehityksen ja testauksen lähestymistavat

4.3.1 Test Driven Development

TDD (Test Driven Development) eli testivetoinen kehittäminen painottaa nimensä mukaisesti testausta. TDD:tä voidaan käyttää ketterissä menetelmissä. Menetelmän ideana on, että samanaikaisesti kirjoitetaan sekä koodia että testitapauksia. Käytännössä tämä tarkoittaa, että uutta toiminnallisuutta lisättäessä kirjoitetaan ensin testitapaus ja testattava koodi.

Tämän jälkeen laaditaan ohjelmiston koodin muutokset ja ajetaan testitapaus onnistuneesti läpi. Testin mentyä onnistuneesti läpi, poistetaan koodin mahdolliset päällekkäisyydet, jonka jälkeen varmistetaan, että testit menevät edelleen onnistuneesti läpi. Edellisen kaltaista toimintaa toistetaan myös seuraavien uusien toiminnallisuuksien kohdalla. On tärkeää huomioida, että myös vanhat testit voidaan ajaa onnistuneesti läpi koodimuutosten jälkeen, koska vanhojen testin läpimeno helpottaa regressiotestausta. (Tuomi-Heinonen 2008, 23–24.)

4.3.2 Behaviour Driven Development

BDD (Behaviour Driven Development) on TDD:n laajennus, joka painottuu liiketoiminnan tavoitteisiin. BDD:ssä testaus perustuu liiketoimintaprosesseihin, loppukäyttäjän toiminnan ja käyttäytymisen ymmärtämiseen sekä liiketoiminnallisesti tärkeisiin toiminnallisuuksiin. Liiketoiminnallisen toiminnallisuuden hyväksymiskriteerit ovat myös tärkeässä osassa BDD-testauksessa. BDD:n oikeaoppinen käyttö voi vähentää ohjelmistonkehityksessä muodostuvia kustannuksia, sillä työpanos painottuu liiketoiminnalle arvoa tuottaviin toiminnallisuuksiin. Menetelmällä on mahdollista kasvattaa kehitystiimin ymmärrystä liiketoiminnallisista vaatimuksia, joka itsessään voi edesauttaa virheiden syntymistä kehitystyössä (Irshad et al 2022, 16; Smart 2015, 15–19; 28–29.)

4.3.3 Acceptance Test Driven Development

ATDD-menetelmää (Acceptance Test Driven Development) käytetään todellisten liiketoiminnallisten käyttötilanteiden ja –tapausten kuvaamiseen ja näitä hyödyntämällä ohjelmiston määrittelemiseen. Ohjelmiston vaatimukset on määriteltävä mahdollisimman tarkalla tasolla, jotta mahdolliset virheet huomattaisiin jo alkuvaiheessa. ATDD-menetelmässä hyväksymistestit kirjoitetaan ennen ohjelmiston koodin valmistumista. Koodin valmistuttua testit ajetaan läpi ja mikäli testit menevät onnistuneesti läpi, on ominaisuus tehty määritysten mukaisesti. ATDD-menetelmä eroaa TDD-menetelmästä liiketoiminnalliseen näkökulmaan painottamisessa, kun taas TDD painottuu enemmän ohjelmiston tekniseen puoleen. (Tuomi-Heinonen 2008, 23–24.)

4.4 Ketterät menetelmät

4.4.1 Extreme Programming (XP)

XP eli Extreme Programming on ketterä ohjelmistokehityksen lähestymistapa, jossa periaatteita ja ohjelmistokehityksen käytäntöjä ohjaa viisi arvoa; kunnioitus, rohkeus, kommunikointi, yksinkertaisuus ja palaute. Edellä mainittujen lisäksi on olemassa myös täydentäviä arvoja kuten vastuullisuus, moninaisuus, reflektio sekä laatu. (ISTQB 2015, 11–13.)

XP sisältää pääkäytänteet, joiden ympärille menetelmän käyttö perustuu:

- pariohjelmointi
- informatiivinen työtila ja verkkaisuus
- viikoittainen ja neljännesvuosittainen sykli
- kymmenen minuutin koonti
- jatkuva integraatio
- tarinat, testilähtöinen ohjelmointi ja vaiheittainen suunnittelu. (ISTQB 2015, 11–13).

Monet nykyisistä käytössä olevista ketteristä menetelmiä ovat ottaneet vaikutteita XP:n käytänteistä, periaatteista ja arvoista. Esimerkiksi Scrummissa tiimien toimintaan sisältyy usein XP:n käytäntöjä. (ISTQB 2015, 11–13.)

4.4.2 Scrum

Scrum on ketterä menetelmä, jossa ohjelmiston kehitys tapahtuu iteraatioissa eli printeissä. Kukin sprintti kestää 1–4 viikkoa. Sprinteissä toteutettavat ominaisuudet valitaan koko projektin tavoitteita ja toiminnallisuuksia sisältävältä listalta eli ´product backlogilta´. Tavoitteiden ja toiminnallisuuksien lisäksi ´product backlog´ sisältää tavoitteiden priorisoinnit ja työtehtävien arvioidut työmäärä. (Todaro 2019, 62.)

Scrummille on ominaista päivittäiset lyhyet tilannepalaverit, sprintin lopussa järjestettävät toteutettujen toiminnallisuuksien esittelytilaisuudet sekä retrospektiivit, joissa tiimin kesken

analysoidaan sprintin onnistumista sekä omaa toimintaa pyrkien löytämään parannuskohteita seuraavaa sprinttiä ajatellen. (Todaro 2019, 62.)

Kehitystiimi toimii yhteisvastuullisesti ja itseorganisoituneesti. Tiimi vastaa yhdessä siitä, että toteutus on valmis ja testattu, integroitu sekä dokumentoitu sprintin loppuun mennessä. Toisin kuin XP:ssä, Scrummissa ei ole ennalta määrättyjä ohjelmistokehityksen tekniikoita eikä se ota kantaa, miten testaus tulisi suorittaa Scrum-projektissa. Näin ollen tiimin on itse päätettävä ja sovittava projektissa käytettävät tekniikat. (ISTQB 2015, 11–13; Todaro 2019, 62.)

Kehitystiimin lisäksi Scrummissa on kaksi muuta oleellista roolia; ´scrum master´ ja tuoteomistaja. Scrum masterin tehtävänä on helpottaa projektin etenemistä ja tiimin työtehtävien toteutumista poistamalla mahdollisia haittatekijöitä. Lisäksi hänen tehtävänänsä on noudattaa ja kehittää prosessimallia. Tuotteen omistajan eli ´product ownerin´ työtehtävänä on koko projektin hallinnointi sekä vastuu siitä, että oleellisimmat ja kriittisimmät asiat toteutetaan. (Todaro 2019, 62.)

5 Erilaiset riskit ohjelmistoprojektien kehittämisessä

Kaikkiin hankkeisiin liittyy luontaisia riskejä, jotka voivat heikentää osaa tai kaikkia projektin tavoitteita. Näistä riskeistä johtuen projekti voi viivästyä, kustannukset voivat ylittyä, lopullinen tuote voi olla huonolaatuista tai se ei vastaa asiakkaan toiveita. Ohjelmistojen testausprojektit eivät eroa yleisesti projekteissa asetettujen tavoitteiden saavuttamisesta. (Ashfaque, 2010.)

Testausprojektiin kohdistuvia riskejä ovat esimerkiksi puutteellinen viestintä, kulttuuriset erot, osaamis-, teknologia-, prosessi-, ja infrastruktuuririskit sekä huono suunnittelu (Ashfaque, 2010). Tässä luvussa käydään läpi erilaisia testauksen toteutukseen vaikuttavia ja liittyviä riskejä. Lisäksi kerrotaan millä tavoin riski vaikuttaa testaukseen toteutuessaan.

5.1 Aikatululliset riskit

Aikatauluihin liittyvät riskit ovat yksi yleisimmistä projektin hallinnan riskeistä. Useiden tietoteknisten projektien epäonnistumisten syynä oli nimenomaan ongelmat aikataulun kanssa. Huonosti tai väärin suunniteltu projektin aikataulu vaikuttaa tuotteen kehittämiseen sekä toimittamiseen. Projektin epäonnistumisilla tai myöhästymisellä on myös negatiivisia vaikutuksia organisaation talouteen. (Artto, Martinsuo & Kujala 2011, 120–122.)

Aikatululliset riskit liittyvät pääosin tuotteen toteutuksen tai luovutuksen myöhästymiseen. Tällöin kehitystiimille on annettu liian tiukka toteutusaikataulu, jossa ei ole kyetty pysymään. Testauksen kannalta tämä voi tarkoittaa tilannetta, jossa testaukselle suunniteltu aika on liian lyhyt tai aikaa testaukselle ei ole suunniteltu lainkaan. Virheellisesti suunniteltuun aikatauluun voi vaikuttaa esimerkiksi kehitykseen kuluvan ajan väärin arviointi ja resurssien väärin allokointi. Myös projektialueen toistuva laajentuminen tai kriittisen virheen löytyminen ja sen korjaaminen sekä laadunvarmistus voivat viivästyttää projektin valmistumista. (Artto et al 2011, 120–122; Ashfaque, 2010.)

Mikäli projektin aikataulusta poiketaan, on syytä huomioida muutokset aikataulutavoitteisiin ja tehdä asiaankuuluvat muutokset. Merkittävät poikkeamat aikataulussa vaativat

projektin aikataulun uudelleensuunnittelun, sillä usein kehitystiimin tehtävät ovat sidoksissa toisiinsa. Lisäresurssien hankinta kehitystiimiin voi olla välttämätöntä, mikäli halutaan nopeuttaa projektin aikataulua tai pysyä suunnitellussa aikataulussa. (Artto et al 2011, 120–122.)

5.2 Kustannuksiin liittyvät riskit

Projektin kustannusten hallinta sisältää projektin budjetoinnin, kustannusten arvioinnin sekä kustannusten seurantaan liittyvät toiminnot. Kustannusten hallinnalla varmistetaan, että projekti toteutetaan kustannustehokkaasti sekä liiketaloudellisesti kannattavasti. (Artto et al 2011, 150–152.)

Projektin kokonaisbudjetin lisäksi kustannusten hallintaan sisältyy myös projektin eri osioiden hinnoittelu ja budjetointi sekä kassavirtojen ja rahoituksen suunnittelu. Tietoteknisissä projekteissa myös mahdolliset muutokset on huomioitava etukäteen alkuperäistä budjettia suunnitellessa. Projektin aikataululliset muutokset vaikuttavat myös kokonaiskustannuksiin. Kustannusten hallinta vaikuttaa niin ikään aikataulun ja resurssien suunnitteluun ja päinvastoin. (Artto et al 2011, 150–152.)

Syitä kustannusten ylittymiseen voivat olla edellä mainitut aikataulu- tai resurssointi haasteet. Muita syitä voivat olla virheelliset alkuperäisbudjetoinnit, projektialan laajentuminen tai kustannusseurannan epäonnistuminen. Joskus kustannuksia voidaan pyrkiä leikkaamaan vähentämällä testaukseen kohdistuvia kustannuksia. Tämä toiminta sisältää riskejä, jotka toteutuessaan voivat aiheuttaa järjestelmän laadun ja toiminnallisuuden heikkenemistä. (Ashfaque, 2010).

5.3 Operationaaliset riski

Operatiivisilla riskeillä tarkoitetaan projektin toimintaan liittyviä riskejä, joita esiintyy päivittäisessä operatiivisessa toiminnassa projektin kehittämisen aikana. Operationaaliset riskit vaihtelevat projektista koosta ja sisällöstä riippuen. Operationaalisia riskejä voidaan ennakoida ja minimoida perusteellisella projektin suunnittelulla. (Ashfaque, 2010.)

Resursseihin liittyvät riskit ovat osa operationaalisia riskejä. Tämän tyyppiset riskit voivat liittyä esimerkiksi yrityksen oman kehitystiimin työntekijöihin tai ulkopuolisiin henkilöstöresursseihin. Haasteena saattaa olla esimerkiksi työntekijöiden puutteellinen opastus tai koulutus tai liian vähäinen työntekijöiden määrä. Kehitystiimin epäselvät roolit ja vastuujaot voivat myös aiheuttaa ongelmia päivittäisessä työympäristössä ja pidemmällä tähtäimellä aiheuttaa riskien muodostumisen projektin läpiviennille. Tietoteknisessä projektissa liiketoiminnan edustajien puuttuminen tai liiketoimintaprosessien ymmärtämättömyys luovat omat riskinsä projektin kehitystoiminnalle. (Artto et al 2011, 175–177; ISTQB 2018, 65–66.)

Operationaalisia riskejä voidaan minimoida perusteellisella projektin suunnittelulla. Projektisuunnitelmassa kuvataan projektin sisältö sekä tavoitteet, projektin sisäiset toimintatavat ja johtamisperiaatteet. Projektin sekä testauksen etenemisen seuraamista varten on oleellista määritellä mittavissa olevat virstanpylväät, jotka voi olla sidonnaisia joko projektin ja testauksen aikatauluun tai tavoitteisiin. Suunnitelmaan on suotavaa tehdä liitteeksi myös viestintäsuunnitelma, jossa käydään läpi viestinnän toimintaperiaatteet projektin sisäisten ja ulkoisten sidosryhmien kesken. (Artto et al 2011, 175–177.)

Projektin johtamistyyliin on syytä kiinnittää alusta saakka huomiota. Avointa kommunikaatiota ja yhteistyötä kehitystiimin sisällä on syytä painottaa, sillä niiden on todettu olevan usein osatekijöinä menestyneissä projekteissa. Huono tai puutteellinen projektijohtaminen sekä konfliktit työtehtävien ja työntekijöiden kesken ovat osa operationaalisia riskejä, jotka vaikuttavat koko projektin etenemiseen. (Artto et al 2011, 175–177; ISTQB 2018, 65–66.)

5.4 Tekniset riskit

Tietoteknisen projektin teknisillä riskeillä tarkoitetaan kehitteillä olevan järjestelmän toiminnallisia tai ei-toiminnallisia riskejä. Teknisiin riskeihin lasketaan myös järjestelmän tietoturvaan ja suorituskykyyn liittyvät riskit. Riskien toteutuminen voi aiheuttaa suurta vahinkoa niin tuotteelle, asiakkaan kuin toimittajan liiketoiminnalle sekä riskistä riippuen yrityksen imagolle. (Ashfaque, 2010; ISTQB 2018, 65–66.)

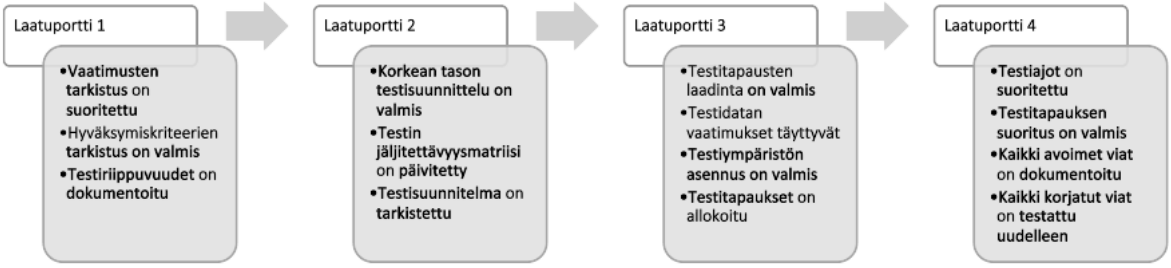
Teknisiä riskejä voi minimoida hyvällä suunnittelulla, ammattitaitoisen kehitystiimin hyödyntämisellä sekä asianmukaisen teknologian käytöllä. Kehitteillä olevan järjestelmän määrityksissä ja vaatimuksissa on keskityttävä yksinkertaisuuteen niin vaatimuksia kirjatessa

kuin toteuttaessa. Liian monimutkaiset määritykset vaikeuttavat toteutusta ja yhtä lailla liian monimutkaisesti toteutettu koodi voi aiheuttaa turhia teknologisia riskejä. (Ashfaque, 2010; ISTQB 2018, 65–66.)

Myös liian laveat tai puutteelliset laatukriteerit muodostavat teknisen riskin. Projektin alussa on hyvä laatia laatusuunnitelma, jossa määritellään projektin laatukriteerit sekä erilaiset käytännöt, proseduurit ja resurssit, joilla laadunhallintaa toteutetaan. Projektin edetessä toimintaa sekä lopputuloksia on syytä verrata määriteltyihin laatukriteereihin ja varmistaa, että laatu on halutun tasoista. (Artto et al 2011, 226–227; ISTQB 2018, 65–66.)

Laatuporttien käyttö projektin eri vaiheissa takaa säännöllisen laadunvarmistuksen. Laatuporttien ideana on, että järjestelmän laadun on vastattava ennalta määriteltyjä laatukriteereitä ennen projektin, määrittelyn tai testauksen siirtymistä seuraavalle tasolle. (Elfieden 2002, 26)

Kuvassa 7 on kuvattuna esimerkki laatuporttien käytöstä testauksen laadunhallinnassa. Laatuportteja voidaan käyttää esimerkiksi siirryttäessä testauksen tasolta toiselle. Tällä tavoin estetään edellisellä testaustasolla olevien virheiden tai puuttumattomien töiden siirtyminen seuraavalle testauksen tasolle. (Elfieden 2002, 26.)



Kuva 7. Esimerkki laatuporttien käytöstä testauksessa Elfieden (2002) mallia mukaillen.

5.5 Projektin ulkopuoliset riskit

Projektin ulkopuoliset riskit ovat nimensä mukaisesti projektin ulkopuolella tapahtuvia tapahtumia, joilla on toteutuessaan vaikutuksia projektin läpivientiin ja lopputulokseen. Nämä

riskit tulevat projektin ulkopuolelta, eikä projektilla ole näin ollen vaikutusvaltaa riskin toteutumiseen. (Ashfaque, 2010.)

Projektin ulkopuolisia riskejä ovat muun muassa markkinoiden nopea kehitys tai muut muutokset markkinoilla. Esimerkkejä markkinoilla tapahtuvista muutoksista ovat esimerkiksi kilpailijoiden lisääntyminen radikaalisti tai uusien teknologioiden synty. Myös muutokset lainsäädännössä voivat vaikuttaa kehitteillä olevaan järjestelmään siinä määrin, että muodostuu projektille operationaalinen tai tekninen riski. Tämän kaltaisessa tapauksessa on syytä varmistaa, että niin toteutuksessa kuin testauksessakin osataan huomioida tarvittavat muutokset ja, että kehitystiimissä on tarvittavaa osaamista (Ashfaque, 2010.)

Rahoituksen päättyminen tai sopimusten menettäminen erinäisistä syistä aiheuttavat projektin toteutukselle suuren riskin. Projekti voi esimerkiksi menettää ulkopuolisen testaustiiminsä testausta tarjoavan yrityksen vastoinkäymisistä johtuen. Tämän kaltaiseen riskiin on pystyttävä varautumaan riskienhallinnan kautta, vaikka itse riskin toteutumiseen ei voisikaan projekti itse vaikuttaa. (Ashfaque, 2010.)

Vuonna 2019 alkanut Covid-19 –pandemia on yksi esimerkki projektin ulkopuolelta tulevasta riskistä. Pandemialla oli suuria negatiivisia vaikutuksia moneen eri alaan, mutta ICT-alaan pandemian vaikutukset olivat monelta osin positiivisia; työtä pystyttiin jatkamaan etänä ja erilaisten tietoteknisten alustojen, järjestelmien ja sovellusten kysyntä kasvoi. (Nieminen & Myllymäki, 2022.)

6 Jatkuva testauksen kehittäminen

Tässä luvussa käydään läpi erilaisia tapoja kehittää testausprosessia. Ensimmäisenä kerrotaan testausstrategian sekä testaussuunnitelman merkityksestä sekä niiden määrittelemistä. Tämän jälkeen käydään läpi PDCA-mallin periaate. Luvun lopussa kerrotaan kehitystiimin sisäisen kommunikaation merkityksestä sekä onnistuneen ja epäonnistuneen kommunikaation vaikutuksista kehitystiimin työskentelyyn. Lopuksi kootaan yhteen erilaiset hyödyt jatkuvan testausprosessin kehittämisestä.

6.1 Testausstrategian määrittely ja testaussuunnitelman laatiminen

Ketterässä ohjelmistokehityksessä painotetaan nopeasti etenevää työskentelytapaa ja raskaiden suunnitteluprosessien puuttumista. Tällä tarkoitetaan käytännössä vesiputousmallista työskentelyä, jossa edetään vaihe kerrallaan ja luodaan verrattain paljon dokumentaatiota prosessin edetessä. Edellä mainittu on voitu tulkittu siten, että ketterissä menetelmissä pyrittäisiin täysin välttämään dokumentaatioiden ja perusteellisten suunnitelmien laatimista. Tämä on kuitenkin väärin, sillä esimerkiksi ketterän kehityksen manifestissa ei ole asiasta lainkaan mainintaa. On myös huomioitava, että projektin onnistuneen läpiviennin mahdollisuudet nousevat, mikäli projektin eri osa-alueiden suunnitteluun on panostettu. Osa testauskokonaisuuden suunnittelua on testausstrategian valinta sekä testaussuunnitelman laatiminen. (Artto et al 2011, 268; ISTQB 2015, 11–13; Rosenberg, Collins-Cope & Stephens 2005, 76.)

Testausstrategian valintaan vaikuttaa yrityksen testauspolitiikka sekä kehitteillä oleva tuote. Määritelty strategia vaikuttaa testausprosessin läpivientiin ja olennaisesti testitapausten suunnitteluun. Testausstrategia voidaan jakaa testausstrategiatyyppeihin eli lähestymistapoihin. Taulukossa 2 on ISTQB perustason testaajasertifikaatissa luetellut (2018, 60) testauksen lähestymistavat.

Taulukko 2. Testausstrategian erilaiset lähestymistavat.

Lähestymistapa	Selite
Analyyttinen	Pohjautuu jonkin tietyn tekijän, kuten vaatimuksen tai riskin analysointiin. Riskipohjainen testaus on osa analyyttista lähestymistapaa.
Mallipohjainen	Testitapaukset suunnitellaan esimerkiksi tietyn ominaisuuden, toiminnon, liiketoimintaprosessin perusteella.
Menetelmällinen	Testauksen kohteena on ennalta määritelty testijoukko tai testattava skenaario kuten laatuominaisuuksien listaus, web-sivujen ulkoasustandardit tai mobiilisovellukset.
Prosesseihin ja standardeihin perustuva	Testien analysointi, suunnittelu ja valmistelu ulkoisten säännösten ja standardien perusteella.
Ohjaava	Testaus perustuu sidosryhmien, liiketoiminta-asiantuntijoiden tai teknologia-asiantuntijoiden neuvoihin, ohjeisiin ja opastuksiin.
Regressiota ehkäisevä	Tarkoituksena välttää olemassa olevien toiminnallisuuksien heikkeneminen. Lähestymistapaa sisältyy testausmateriaalin uudelleenkäyttö, testitapausten automatisointi sekä vakiintuneet testijoukot.
Reaktiivinen	Testaus reagoi testattavana olevaan järjestelmään tai komponenttiin sekä testauksen aikana tapahtuviin tilanteisiin. Tutkiva testaus on reaktiivinen lähestymistapa.

Käytännössä testauksessa käytetään usein useampaa lähestymistapaa riippuen testauksen kohteesta. Erilaiset testauksen lähestymistavat täydentävät toisiaan ja niiden yhdessä käyttämisellä voi olla testausta tehostava vaikutus. (ISTQB 2018, 60.)

Testausprosessin kannalta oleellisin suunnitelma on testaussuunnitelma. Testaussuunnittelussa linjataan testaustoimenpiteet kehitys- ja ylläpitoprojekteissa. Suunnitelman laadintaan vaikuttaa muun muassa yrityksen oma mahdollinen testauspolitiikka ja valittu testausstrategia. Testauksen suunnittelu on koko kehitysprosessin ajan jatkuvaa ja näin ollen testaussuunnitelman sisältö voi päivittyä projektin edetessä. (Claesson 2022; ISTQB 2018, 59–61.)

Testaussuunnitelma voidaan jakaa erilaisiin tasokohtaisiin kokonaisuuksiin tai suunnitelmat voidaan vaihtoehtoisesti yhdistää yhdeksi yhtenäiseksi kokonaistestaussuunnitelmaksi. Eri tasokohtaisia suunnitelmia ovat esimerkiksi järjestelmä- ja hyväksymistestaussuunnitelmat. Suunnitelmat voidaan jakaa myös testaustyypeittäin, jolloin esimerkiksi käytettävyys- ja suorituskykytestauksille on laadittu omat testaussuunnitelmat. (Claesson 2022; ISTQB 2018, 59–61.)

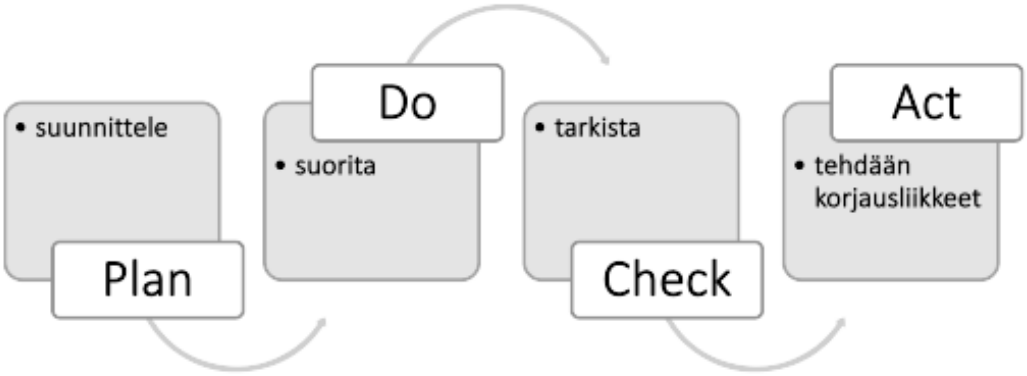
Testaussuunnitelmassa otetaan kantaa testauksen laajuuteen, testauksen tavoitteisiin sekä mahdollisiin riskeihin testausprosessissa. Suunnitelmaan sisällytetään myös muun muassa testauksen yleisperiaatteet, resurssien käyttö testauksessa, testitapausten suunnittelu, valmistelu ja suorittaminen. Oleellista on myös määritellä testausprosessin laatuportit ja mittarit testauksen etenemistä varten. (Claesson 2022; ISTQB 2018, 59–61.)

ISO-standardi ISO/IEC/IEE 29119-3 sisältää esimerkkejä testaussuunnitelmista. Suunnitelmien sisältö vaihtelee yrityksen sekä testattavan kohteen mukaan. On hyvä huomioida, että testaussuunnitelma voi kattaa myös enemmän tai vähemmän asioita kuin edellä on mainittu. (ISTQB 2018, 59–61.)

6.2 PDCA-malli

PDCA-mallin tarkoituksena on parantaa kohteena olevaa toimintaa. PDCA-mallissa prosessi kulkee uudelleen toistuvassa kehässä, jolloin toiminnan suorituksen laatu tai suorituskyky paranee jokaisella kierroksella. Mallin nimi tulee englannin kielen sanoista ´Plan´, ´Do´, ´Check´ ja ´Act´. (Torkkola 2015, 72.)

Mallin ensimmäinen kohta on ´Plan´ eli suunnittelu (kuva 8). Suunnitteluvaihe on prosessin keskeisin vaihe ja siihen on suotavaa panostaa. Suunnitteluvaiheessa määritellään hypoteesi sekä mittarit, joiden avulla selvitetään, onko kohteena oleva toiminta onnistunut. Mallin toisessa suoritetaan (´Do´) kohteena oleva toiminto, jonka jälkeen tarkistetaan (´Check´) onnistuiko toiminto suunnitelman sisällön ja mittarien mukaisesti. Mikäli toiminnasta löydettiin virheitä, esteitä tai jokin muu asia ei toiminut odotetusti, laaditaan ja toteutetaan korjausliikkeet (´Act´), jolloin sykli alkaa alusta. Muutokset on hyvä pitää pieninä, jolla prosessin läpivienti olisi nopeaa ja tulokset helpommin nähtävillä. Hypoteesia, toteutustapaa ja tavoitteita voidaan tarpeen mukaan muokata ennen uuden syklin aloittamista. (Torkkola 2015, 72.)



Kuva 8. PDCA-sykli Torkkolan (2015, 73) mallia mukaillen.

PDCA-mallia voidaan käyttää erilaisissa ympäristöissä ja prosessin kohteena on voi olla hyvinkin erilaisia asioita. Testausprosessissa PDCA-mallia voidaan hyödyntää testitapausten läpiviennissä kuin itse testausprosessin kehittämisessä ja parantamisessa. Tutkivan testauksen prosessi on mukaelma PDCA-mallista (Kinsbruner 2018, 118; Torkkola 2015, 72.)

6.3 Kehitystiimin sisäiseen kommunikointiin panostaminen

Ketterässä kehitysympäristössä kommunikaation merkitys korostuu entisestään. Yhteistyö tiimin ja sidosryhmien kesken suoritetaan usein kasvotusten kirjallisten dokumentaatioiden sijaan. Palaverit ovat lyhytkestoisia ja sisällöltään tiiviitä, jolloin viestinnän on oltava tehokasta. Toistuva ja avoin viestintätapa edesauttaa hyvän kommunikaatiosuhteen kehittymistä tiimiläisten kesken. (ISTQB 2018, 22–23.)

Tiimin ja koko projektin sisäistä viestintää voi heikentää useampi seikka. Selkeä kommunikaatiotaidon puute niin suullisesti kuin kirjallisesti on suurin riski viestinnän epäonnistumiselle. Myös haluttomuus jakaa tietoa tai huomioitaan muiden tiimiläisten kanssa voi hankaloittaa ja hidastaa työtehtävien etenemistä. (ISTQB 2018, 22–23.)

Ajoittain kehittäjien ja testaajien välille voi muodostua kuilu useasta eri syystä; kehittäjät voivat kokea testaajat oman työnsä kritisoijina eivätkä koe testaajien olevan osa tiimiä. Tämä asetelma voi korostua varsinkin, jos testaajat ovat yrityksen ulkopuolisia resursseja. Edellä mainittu voi johtaa tilanteeseen, jossa testaajat jätetään pois relevantista viestinnästä. Myös

tietojen jakaminen puolin ja toisin vaikeutuu. Jotta vastaavaan tilanteeseen ei päädytä, on yhteishenkeä, yhteistä tavoitetta sekä avointa keskustelua ja ilmapiiriä painotettava alusta lähtien. (Baumgartner et al 2021, 39.)

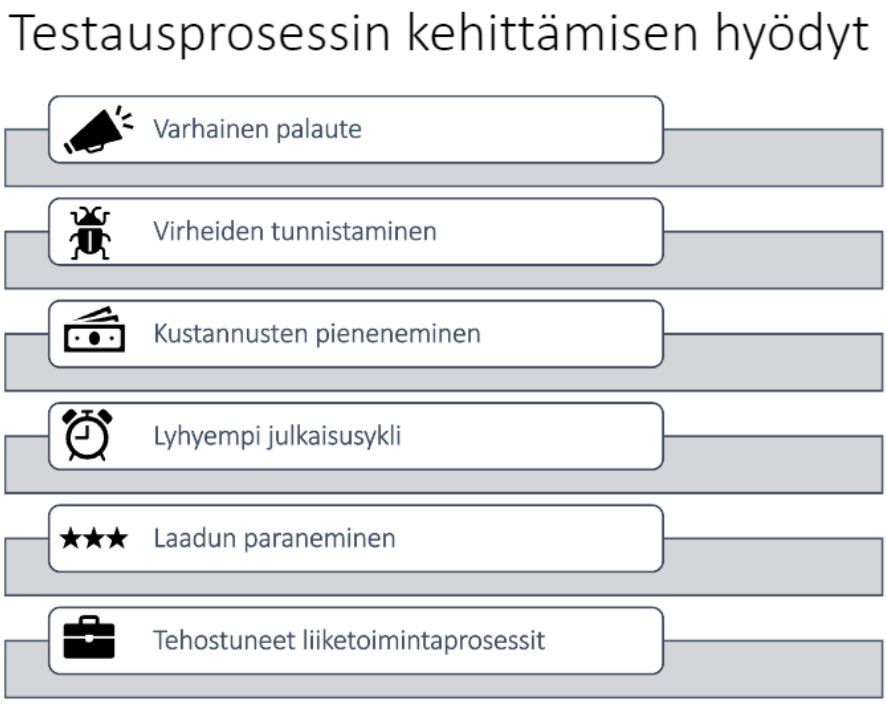
Palautteenanto- sekä vastaanottokyky ja kehittävien muutosten ideointi on tärkeä osa ketterää kehitystä. Tiimin ilmapiirin olisi hyvä olla siinä määrin suotuisa, että jokaisella olisi mahdollisuus ilmaista mielipiteitään ja kehitysehdotuksiaan. Toiminnan jatkuvalla kehittämisellä ja kehitystyön ripeällä suorittamisella on tärkeä osa ketterässä kehityksessä, jonka vuoksi avoin viestintä on oleellinen osa onnistunutta järjestelmänkehitysprojektia. (Baumgartner et al 2021, 39; ISTQB 2018, 22–23.)

6.4 Testausprosessin kehittämisen hyödyt

Testausprosessin kehittäminen ja parantaminen ovat tärkeä osa niin järjestelmän kuin liiketoimintaprosessien laadun varmistamista. Prosessin parantamisesta voidaan hyötyä monella eri tavoin; vaikutukset koskettavat niin henkilöstö- kuin asiakastyytyväisyyttä, kehitteillä olevan järjestelmän kehitysprosessia ja laatua sekä projektin kustannusten pienenemistä. Hyödyt voidaan jakaa suoriin ja epäsuoriin hyötyihin. (Watkins & Mills 2011, 102–105; Madeyski 2010, 87–90.)

Testausprosessin kehittämisen suoria hyötyjä ovat tuottavuuden kasvu, kustannusten laskeminen, varhaisten vikojen havaitseminen ja näin ollen laadun paraneminen. Epäsuoriin hyötyihin sisältyy muun muassa optimoitu työympäristö, tyytyväisemmät tiiminjäsenet sekä lisääntynyt asiakasuskollisuus ja -tyytyväisyys.

Kuvaan 9 on koottuna testausprosessin kehittämisen oleellisimpia hyödyt. Varhaisella palautteella tarkoitetaan niin tiimin sisäistä kuin sidosryhmien antamaa palautetta. Rakentavan palautteen antamisen mahdollistaminen eri kehitysvaiheissa nopeuttaa järjestelmän kehittämisen pienemmällä määrällä vikoja. Sidosryhmien antama palaute taas tuo ilmi mahdolliset virheet tai väärinymmärrykset määrittelyissä tai kehityksessä. Mikäli tämän kaltaisiin palautteisiin on mahdollista puuttua aikaisessa vaiheessa, vältytään isoilta korjauksilta kehityksen myöhemmissä vaiheessa ja säästetään aikaa sekä kustannuksia.



Kuva 9. Testausprosessin kehittämisen hyödyt (Madeyski 2010, 87–90)

Testauksen kehittäminen ja testauksen läpiviennin tehostaminen nopeuttaa testauksen läpimenoaikaa, mahdollistaa virheiden tunnistamisen jo aikaisessa vaiheessa kehitystä sekä alentaa virheiden korjaamiseen liittyviä kokonaiskustannuksia. Tehostunut testaus ja virheiden korjaus vaikuttavat positiivisesti myös julkaisusykleihin. Myös mahdolliset käyttökatkot järjestelmässä saadaan lyhennettyä minimiin optimoidulla testausprosessilla. Testausprosessin kehittämisellä voidaan vaikuttaa positiivisesti niin yrityksen omiin kuin asiakkaankin liiketoimintaprosesseihin laadunhallinnan lisäksi.

7 Tutkimuksen toteutus

Tässä luvussa käydään läpi, miten tutkimus toteutettiin. Ensiksi kerrotaan, kuinka tutkimuksen aineisto kerättiin ja tämän jälkeen käydään läpi tutkimuksessa käytetty analyysimenetelmä.

7.1 Kohdeorganisaation esittely

Kohdeyritys on suomalainen digitaalisten palveluiden suunnitteluun ja kehittämiseen keskittynyt yhtiö, jonka liikevaihto oli vuonna 2022 lähes 150 miljoonaa euroa. Henkilöstöä yrityksellä on noin 1 300. Kohdeyritykseltä löytyy toimipisteitä niin Suomesta kuin useasta muusta Euroopan maasta.

Tässä diplomityössä kohdeyritystä ei nimetä, sillä haastatteluissa sekä testausprosessien analysoinnissa ja läpikäynnissä käy ilmi liiketoimintasalaisuuksia sekä yrityksen asiakkaiden nimiä. Kilpailuasetelman vuoksi on myös oleellista, että kohdeyrityksen sisäiset toimintaprosessit eivät ole kilpailijoiden tiedossa.

7.2 Tutkimusprosessin kuvaus

Taulukossa 3 on esitettynä tutkimusprosessin vaiheet. Taulukossa on sarakkeittain tutkimusprosessin vaihe, tutkimusmateriaali, analyysimenetelmä sekä tutkimusprosessin vaiheesta syntynyt tuotos.

Taulukko 3. Tutkimusprosessin vaiheet.

Tutkimusprosessin vaihe	Tutkimusmateriaali	Analyysimenetelmä	Tuotos
Aihealueeseen tutustuminen	Keskeiset käsitteet	Kirjallisuuskatsaus	Teoreettinen viitekehys
Haastattelut	Kuusi haastateltavaa 288 minuuttia nauhoitettuja haastatteluaja	Aineistolähtöinen sisällönanalyysi Juurisyyanalyysi	Tämän hetkisen estausprosessin analyysi “Viisi kertaa miksi” -analyysin tuotokset
Prosessimallin kehittäminen	Kaikki tutkimusmateriaali	Erilaisten ratkaisuvaihtoehtojen analysointi Ratkaisuvaihtoehdon valinta perustuen toimeksiantajan tarpeisiin ja resurssointiin	Prosessimalli Prosessin kuvaus Toimintaehdotus prosessimallin lisäksi

Tutkimusprosessin ensimmäinen vaihe sisälsi aihealueeseen tutustumisen kirjallisuuskatsauksen keinoin, jolloin tuotoksena syntyi työn teoreettinen viitekehys. Toinen vaihe käsitti tutkimusprosessin empiirisen tutkimuksen eli kuuden toimeksiantajan työntekijän haastattelut. Haastattelumateriaali analysoitiin ja esiin nousseista asioista tehtiin juurianalyysimenetelmän avulla johtopäätöksiä. Tutkimusprosessin viimeinen vaihe oli prosessimallin kehittäminen. Prosessimallin tarkoituksena on parantaa toimeksiantajan testausprosessia. Tämän vaiheen tuotoksena syntyi kehitetty prosessimalli, kuvaus prosessimallin käyttöönottoon sekä toimintaehdotus prosessimallin käytön lisäksi.

7.3 Aineiston keruu ja haastateltavat

Tutkimuksessa on käytetty laadullista eli kvalitatiivista tutkimusmenetelmää. Tutkimuksen aineiston keruu suoritettiin haastattelujen kautta. Kohdeyrityksen yhteyshenkilö valitsi haastateltavat, jotka ovat listattuna taulukkoon 4. Kriteerinä haastateltavien valintaan oli eri työtehtävät haastateltavien kesken, jotta haastatteluissa nousisi helpommin esiin eri roolien

kohtaamia haasteita. Kolmella haastateltavalla oli sama titteli, mutta he työskentelivät eri projekteissa ja heidän työtehtävänsä erosivat hyvin paljon toisistaan.

Taulukko 4. Haastatellut henkilöt.

Haastateltava	Titteli	Haastattelun ajankohta	Haastattelun kesto
A	Project Manager	28.2.2023	44 minuuttia
B	Test Manager	2.3.2023	57 minuuttia
C	Test Manager	2.3.2023	43 minuuttia
D	Test Manager	3.3.2023	44 minuuttia
E	Service owner for test automation	8.3.2023	48 minuuttia
F	Test Automation Consultant	2.3.2023	52 minuuttia

Haastattelussa käytettiin kysymysrunkoa, joka loi raamit haastattelulle, mutta muutoin haastattelu oli avointa ja haastateltavaa kehotettiin kertomaan vapaasti havainnoistaan.

Haastattelussa käytettyä haastattelurunko oli seuraavanlainen:

1. Mikä on nykyinen työtehtäväsi ja miten se liittyy laadunvarmistukseen?
2. Minkälainen on tämänhetkinen testausprosessi projektissa, jossa työskentelet?
3. Mitä laadunvarmistusprosessissa on mielestäsi toimivaa?
4. Mitä laadunvarmistusprosessissa olisi mielestäsi kehitettävää? Mitkä asiat ovat huonosti?
5. Mitkä asiat nostaisit toimiviksi tai hyviksi ketterissä menetelmissä?
6. Mitkä asiat koet toimimattomiksi tai huonoiksi ketterissä menetelmissä?
7. Onko testausprosessikokonaisuus mielestäsi hallinnassa?
8. Nouseeko mieleesi mitään muuta lisättävää?

Kvalitatiivisessa tutkimuksessa on ominaista, että haastatteluja jatketaan, kunnes saturaatio on saavutettu. Saturaatiolla tarkoitetaan tilannetta, jossa uutta tietoa ei haastattelujen aikana enää ilmene (Hirsjärvi & Hurme 2001, 60). Tämän diplomityön haastatteluissa toistuivat samat asiat jokaisen haastateltavan kohdalla, mutta haastattelut suoritettiin loppuun muun muassa haastattelumäärän vähyydestä johtuen. Uusien haastateltavien etsimiselle ei ollut tarvetta ja saturaatiopiste voitiin katsoa täyttyneeksi.

Haastateltaville kerrottiin ennen haastatteluja, että haastateltavien nimiä ei tulla julkaisemaan eikä haastattelussa ilmenneitä asioita kerrota tunnistettavasti eteenpäin kohdeyrityksen sisällä tai diplomityössä. Haastattelut nauhoitettiin ja litteroitiin tutkimuksen analysointiosiota varten. Nauhoitukset tuhottiin heti analysointiprosessin jälkeen.

7.4 Tutkimustulosten analyysimenetelmä

Laadullisen tutkimuksen analyysin tarkoituksena on vastata tutkimuskysymyksiin ja kuvata sanallisesti tutkittavaa ilmiötä (Tuomi & Sarajärvi 2013, 108). Tässä diplomityössä aineisto analysoitiin aineistolähtöisen sisällönanalyysin keinoin. Litteroitu teksti hajotettiin ensin käsitteellisiksi osiksi, jonka jälkeen se järjestettiin tiiviiseen ja selkeään muotoon.

Aineistolähtöisessä sisällönanalyysissä tutkimuksen analyysi alkaa haastattelujen litteroinnilla. Litteroinnin tarkoitus on antaa kokonaisvaltainen kuva aineistosta ja mahdollistaa uusien näkökulmien nouseminen tutkittavaan aiheeseen. Litteroinnin jälkeen suoritetaan aineiston tarkastelu. Tämän vaiheen aikana luokitellaan tekstin sanat ja ilmaisut niiden teoreettisten merkitysten perusteella. Pelkistettyjä ilmaisuja voidaan vertailla keskenään ja yhdistää sisällöltään samankaltaiset ilmaisut yhdeksi alakategoriaksi. Analyysin viimeistä vaihetta kutsutaan abstrahoinniksi. Abstrahoinnissa alakategoriat yhdistetään ja niistä muodostetaan yläkategorioita. Nämä yläkategoriat vastaavat tutkimuskysymyksiin ja ovat näin ollen myös tutkimuksen tuloksia. (Tuomi & Sarajärvi 2013, 109).

7.5 Juurisyyanalyysi

Haasteita ja ongelmakohtia lähdettiin analysoimaan juurisyyanalyysin avulla. Osa analyysista tehtiin jo haastattelujen aikana haastateltavien kanssa ja analyysit viimeisteltiin haastattelukierroksen päätyttyä.

Juurisyyanalyysi on tieteellinen menetelmä, jossa kokonaisuus pilkotaan pienempiin osiin syy - seuraus –ketjun avulla. Analyysin alussa luodaan kuvaus käsiteltävästä ongelmasta tai haasteesta. Tämän jälkeen esitetään kysymys “miksi” niin monta kertaa, että ongelman tai haasteen perimmäinen syy on selvitetty. Englanniksi menetelmää kutsutaan “Five whys” -menetelmäksi. Perimmäisen syyn selvittyä on ryhdyttävä miettimään, miten tilanne korjataan tai vältetään tulevaisuudessa. (Andersen & Fagerhaug 2006, 129–133)

Luvussa 8.4 käydään läpi juurisyyanalyysin tuloksia.

8 Testausprosessin analysointi kohdeorganisaatiossa

Tässä luvussa käydään läpi haastattelujen sisältöä sekä nostetaan esiin oleellisimpia asioita haastatteluista. Haastattelut suoritettiin haastattelurungon mukaisesti. Vaikka haastateltavat työskentelivät eri projekteissa ja erilaisissa työtehtävissä, olivat heidän vastauksensa hyvin samankaltaisia keskenään. Haastatteluissa nousi selkeästi esille neljä suurempaa aihekokonaisuutta, jonka ympärille haastattelujen sisällöt tavalla tai toisella liittyi. Taulukossa 5 nähdään kooste haastatteluissa eniten esiin nousseista asioista. Aihekokonaisuudet on jaettu neljään yläkategoriaan; asiakkuus, laadunvarmistus, ketterät menetelmät ja tulevat muutokset. Asia-sarakkeessa on tarkemmin lueteltu yksittäiset asiat, jotka yläkategoriaan liittyi.

Taulukko 5. Kooste haastattelujen keskeisimmistä asioista.

Aihekokonaisuus	Asia
Asiakkuus	Kommunikaation parantaminen Resurssien stabilointi Testauspalvelu kokonaisuutena Product ownereillä liikaa töitä Asiakkaan tahtotila ei ole aina selkeä Etupainotteisen työn lisääminen
Laadunvarmistus	Suunnitelmallisuuden puuttuminen Kokonaisuuden hallinta takkuaa Läpinäkyvyys puuttuu Testiympäristöt epävakaita
Ketterät menetelmät	Kuka on vastuussa? Siiloutuminen Järkevät proseduurit unohtuvat Kommunikaation tiimien välillä heikko Ketterien menetelmien kouluttamista lisää Kuplassa työskentely Itseohjautuvuus
Tulevat muutokset	Tekoälyn tuomat mahdollisuudet Testaajan rooli muuttuu ja kehittyy Testiautomaatio tulee lisääntymään entisestään Uusia menetelmiä tulevaisuudessa

Haastatteluissa keskusteltiin paljon yhteistyöstä asiakkaan kanssa sekä mahdollisista haasteista konsulttien ja asiakkaiden välillä. Taulukossa 5 on mukana asiakokonaisuuksia, joita käsiteltiin haastattelujen aikana asiakkuuksiin liittyen. Yksi tärkeä asiakokonaisuus oli kommunikaatio, jota käsitellään tässä luvussa myöhemmin lisää. Muut asiakkuuteen liittyvät asiakokonaisuudet liittyvät enemmän yritysten välisiin sopimuksiin, toimeksiantajayrityksen resurssoinnin toimintatapoihin ja asiakasyrityksen sisäiseen resurssointiin. Tämän vuoksi asiakkuutta ei ole enää myöhemmin nostettu omaksi kokonaisuudeksi, sillä kyseiset osa-alueet eivät suoranaisesti liity testausprosessin kehittämiseen. Product owneriin liittyvää asiakokonaisuutta käsitellään Testausprosessi ja kokonaisuuden hallinta- otsikon alla.

Haastatteluissa keskusteltiin myös tulevaisuudesta sekä siitä millaisena haastateltavat näkevät tulevaisuuden testauksen saralla. Taulukossa 5 on nähtävillä asiakokonaisuudet, joita haastateltavat toivat haastatteluissa esiin. Osa tämän aihepiiriin asioista on suoraan kytköksissä testausprosessin ja kokonaisuuden hallintaan. Tämän vuoksi kokonaisuudelle ei ole erikseen omaa osiota, vaan haastatteluissa esiin nousseita asioita käsitellään Testausprosessi ja kokonaisuuden hallinta –otsikon alla.

Muut kaksi asiakokonaisuutta ovat ketterien menetelmien toteuttaminen käytännössä sekä laadunvarmistus eli testausprosessin ja kokonaisuuden hallinta. Erilaiset ongelmat kommunikaatiossa nousivat esille useamman kerran haastattelujen aikana, joten kommunikaation merkittävyyden vuoksi aihe on nostettu oman otsikkonsa alle.

Jokainen haastatelluista koki haasteita ja ongelmia edellä mainituissa kokonaisuuksissa omien projektiensa kohdalla. Haastateltavien projektit ja työtehtävät erosivat toisistaan, mutta samat haasteet olivat tunnistettavissa siitä huolimatta.

Tässä luvussa käydään läpi tarkemmin haastatteluissa esiin tulleita asioita jaoteltuna kolmeen tärkeimpään kokonaisuuteen.

8.1 Ketterien menetelmien toteutus

Haastatteluissa keskusteltiin ketteristä menetelmistä ja siitä miten ketteriä menetelmiä toteutetaan projekteissa. Lisäksi haastateltavilta pyydettiin kertomaan omasta mielestään ketterien menetelmien hyvät ja huonot puolet.

Ketterien menetelmien implementointi

Yhtä haastateltavaa lukuun ottamatta kaikki haastateltavat työskentelivät projekteissa, joissa käytettiin ketteriä menetelmiä. Vaikka jokainen haastateltava koki ketterät menetelmät hyvänä asiana, tuli haastatteluissa ilmi niin sanottua kyllästymistä ketteryyteen. “Toiminta on välillä liian ketterää”, totesi yksi haastateltavista. Toinen haastateltava ennusti, että ketterien menetelmien tilalle nousee lähitulevaisuudessa jokin täysin uusi menetelmä, joka olisi ideologialtaan ketterän ja vesiputousmallin välimaastossa.

Haastatteluissa kävi ilmi, että asiakasorganisaatioissa oli saatettu ottaa jokin ketterä menetelmä nopealla aikataululla käyttöön, sisäistämättä menetelmän todellista ideologiaa tai luonnetta. Tämä konkretisoitui siten, että menetelmää käytettiin osittain tai vain tiettyjen aspektien osalta. Ajatus ketteryyden tavoittelusta saattoi näkyä käytännössä myös siinä, että työntahti pidettiin liian nopeana työnlaadun kustannuksella. Eräs haastateltava totesi kuitenkin, että yleisesti ottaen ymmärrys ketteriin menetelmiin niin asiakasorganisaatioissa kuin konsulttienkin kesken on kasvanut viime vuosien aikana. Toimeksiantaja yrityksellä on käytössään ketterien menetelmien kouluttajia, jotka auttavat tarpeen tullen ketterin menetelmien käyttöönotossa.

Hyvät puolet ketterissä menetelmissä

Jokainen haastateltava totesi hyväksi asiaksi ketterissä menetelmissä vapauden sekä mahdollisuuden nopean palautteeseen. Molemmat edellä mainitut asiat vaikuttivat positiivisesti työtehtävien suorittamiseen ja työtehtävien edistämiseen.

Huonot puolet ketterissä menetelmissä

Huonoksi asiaksi ketterissä menetelmissä muutama haastateltava mainitsi suunnitelmallisuuden puutteen sekä tiimien siiloutumisen. Nämä molemmat asiat vaikuttavat osaltaan myös testausprosessin ja kokonaisuuden hallintaan. Tämän vuoksi asiakokonaisuutta käsitellään myös seuraavassa luvussa.

Eräs haastateltava kertoi, että asiakasorganisaatiossa ketterien menetelmien pyrkimys vähentää turhaa dokumentaatiota oli tulkittu tarkoittavan, ettei dokumentaatiota tai suunnitelmallisuutta tarvittaisi lainkaan. Suunnitelmallisuuden puuttuminen näkyi eräässä projektissa esimerkiksi siten, että testauksen toteutusta ei erikseen suunniteltu, vaan testaaja testasi kokonaisuuden, kuten parhaaksi näki. Toisen haastateltavan projektissa suunnitelmallisuuden puutetta osoitti se, että iteraation työmäärät olivat jatkuvasti liian suuria tai työtehtäviä lisättiin kesken iteraation. Tämä johti siihen, että työtehtäviä jouduttiin säännöllisesti siirtämään iteraatiosta toiseen. Se johtuiko iteraatioiden huono suunnittelu yksin Scrum masterista tai aiheuttiko haasteet jokin muu asia kehitystiimissä, jäi haastattelussa epäselväksi.

Toiseksi huonoksi asiaksi ketterissä menetelmissä haastateltavat nostivat esiin tiimien siiloutumisen. Ketterien menetelmien mukaisesti tiimit työskentelevät horisontaalisesti ja itsenäisesti. Tiimien itsenäinen työskentely oli kuitenkin useammassa projektissa johtanut siihen, että tiimit työskentelivät niin sanotussa kuplassa, tietämättä mitä toisessa tiimissä tapahtuu tai minkä asian parissa toinen tiimi työskentelee. Tämän kaltainen tilanne oli johtanut yhdessä projektissa useamman kerran tilanteeseen, jossa toisen tiimin työskentely keskeytyi tai vaikeutui toisen tiimin tekemisten vuoksi. Erään haastateltavan mukaan tiimien välinen kommunikaatio oli olematonta, joka entisestään korosti tiimien siiloutumista.

8.2 Testausprosessin ja kokonaisuuden hallinta

Testiautomaatio

Haastatteluissa kävi selkeästi ilmi, että automaatiotestauksen osuus erilaissa projekteissa on kasvanut ja tulee kasvamaan entisestään tulevaisuudessa. Muutaman haastateltavan kanssa käytiin keskustelua testaajan roolista tulevaisuudessa automaatioiden lisääntyessä. Kyseiset haastateltavat olivat sitä mieltä, että perinteisen testaajan rooli tulee väistämättä muuttumaan

ja todennäköisesti laajentumaan enemmänkin jonkin tietyn testauksen osa-alueen asiantuntijaksi. Yleisesti ottaen kaikki haastateltavat kokivat automaatiotestauksen positiivisena asiana. Eräs haastateltava kertoi, että automaatiotestauksen trendi oli verrattavissa aiempaan ketterien menetelmien trendiin; asiakasorganisaatiot ottivat innoissaan automaatiotestauksen käyttöön perehtymättä tarkemmin mitä prosessi todellisuudessa tarkoittaa. Haastateltavan mukaan monille asiakasorganisaatioille saattaa tulla yllätyksenä esimerkiksi se, että automaatiotestaus on pidempi aikaista toimintaa.

Ymmärryksen kasvaminen

Erään haastateltavan mukaan yhteisen tavoitteen ymmärrys ja merkitys testausprosessissa on kasvanut viime vuosien aikana. Haastateltavan mukaan ymmärrys siitä, että jokainen on roolistaan riippumatta niin sanotusti "luomassa laatua" on laajentunut työntekijöiden kesken. Tämä ajatusmalli osaltaan vaikuttaa myös siihen, että lopputulos pyritään saavuttamaan yhdessä.

Laadunvarmistuksen näkökulmasta tämä konkretisoitui esimerkiksi kehittäjien yksikkötestien suorittamisella. Jokainen kehittäjä testaa omat tuotoksensa ja näin varmistaa omien työtehtäviensä laaduntason.

Testausprosessin haasteet

Asiakasorganisaation työntekijöiden osuus projektissa tai kehitystiimissä riippui paljolti projektista, mutta useimmiten "asiakas toivoo valmista kauraa" eli asiakas haluaa olla mahdollisimman vähän mukana. Tämä tarkoittaa käytännössä sitä, että määritysdokumentaatiot on oltava tarkalla tasolla ja asiakkaan tahtotila hyvin tiedossa.

Haasteltavat luettelivat useita erilaisia haasteita testausprosessissa. Suurin haaste, jonka jokainen haastateltava mainitsi, oli vastuunottaminen. Haastatteluissa toistui seuraavat kysymykset: kuka on vastuussa kokonaisuudesta, kuka sanoo mihin keskitytään, mikä on kenenkin vastuulla ja kenelle jokin asia kuuluu, kuka päättää mikä on tarpeeksi hyvää?

Vaikka haastateltavat totesivat ketterien menetelmien hyväksi puoleksi vapauden, olivat haastateltavat kuitenkin sitä mieltä, että vapautta oli prosessin sisäisesti monin paikoin liikaa.

Useampi haastateltava mainitsi tilanteen, jossa työtehtäviä jää tekemättä tai niitä ei suoriteta täysin loppuun, sillä työtehtävä ei ole kuulu suoranaisesti kenenkään vastuualueeseen. Eräs haastateltava kertoi, että vapaus valita omia työtehtäviään oli johtanut eräässä projektissa siihen, että työntekijät valitsivat itselleen mieluisimmat työtehtävät ja niin sanotusti ikävimmät työtehtävät jäivät helposti tekemättä tai ne suoritettiin aivan viime tipassa.

Työtehtäviin liittyvä haaste oli myös edellä luvussa mainittu työtehtävien kasaantuminen per iteraatio. Haastateltavan mukaan tämä ongelma toistui iteraatiosta toiseen. Ketterien menetelmiin kuuluu vahvasti tehdyn työn arviointi sekä havaittujen virheiden tai huonojen työskentelytapojen korjaaminen tai muuttaminen. Haastattelussa ilmeni, että kyseisessä projektissa järjestettiin retrospektiivejä tiimien sisäisesti. Näissä palavereissa ei käsitelty tiimien välisiä tai projektitasoisia ongelmia.

Toinen haaste vastuunottamisessa liittyi projektin sekä testausprosessin yleisen kokonaisuuden hallintaan. Useampi haastateltava koki, että projekteista puuttui henkilö, joka vastasi siitä, että kokonaisuus on hallinnassa. Erityisesti haastatteluissa nousi esille tiimien välisten tai yhteisten työtehtävien hallinta, joka ei tuntunut olevan haastateltavien mukaan kenenkään vastuulla. Tämä aiheutti monesti epäselvyyksiä ja haasteita, sillä toisen tiimin tekemiset saattoivat vaikeuttaa tai jopa estää toisen tiimin työtehtävien suorittamisen. Eräs haastateltava mainitsi, että projektin sisäisesti ja kehitystiimien välillä puuttui selkeä etenemisen ja työtehtävien näkyvyys.

Ketterissä menetelmissä kehitystiimi on itsestään ohjautuvaa ja jokainen tiimin jäsen vastaa tuotostensa laadusta. Tästä huolimatta useampi haastateltava mainitsi, että he kaipaisivat kehitystiimiin tai projektiin henkilöä, joka ottaisi kantaa siihen, onko työtehtävät tehty riittävällä tasolla ja onko laadun taso asiakkaan tahtotilan mukainen.

Product ownerien työmäärät

Esiin nostettiin myös asiakasorganisaatioiden Product ownerien suuret työmäärät. “Monella PO:lla on liian monta hattua päässään”, totesi eräs haastateltava. Suuret työmäärät ja useat päällekkäiset projektit vaikuttivat suoraan siihen, että PO ei välttämättä ehtinyt perehtyä jokaiseen projektiin tarvittavalla tasolla. Tämä taas heijastui suoraan kehitystiimien

työskentelyyn; kiireisillä product ownereilla ei esimerkiksi ollut aikaa paneutua syvällisemmin yksittäisen projektin avoimiin kysymyksiin tai vastauksia jouduttiin odottamaan.

8.3 Kommunikaatio

Tiimien välinen kommunikointi

Useampi haastateltava koki tiimien välisen kommunikoinnin ongelmalliseksi. Joko viestintää tiimien kesken ei ollut lainkaan tai se oli hyvin vähäistä. Erään haastateltavan mukaan kehitystiimit eivät lainkaan kommunikoineet keskenään, mikä johti siihen, että toisen tiimin työt saattoivat hankaloittaa tai jopa estää toisen tiimin työskentelyn. Sama haastateltava korosti tässä kohdin jo aiemmin mainittua kehitystiimien "kuplaa", jolla hän tarkoitti tilannetta, jossa kehitystiimi työskentelee itsenäisesti omien työtehtäviensä parissa pohtimatta lainkaan muiden kehitystiimien tilannetta tai työtehtäviä.

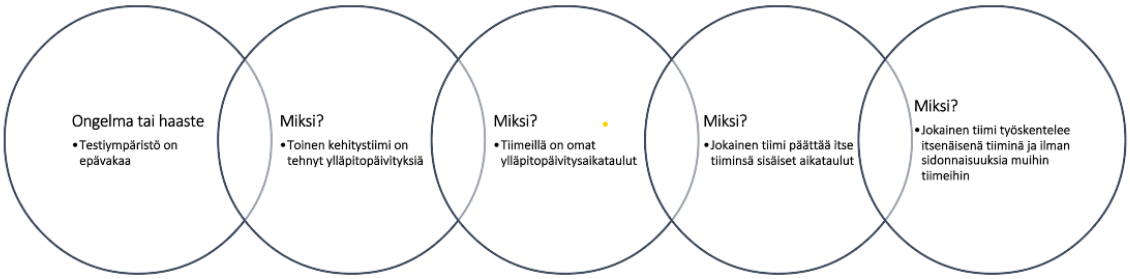
Päällekkäiset viestintäkanavat

Kaksi haastateltavista mainitsi, että viestintää vaikeutti useat päällekkäiset käyttäjätunnukset sekä viestintäsovellukset. Yksi haastateltava kertoi, että hänellä oli useampi työsähköposti ja hän oli osallisena useissa eri pikaviestintäsovelluksissa. Tämä oli johtanut projektissa siihen, että viestit tai palaverikutsut saattoivat jäädä huomaamatta. Myös informaation jako moniin eri kanaviin ja näin ollen eri viestintäkanavien säännöllinen seuraaminen, loi omat haasteensa muutaman haastateltavan mukaan.

Yksi haastateltava toivoi avoimempaa kommunikaatiota asiakasorganisaation ja kehitystiimin välille. Hän mainitsi esimerkkinä, että useamman kerran asiakas oli olettanut kehitystiimin automaattisesti ymmärtävän mitkä asiat ovat liiketoiminnallisesti oleellisia asioita ja mitkä ei. Samainen haastateltava kertoi myös, että asiakas ei selkeästi esittänyt toiveitaan tuotteen kehityksen suhteen. Toisaalta kehitystiimikään ei tuonut haasteltavan mukaan omia kehitysehdotuksiaan esille tai toiminut etupainotteisesti.

8.4 Juurisyyanalyysin tulokset

Kuvassa 10 on nähtävillä testausprosessiin liittyvän haasteen juurisyyanalyysi. Kuvassa nähdään alkuperäinen ongelma eli “testiympäristö on epävakaa”. Juurianalyysin tavoin “miksi” -kysymystä esitettiin niin monta kertaa, että perimmäinen ongelma oli tullut esille. Tässä kyseisessä tapauksessa juurisyy oli tiimien siiloutuminen ja omassa “kuplassaan” työskentely.

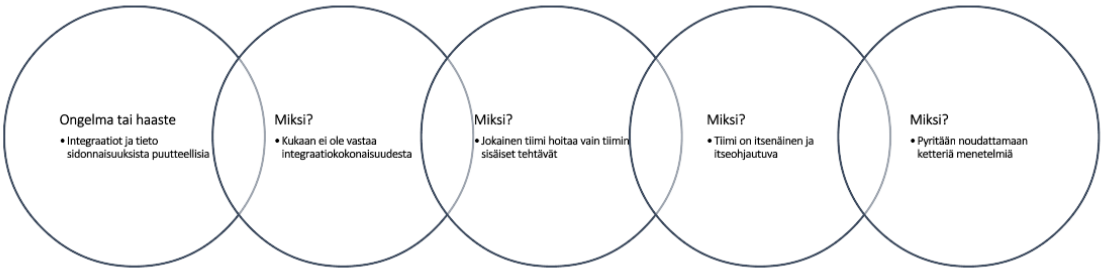


Kuva 10. Testiympäristön epävakauden juurisyyanalyysi.

Kuvassa 11 on epäselvät määritykset -juurisyyanalyysi. Alkuperäinen ongelma tai haaste oli määritysten riittämättömät määritykset. Riittämättömyyden syyksi todettiin se, ettei määrityksiä oltu määritelty kehitykselle ja testaukselle tarpeellisen tarkalla tasolla. Syy tähän oli asiakkaan epäspesifit toiveet. Juurisyyanalyysin perusteella todettiin, että perimmäinen syy epäselville määrityksille oli kommunikaation heikkous kehitystiimin ja asiakkaan välillä, sillä kehitystiimi ei ollut tehnyt asiakkaalle relevantteja kysymyksiä määrityksiin liittyen. On tärkeää huomata, että asiakas ei välttämättä osaa pukea sanoiksi tahtotilaansa eikä voida olettaa, että asiakas osaisi ottaa kantaa teknisiin asioihin. Tämän vuoksi kehitystiimin on itse työskenneltävä etupainotteisesti, asiakkaan tasoisesti sekä esitettävä tarpeelliset kysymykset määritysprosessien aikana.

Kuva 11. Epäselvät määritykset -juurisyyanalyysi.

Kuvassa 12 on integraatioihin ja sidonnaisuuksiin liittyvän ongelman juurisyyanalyysi. Alkuperäinen ongelma oli, että integraatiot ja tieto sidonnaisuuksista esimerkiksi tietojärjestelmien tai tiimien työtehtävin välillä oli puutteellista tai kokonaan olematonta. Syyksi tälle selvisi se, ettei kukaan varsinaisesti vastannut kyseistä kokonaisuudesta. Se miksi näin oli, johtui siitä, että jokainen tiimi hoitaa vain tiimilleen kuuluvia työtehtäviä ja esimerkiksi integraatiot eivät ole suoranaisesti tiimin sisäinen asia. Perimmäinen syy tähän toimintatapaan liittyy ketterien menetelmien noudattamiseen; koska työtehtävät halutaan suorittaa nopeissa sykleissä oman tiimin sisällä, ja tiimien sekä työntekijöiden itsenäisyyttä korostetaan, unohdetaan keskittyä tiimien välisiin sidonnaisuuksiin sekä tässä tapauksessa integraatioihin.



Kuva 12. Integraatio ja sidonnaisuus ongelma –juurisyyanalyysi.

Juurisyyanalyysi toistettiin myös muiden haastatteluissa esiin nousseiden haasteiden ja ongelmien kanssa. Analyysien perusteella suurimpana haasteena eri projekteissa oli kokonaisuuden hallinta ja kommunikaation puute tiimien kesken.

9 Testausprosessin kehittäminen

Haastatteluissa nousi selkeästi esille kolme erillistä kokonaisuutta, joiden kanssa eri projekteissa koettiin haasteita. Nämä kolme kokonaisuutta olivat ketterien menetelmien toteuttaminen käytännössä, testausprosessin ja kokonaisuuden hallinta sekä erilaiset ongelmat kommunikaatiossa.

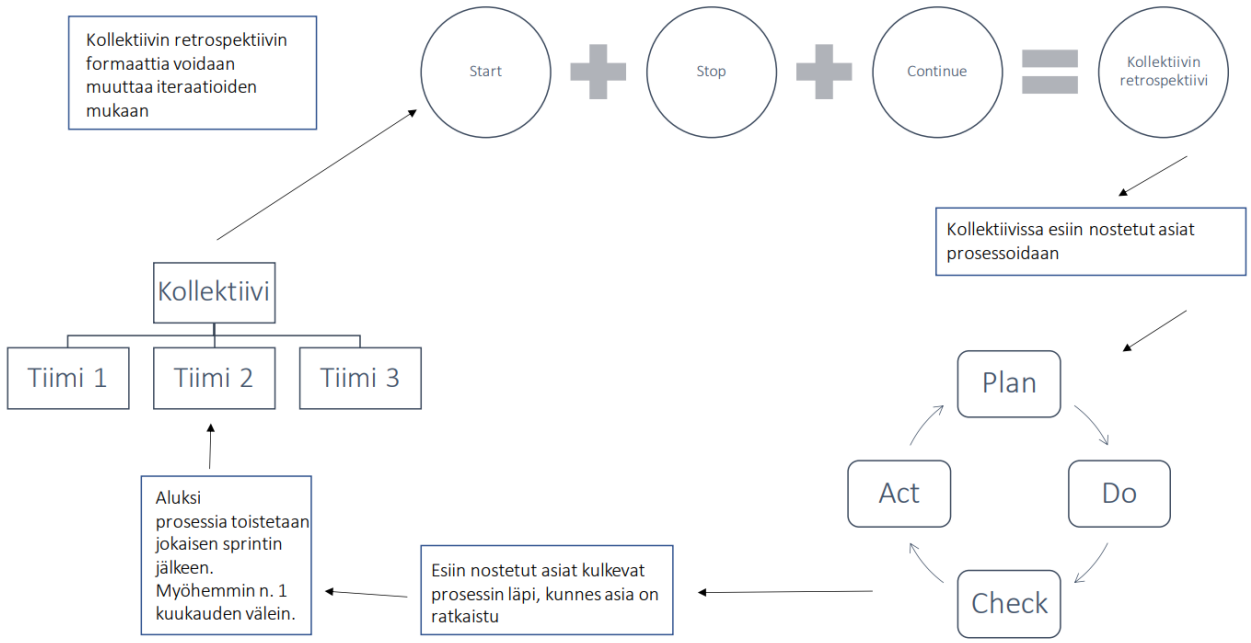
Aihekokonaisuuksista valittiin kehittämisen kohteeksi testausprosessin ja kokonaisuuden hallinta ja kommunikaation parantaminen. Valinta tehtiin siitä syystä, että ketterien menetelmien käyttöönottoon ja käytäntöjen korjaamiseen on olemassa ketterät valmentajat, joiden tehtävänä on puuttua tämän työn tutkimuksessa esiin tulleisiin ketterien menetelmien ongelmiin. Ketteristä valmentajista kerrotaan lisää tämän luvun lopussa.

9.1 Kollektiivien retrospektiivi malli

Haastattelujen perusteella eri projektien suurimmat haasteet liittyivät kommunikointiin ja kokonaisuuden hallintaan. Haastatteluissa esiin tulleiden asioiden pohjalta luotiin kollektiivien retrospektiivi malli, jonka avulla voidaan tarttua helpommin sekä kokonaisuuden hallintaan että kommunikaatiovajeen aiheuttamiin haasteisiin. Kollektiivilla tarkoitetaan tässä yhteydessä ryhmää tai tiimiä, joka työskentelee yhdessä kohti yhteistä tavoitetta. Retrospektiivi on tilaisuus, jossa tarkastellaan lähimenneisyyttä ja analysoidaan onnistumisia ja epäonnistumisia.

Kuvassa 13 on nähtävillä kollektiivien retrospektiivi malli kokonaisuudessaan. Kollektiivien retrospektiivin prosessi käy läpi kolme pienempää prosessia, jotka käydään yksitellen seuraavaksi läpi.

Prosessia toistetaan, kunnes on saavutettu tilanne, jossa käsiteltäviä asioita ei enää nouse esille. Kollektiivien retrospektiivejä on suotavaa aluksi järjestää jokaisen sprintin jälkeen, jotta sprintin aikaiset haasteet tulevat ilmi. Kun prosessi on saatu onnistuneesti implementoitua projektiin tai mikäli käsiteltävien asioiden määrä vähenee huomattavasti, voidaan retrospektiivien ajankohta muuttaa tapahtuvaksi kuukauden välein.



Kuva 13. Kollektiivien retrospektiivi.

Kollektiivi

Kollektiivien retrospektiivin prosessi alkaa kollektiivien valinnalla (kuva 14). Jokaisesta projektin tiimistä valitaan yksi henkilö edustamaan tiimiä. Henkilö voi vaihtua tilanteen mukaan eikä henkilön tarvitse olla esimerkiksi byrokraattisella tasolla tiimin vetäjä.

Tiimeistä valitut henkilöt muodostavat kollektiivin, joka kokoontuu läpikäymään tiimien sisällä esiin nousseita ongelmia ja onnistumisia.

Kuva 14. Kollektiivi.

Retrospektiivi

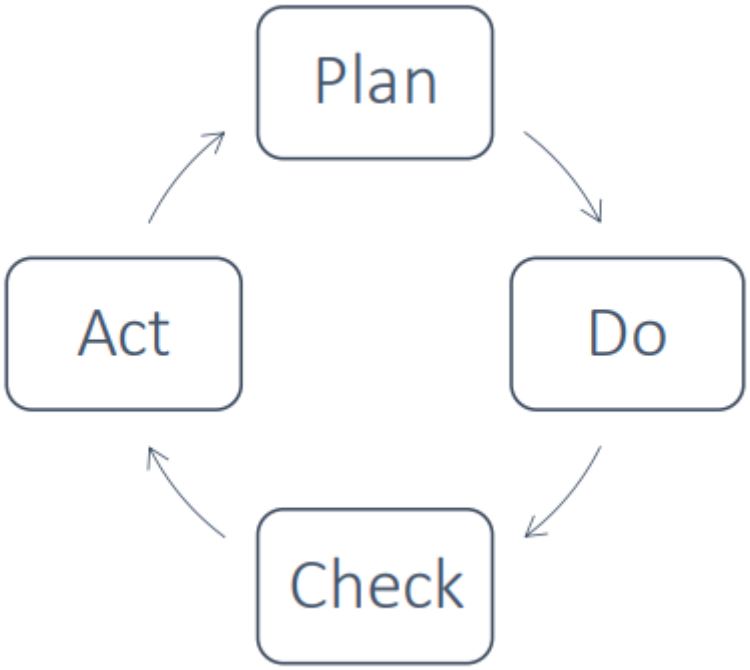
Tiimien kollektiivi kokoontuu retrospektiiviin keskustelemaan edellisen sprintin onnistumisista ja epäonnistumisista. Erityisesti esille tuodaan asiat, joihin liittyy tiimien yhteistyö tai työtehtävät, joiden suorittaminen vaikuttaa toisen tiimin työskentelyyn. Tapaamiset toimivat myös informaatiokanavana tiimien välillä.

Kollektiivin retrospektiivin formaattia voidaan muuttaa käsiteltävien asioiden mukaan. Kuvassa 15 on esitetty Start-Stop-Continue –retrospektiivimalli. Kyseissä mallissa käydään läpi mitä asioita seuraavassa sprintissä ryhdytään työstämään (Start), mitä asioiden työstäminen lopetetaan (Stop) sekä mitä asioiden työstäminen vielä jatkuu (Continue) (Gregory & Crispin 2015, 445.).

Kuva 15. Retrospektiivi

PDMA-menetelmän implementointi

Retrospektiivissä esiin nousseet haasteet ja ongelmat prosessoidaan PDCA-menetelmän mukaisesti; suunnitellaan mitä asialle tehdään, toimitaan suunnitelman mukaan, tarkistetaan, miten suunnitelma on toteutunut ja lopuksi muutetaan toimintatapaa, mikäli tarpeellista (kuva 16). PDCA-menetelmää toistetaan per haaste tai ongelma, kunnes asia on ratkaistu kaikille hyväksyttävälle tasolle.



Kuva 16. PDCA-malli.

Prosessimallin tarkoitus ja hyödyt

Kollektiivi retrospektiivin tarkoituksena on ensisijaisesti toimia informointikanavana ja tiimien välisen siiloutumisen sekä informaatiokuilun poistajana. Kollektiivi retrospektiivissä esiin nousevien haasteiden läpikäynnin myötä tiimien välisten työtehtävien

sidonnaisuuksien ymmärtäminen kasvanee. Oikein suoritettuna prosessi vähentää myös ongelmien syntymistä ja madaltaa mahdollisten haasteellistenkin asioiden käsittelyä.

9.2 Ketterä valmentaja

Haastatteluissa tuli ilmi, että ketterin menetelmien toteuttamisessa on ajoittain haasteita asiakasorganisaatioissa tai yksittäisissä projekteissa. Haastattelujen perusteella haasteet liittyvät esimerkiksi rooleihin ja työtehtäviin, epäselviin vastuujakoihin tai väärin tehtyihin työarvioihin per iteraatio. Tämän kaltaisissa tilanteissa olisi suotavaa hyödyntää ketterien valmentajien eli agile coachien osaamista.

Ketterä valmentaja on fasilitaattori, joka auttaa yksilöitä ja organisaatioita siirtymään ketterään työskentelytapaan. Ketterät valmentajat opastavat ja tukevat tiimejä sekä yksilöitä sopeutumaan uusiin työskentelytapoihin, auttavat mahdollisissa haasteissa ja jakavat tietoa ketterien menetelmien eduista. (Adkins 2010, 32–37; Ward 2021, 78–82)

Ketterä valmentaja on henkilö, joka auttaa tiimejä ja organisaatioita siirtymään ketterään työskentelytapaan ja omaksumaan sen. He auttavat ihmisiä ymmärtämään ja omaksumaan ketterät arvot ja periaatteet sekä antavat ohjausta ja tukea ketterien käytäntöjen soveltamiseen tavalla, joka toimii parhaiten tiimille tai organisaatiolle. Yrityksen sisäinen ketterä valmentaja työskentelee organisaatiotasolla ja tekee yhteistyötä eri osastojen välillä, kehittää johtajia ja laajentaa ketteryyttä helpottaen organisaation visiota ja strategisia tavoitteita. (Adkins 2010, 32–37; Ward 2021, 78–82.)

Ketterän valmentajan osaamisen hyödyntäminen on suotavaa toimeksiantajan projekteissa, joissa on kohdattu haasteita ketterien menetelmien implementoinnissa. Ketterän valmentajan avulla projekteissa voidaan paneutua projektikohtaisiin haasteisiin ja korjata toimintatavat ketterien menetelmien mukaiseksi.

10 Johtopäätökset

Tässä luvussa käsitellään opinnäytetyön tutkimuskysymyksiä ja sitä, miten tehty tutkimus onnistui vastaamaan asetettuihin tutkimuskysymyksiin. Johtopäätökset esitetään määriteltyjen tutkimuskysymysten mukaisessa järjestyksessä.

10.1 Tutkimuskysymyksiin vastaaminen

Mitä on ketterä ohjelmistonkehitys ja miten testaus toteutetaan ketterässä ohjelmistokehityksessä?

Ketterällä ohjelmistokehityksellä tarkoitetaan nopea tempoista työskentelytapaa, jossa työtä suoritetaan yleensä syklittäin. Ketterässä ohjelmistokehityksessä ihmiskeskeisyys on keskiössä ja ketterän manifestin mukaan jatkuvaan kommunikointiin ja kanssakäymiseen tulisi panostaa enemmän kuin työkaluihin ja prosesseihin. (Huttunen 2006, 15–16; ISTQB 2015, 8–9.)

Ketterän testauksen prosessi etenee kehityksen mukaisesti syklittäin. Yhden iteraation testausprosessi alkaa vaatimusten analysoinnilla ja valinnalla, jonka jälkeen suunnitellaan testaus toteutettavaan toiminnallisuuteen. Toteutuksen valmistumisen jälkeen suoritetaan manuaalista ja automatisoitua testausta yhdistelemällä yksikkö-, integraatio-, järjestelmä- ja hyväksymistestaus. Julkaisun jälkeen käynnistyy uusi iteraatio ja testausprosessi alkaa alusta. (Perry 2006.)

Haastatteluissa ilmeni, että ketterät menetelmät ovat käytössä asiakasprojekteissa. Ketterien menetelmien käyttöönotossa on kuitenkin eroja projektien kesken ja käytännöt ovat projektikohtaisia. Järjestelmänkehityksen toteutustasolla ketterät menetelmät oli otettu hyvin käyttöön, mutta testauksessa ja kokonaisuuden hallinnassa menetelmien käyttö heikkeni.

Erään haastateltavan mukaan kahden viikon syklit tuottivat manuaaliselle testaukselle haasteita, joita lisäsi entisestään se, että testiympäristö oli ajoittain epävakaa. Haastatteluissa kävi ilmi, että automaatiotestausta pyritään lisäämään entisestään tulevaisuudessa ja tutkivan testauksen osuutta tullaan nostamaan manuaalisen testauksen jäädessä pois.

Automaatiotestaus huomioidaan jo määrittelyvaiheessa; tarkoituksen mukaista on, että määritykset on laadittu niin tarkalla tasolla, että automatisointi scriptit voidaan luoda suoraan sovelluksen end-to-end -tasolle. Tämä tapa lisää laadunvarmistusta, mutta myös vähentää työtaakkaa automatisoinnin myöhemmissä vaiheissa.

Kaiken kaikkiaan voidaan todeta, että testaus on ketterissä ohjelmistokehitysprojekteissa suurien muutosten edessä. Manuaalista testausta pyritään vähentämään, sillä se on aikaa vievää. Toisaalta tutkivaa testausta lisätään, sillä automaatiotestaus ei siihen kykene. Oman lukunsa testauksen kokonaisuuteen tuo tekoäly, jonka suomat mahdollisuudet voivat muuttaa myös osaltaan sen, miltä testaus tulee tulevaisuudessa näyttämään ketterissä ohjelmistoprojekteissa.

Mitkä ovat onnistuneen testauksen edellytykset ja haasteet?

Onnistunutta laadunvarmistusta voidaan tarkastella kahdesta eri näkökulmasta; asiakkaan ja kehitystiimin. Asiakkaan näkökulmasta testaus on onnistunutta, mikäli ohjelmisto toimii asiakkaan toimimalla tavalla ja siinä ei ole kriittisiä tai asiakasta häiritseviä virheitä. Kehitystiimin näkökulmasta onnistunut laadunvarmistus voi tarkoittaa selkeitä määrityksiä, yksinkertaista koodia ja testattavissa olevaa ohjelmistoa. (Ashfaque, 2010; ISTQB 2018, 65–66.)

Tässä työssä keskityttiin kehitystiimin näkökulmaan. Laadunvarmistuksen haasteeksi haastatteluissa mainittiin muun muassa epävakaa tai kokonaan toimimaton testiympäristö, kiire aikataulu ja suunnitelmallisuuden puuttuminen.

Onnistunut laadunvarmistus vie aikaa. Nykyaikana prosessien on liikuttava nopeasti ja asiakas toivoo ripeitä toimituseriä. Väistämättä se näkyy jossakin, usein laadunvarmistuksessa. Jotta laatu pysyisi tasaisena tulisi kiinnittää huomiota kahteen asiaan; selkeisiin ja täsmällisiin määrityksiin sekä testaukseen annettavaan aikaan. Tähän voidaan vaikuttaa muun muassa työmäärältään lukituilla iteraatioilla.

Selkeä kommunikaation puute on suurin riski viestinnän epäonnistumiselle. Myös haluttomuus jakaa tietoa tai huomioitaan muiden tiimiläisten kanssa voi hankaloittaa ja hidastaa työtehtävien etenemistä. (ISTQB 2018, 22–23.) Tämän työn tutkimuksen mukaan useissa projekteissa oli eritasoisia ongelmia viestinnän kanssa; joko viestintätyökaluja oli liikaa tai

informaatiota ei jaettu lainkaan. On ongelma tai haaste minkälainen vain, tulisi siihen puuttua ensi tilassa. Mikäli viestintäongelmat vaikeuttavat työtehtävissä suoriutumista tai jopa estävät sen, on viestinnän epäonnistumisen riski toteutunut.

Koska viestinnällä on suuri merkitys projektin onnistumiseen sekä ketterässä tiimissä työskentelyyn (Baumgartner et la 2021, 39), tulisi viestintäsuunnitelman laadintaan panostaa heti projektin alussa.

Miten testauksen toteutusta voi kehittää?

Ketterissä menetelmissä käytetään usein PDMA-mallin muunnelmia, kun käsitellään edellisissä iteraatioissa hyvin ja huonosti menneitä asioita. Ketterän ohjelmistokehityksen manifestissa painotetaan, että muutokset ovat tervetulleita. Tällä tarkoitaan myös projektin sisäisten toimintatapojen muuttamista, mikäli huomataan, että asian voi tehdä paremmin tai tehokkaammin. (ISTQB 2015, 8–9; Kinsbruner 2018, 118; Torkkola 2015, 72.)

Tutkimuksessa ei näkynyt selkeitä viitteitä siihen, että projekteissa pyrittäisiin korjaamaan huonoja toimintatapoja; esimerkiksi juurisyyanalyysin tulokset olivat monelta osin jo ennestään projektien tiedossa, mutta asioihin ei silti oltu puututtu. Pieniä korjausliikkeitä oli nähtävillä ja joihinkin asioihin oli tehty muutoksia jo heti haastattelukierroksen päätyttyä. Kuitenkin suurimmat haasteet ja ongelmat jäivät projekteissa käsittelemättä osaksi kiireen ja osaksi oletettavasti osaamattomuuden vuoksi. Ongelmat, jotka liittyvät suoraan ketterien menetelmien käyttöönottoon ja suorittamiseen, tulisi käydä läpi ketterän valmentajan kanssa. Tarvittavat muutokset työskentelytapoihin olisi hyvä saada implementoitua arkeen mahdollisimman pian, sillä projektin haasteilla on suora vaikutus projektien kestoon ja kustannuksiin.

Tiimien väliseen kommunikaatiovajeeseen ei myöskään projekteissa tartuttu, vaikka testaajat olivat tuoneet asian esille useampia kertoja. Haastatteluissa tuli selkeästi ilmi kiire, joka edesauttoi asioihin puuttumattomuutta. Kollektiivisen retrospektiivin avulla kommunikaatiota saisi lisättyä tiimien välille. Myös erinäiset haasteet tulisi kaikkien tietoon, koska kollektiivinen retrospektiivi toimii myös informaatiokanavana. Kollektiivin retrospektiivin mahdollistaman yhteistyön avulla myös työskentelytapojen muutoksiin ja kehitystarpeisiin olisi helpompi tarttua.

Kuten edellisessä kohdassa korostettiin, on avoimeen viestintään ja sujuvaan kommunikaatioon panostettava. Ei ole suotavaa olettaa, että asiat luonnistuvat itsekseen ilman panostusta. Viestintäsuunnitelman laadinta projektin alussa luo raamit projektin aikaiselle viestinnälle.

Tulevaisuudessa automaatiotestaus tulee korostumaan entistä enemmän ja manuaalinen testaus tulee väistämättä jäämää taka-alalle. Se, onko tämä täysin hyvä asia, riippuu keneltä asiaa kysyy. On kuitenkin huomioitava, että tutkivan testauksen osuutta on hyvä olla testausprosessissa mukana, sillä sen kaltaista automaatiotestaus ei voi kattaa.

Manuaalitestauksen suorittamiseen sinällään ei tarvita kehitystoimenpiteitä. Kehittämiskohteet ovat enemmänkin tiimien välisessä kommunikaatiossa, jotta testiympäristö pysyy luotettavan testauksen suorittamiselle.

10.2 Tutkimuksen rajoitteet

Teoreettinen viitekehys

Työn teoreettinen viitekehys on koottu laajalla ja yleispätevällä otteella työn aihepiiristä. Työn pääasiallisista aiheista, ketteristä menetelmistä sekä tietojärjestelmän testauksesta, on olemassa kattavasti kirjallisuutta ja teoreettinen viitekehys olisi voitu laatia myös huomattavasti laajemmin tai syvällisemmin kuhunkin aihealueeseen paneutuen. Syvällisempi paneutuminen ei olisi kuitenkaan tuonut työlle lisäarvoa eikä sillä olisi ollut merkittävää vaikutusta tutkimusprosessille tai työn tuotokselle.

Tutkimuksen tulokset

Diplomityön tutkimus rajoittui toimeksiantajan asiakasorganisaatioihin ja heidän projekteihinsa. Tämän vuoksi haastatteluissa esiin tulleita asioita, tai tämän työn johtopäätöksiä ei voi yleistää.

Luotettavuus

Haastattelut suoritettiin verrattain pienelle otannalle. Haastateltavia oli yhteensä kuusi kappaletta. Voidaan olettaa, että mikäli haastateltavia olisi ollut enemmän, olisi erilaisia haasteita voinut nousta lisää esiin. On kuitenkin huomioitava, että jo kuuden haastateltavan haastattelujen sisällöt alkoivat toistaa toisiaan, vaikka henkilöt työskentelivät eri projekteissa. Näin ollen saturaatiopiste saavutettiin haastatteluissa eikä haastateltavien suurempi lukumäärä olisi tuonut suurta lisäarvoa tutkimukselle.

10.3 Jatkotutkimusaiheet

Tutkimuksessa ilmeni, että suurimmat haasteet liittyvät projektin sisäiseen kommunikointiin sekä projektien kokonaisuuden hallintaan. Myös tiimien siiloutuminen koettiin suurena ongelmana.

Tiimien siiloutumisen vaikutuksia projektin etenemiseen sekä projektin sisäiseen laatuun tulisi tutkia enemmän. Myös siiloutumisen vaikutukset niin kommunikointiin kuin tiimiläisten yhteenkuuluvuuteen ovat varteenotettavia tutkimuksen aiheita.

Lähteet

Adkins, L. 2010. Coaching Agile Teams. A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition. Pearson Education.

Andersen, B & Fagerhaug, T. 2006. Root Cause Analysis. Milwaukee. ASQ Quality Press.

Artto, K., Martinsuo, M. & Kujala, J. 2011. Projektiliiketoiminta. Helsinki.

Ashfaque, A. 2010. Software Testing as a Service. Auerbach Publications.

Baumgartner, M., Klonk, M., Mastnak, C., Pichler, H., Seidl, R. & Tanczos, S. 2021. Agile Testing. The Agile way to Quality. Springer.

Black, R. 2017. Agile Testing foundations: an ISTQB foundation level agile tester guide. BCS Learning & Development.

Claesson, A. 2022. Test Strategies in Agile Projects. EuroSTAR Huddle Software Testing & QA Learning Resource.

Cloudt, G. 2021. What is software quality? Understanding what really matters in software development. Independently published.

Elfriede, D. 2002. Effective Software Testing: 50 Specific Ways to Improve Your Testing. Addison-Wesley.

Ghahrai, A. Problems with Test Automation and Modern QA. https://devqa.io/problems-test-automation-modern-qa/

Gregory, J., & Crispin, L. 2015. Agile Testing: A Practical Guide for Testers and Agile Teams. Pearson Education

Haikala, I. & Märijärvi, J. 2004. Ohjelmistotuotanto, Talentum, Helsinki.

Hambling, B. 2015. Software testing: an ISTQB-BCS certified tester foundation guide. BCS Learning & Development.

Hirsjärvi, S & Hurme, H. 2001. Tutkimushaastattelu; Teemahaastattelun teoria ja käytäntö. Helsinki. Yliopistopaino.

Hoory, L. & Bottoff, C. 2022. Agile vs Waterfall: which project management methodology is best for you? Forbes Media LLC.

Huttunen, J. 2006. Ketterän ohjelmistokehitysmenetelmän määrittely, vertailu ja käyttäjäkysely. Teknillinen korkeakoulu.

Irshad, M., Börstler, J., & Petersen, K. 2022. Supporting refactoring of BDD specifications—An empirical study, Information and software technology, 2022–01, Vol.141, p.106717.

ISTQB. 2015. Foundation Level Syllabus Agile Tester. International Software Testing Qualifications Board.

ISTQB 2018. Sertifioitu testaaja. Perustason sertifikaattisisältö. International Software Testing Qualifications Board.

Kinsbruner, E. 2018. Continous Testing for Devops Professionals. ICGtesting

Madeyski, L. 2010. Test-Driven Development: An Empirical Evaluation of Agile Practice. Springer.

Mahfuz, A. 2021. Software Quality Assurance: Integrating Testing, Security and Audit. CRC Press.

Niemiaho, A. & Myllymäki, J. 2022. Covid-19-pandemian vaikutukset ICT-alalla ja kaivosteollisuudessa. Centria-ammattikorkeakoulu.

Perry, W. 2006. Effective methods for software testing. Wiley.

Pohjolainen, P. 2003. Ohjelmiston testauksen automatisointi. Kuopion yliopisto.

Roima, M. 2013. Käytännön ketterät menetelmät laaja-alaisessa ohjelmistokehityksessä. Hämeen ammattikorkeakoulu.

Rosenberg, D., Collins-Cope, M. & Stephens, M. 2005. Agile Development with ICONIX Process: People, Process, and Pragmatis. Apress.

Salminen, A. 2011. Mikä kirjallisuuskatsaus?. Opetusjulkaisuja 62. Vaasan yliopisto. https://www.uwasa.fi/materiaali/pdf/isbn_978-952-476-349-3.pdf

Smart, J. F. 2015. BDD in Action. Manning.

Todaro, D. 2019. The Epic Guide to Agile. R9.

Torkkola, S. 2015. Lean asiantuntijatyön johtamisessa. Alma Talent.

Tuomi, J. & Sarajärvi, A. 2013. Laadullinen sisällönanalyysi. Helsinki. Tammi.

Tuomi-Heinonen, N. 2008. Testauksen käytäntö Scrumissa. Jyväskylän ammattikorkea-koulu.

Watkins, J., & Mills, S. 2011. Testing IT: An Off-the-Shelf Software Testing Process. Cambridge University Press.

Ward, J. 2021. The Agile Coach's Cookbook. Austin Macauley Publishers Limited.