Juuso Järvi

KETTERÄN OHJELMISTOKEHITYKSEN MENESTYS-TEKIJÄT

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2018

TIIVISTELMÄ

Järvi, Juuso
Ketterän ohjelmistokehityksen menestystekijät
Jyväskylä: Jyväskylän yliopisto, 2018, 79 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaaja: Ojala, Arto

Tässä tutkielmassa tarkasteltiin ketterää ohjelmistokehitystä ja ketterän ohjelmistokehityksen menestystekijöitä. Vaikka ketterä ohjelmistokehitys yhdistetään yhä suuremmissa määrin onnistuneisiin ohjelmistokehitysprojekteihin, on ilmiö kuitenkin vaikeasti määriteltävissä: aiheen tutkimus on epäselvää, suuri osa akateemisista artikkeleista ei perustu teorialle ja täten tutkimustieto ei ole kovinkaan yhtenäistä, minkä takia aiheen tutkiminen on mielekästä. Tässä tutkielmassa ketterää ohjelmistokehitystä tutkittiin kriittisten menestystekijöiden konseptin näkökulmasta. Tutkielman päätutkimuskysymyksenä, johon haettiin vastausta laadullisen teemahaastattelun keinoin, oli: ”Mitkä ovat ketterän ohjelmistokehityksen kriittisiä menestystekijöitä?”. Tutkimusongelmaa pyrittiin valottamaan myös kysymyksillä ”Miten ketterä ohjelmistokehitys on määritelty akateemisessa kirjallisuudessa?” ja ”Miten yritys voi saavuttaa ketteryyttä?”, joihin vastattiin kirjallisuuskatsauksen avulla. Empiirisen tutkimuksen tulosten pohjalta tunnistettiin seitsemän ketterän ohjelmistokehityksen menestystekijää: asiakasyhteistyö, kommunikointi, julkaisustrategia, kompetenssi, harjoittelu ja oppiminen, yrityskulttuuri sekä päätöksenteon nopeus. Tuloksien valossa tämä tutkimus tuo oman näkemyksensä ketterän ohjelmistokehityksen menestystekijöistä aikaisemmin toteutettujen tutkimuksien rinnalle, vahvistaen osaa aikaisemmista havainnoista ja haastaen osan niistä. Empiirisen tutkimuksen tulosten pohjalta tunnistetut seitsemän menestystekijää antavat ketterää ohjelmistokehitystä tekeville yrityksille mahdollisuuden peilata omaa toimintaansa suhteessa esitettyihin menestystekijöihin ja kehittää ohjelmistokehityksensä ketteryyttä tämän pohjalta.

Asiasanat: ketterä ohjelmistokehitys, ketterät menetelmät, menestystekijät, Scrum, XP, Lean, Kanban

ABSTRACT

Järvi, Juuso
The success factors of agile software development
Jyväskylä: University of Jyväskylä, 2018, 79 p.
Information Systems Science, Master's Thesis
Supervisor: Ojala, Arto

This research studied agile software development and the success factors of agile software development. Although software development agility has been considered as a crucial predecessor of software development project success, there is ambiguity surrounding the subject: the research seems vague and there is considerable amount of academic literature which is not based on sound theory. Hence research on this subject seems reasonable. In this research agile software development was studied through the concept of critical success factors. The main research question was "What are the critical success factors of agile software development?". In addition to the main question, two supporting questions were presented: "How is agile software development defined in academic literature?" and "How can an organization attain agility?". The latter two were answered through literature review and the empirical data to answer to the main research question was gathered through qualitative interviews. Seven success factors were identified: customer collaboration, communication, delivery strategy, competency, training and learning, corporate culture and decision time. This research provides a new viewpoint on success factors of agile software development along with previous research on this matter. The research confirms some previous findings and questions some of them. Based on the results of the empirical study, organizations performing agile software development are able to evaluate their operations according to the proposed success factors and develop their operations accordingly.

Keywords: agile software development, agile methods, success factors, Scrum, XP, Lean, Kanban

KUVIOT

TAULUKOT

SISÄLLYS

1 JOHDANTO

Tässä tutkielmassa tarkastellaan ketterää ohjelmistokehitystä ja ketterän ohjelmistokehityksen menestystekijöitä. Ohjelmistokehityksen ketteryys on jo pitkään kiinnostanut IT-alan asiantuntijoita sekä tutkijoita. Ketterän ohjelmistokehityksen julistus (engl. Manifesto for Agile Software Development) julkaistiin vuonna 2001 (Agile Alliance, 2001) ja viime vuosina juuri ohjelmistokehityksen ketteryyttä on pidetty yhtenä tärkeimmistä ohjelmistoprojektien onnistumisen edellytyksistä (Standish Group International, 2013).

Ketterä ohjelmistokehitys on ilmiönä hankalasti määriteltävissä. Conboyn (2009) mukaan ketterien menetelmien luomisesta ja niiden käytön edistämisestä ovat ilmiön alkuaikoina vastanneet lähinnä asiantuntijat ja konsultit, mikä on osaltaan johtanut tieteellisen tutkimuksen epäselvyyteen, heikkoon yhteneväisyyteen ja rajoittuneeseen sovellettavuuteen. Myös Dingsøyr, Nerur, Balijepally ja Moe (2012) kaipaavat yhdenmukaisuutta ketterän ohjelmistokehityksen tutkimukseen. Heidän mukaansa juuri Conboy (2009) on määritellyt ohjelmistokehityksen ketteryyden kattavimmin:

> Tietojärjestelmäkehityksen metodin jatkuva valmius luoda muutosta, proaktiivisesti tai reaktiivisesti vastaanottaa ja hyväksyä muutos ja oppia muutoksesta samalla lisäten asiakkaan kokemaa arvoa (taloudellinen, laadullinen ja yksinkertaisuus) hyödyntäen sen osia ja suhteita ympäristöön. (Conboy, 2009, s. 340)

Tutkimuksen näkökulmasta on oleellista erottaa ketterän toiminnan periaatteet ketteristä menetelmistä. Ketterät periaatteet perustuvat Ketterän ohjelmistokehityksen julistukseen. Ne eivät ole virallinen ketteryyden määritelmä, vaan pikemminkin ne tarjoavat suuntaviivoja siihen, miten laadukasta ohjelmistoa voidaan tuottaa ketterällä tavalla. (Dingsøyr ym., 2012.) Cohenin, Lindvallin ja Costan (2004) mukaan ketterät menetelmät koostuvat tekniikoista ja käytänteistä, jotka pohjautuvat samoihin arvoihin ja perusperiaatteisiin: iteratiiviseen kehittämiseen, vuorovaikutukseen, kommunikaatioon ja pyrkimykseen vähentää resursseja vieviä väliaikaisia tuotoksia, jotka eivät lisää arvoa lopputuotokseen.

Ketteriä menetelmiä ovat mm. Feature-Driven Development, Lean Software Development, Scrum ja Extreme Programming (Dybå & Dingsøyr, 2008).

John Rockart esitteli kriittisten menestystekijöiden (engl. Critical Success Factors) konseptin vuonna 1979 (Rockart, 1979). Alun perin konseptia hyödynnettiin johtamisen alueella määrittelemään sellaisia tekijöitä, joista liikkeenjohdolla tulisi olla saatavilla kaikki tarpeellinen informaatio. Kriittiset menestystekijät ovat tärkeitä toiminnan avainalueita, joissa tarvitaan suotuisia tuloksia, jotta liikkeenjohdon tavoitteet voidaan saavuttaa – näillä alueilla asioiden tulee mennä oikein, jotta liiketoiminta voi kukoistaa (Bullen & Rockart, 1981). Chowin ja Caon (2008) mukaan formaalia tutkimusta kriittisistä menestystekijöistä ketterän ohjelmistokehityksen kontekstissa ei ole aiemmin tehty. Sen sijaan esimerkiksi tapaustutkimuksia onnistumisista ja epäonnistumisista ketterissä ohjelmistoprojekteissa on olemassa (Chow & Cao, 2008). Tässä tutkielmassa tutkimusongelman ulkopuolelle rajattiin menestystekijät ketterään ohjelmistokehitykseen siirtyessä (engl. agile transformation, agile implementation, agile adaption), koska näiden katsottiin määrittävän eri ilmiötä kuin ketterän ohjelmistokehityksen hyödyntämisessä esiintyvien menestystekijöiden. Myöhemmin tutkimusta menestymiseen vaikuttavista tekijöistä ketterässä ohjelmistokehityksessä ovat tehneet muun muassa Misra, Kumar ja Kumar (2009), França, Silva ja Sousa Mariz (2010) sekä Stankovic, Nikolic, Djordjevic ja Cao (2013).

Jotta ketterää ohjelmistokehitystä voidaan tehdä onnistuneesti, tulee tietää ketterän ohjelmistokehityksen tärkeät menestystekijät. Tämän tutkielman tutkimuskysymyksenä on:

> Mitkä ovat ketterän ohjelmistokehityksen kriittisiä menestystekijöitä?

Ketterän ohjelmistokehityksen aihepiirin sekä tutkimusongelman ymmärtämiseksi tulee tutkia myös ketteryyden määritelmää ja periaatteita, käytänteitä sekä menetelmiä, joilla ketteryyttä voidaan saavuttaa. Täten tutkielman tutkimusongelmaa pyritään valottamaan myös kysymyksillä ”Miten ketterä ohjelmistokehitys on määritelty akateemisessa kirjallisuudessa?” ja ”Miten yritys voi saavuttaa ketteryyttä?”

Tutkielman ensimmäisessä osassa pyritään kirjallisuuskatsauksen avulla vastaamaan kahteen jälkimmäiseen tutkimuskysymykseen sekä valottamaan aikaisempien tutkimuksien tuloksia varsinaisen tutkimuskysymyksen osalta. Lähdeaineiston hakeminen aloitettiin Dybån ja Dingsøyrin (2008) sekä Dingsøyrin ym. (2012) systemaattisista kirjallisuuskatsauksista ja näiden lähteistä. Erityisesti Dingsøyrin ym. (2012) artikkelissa esitetty analyysi tutkijoista, jotka ovat kirjoittaneet merkittävimpiä artikkeleita aihepiiristä, osoittautui hyväksi lähtökohdaksi. Löydettyjen artikkelien lähteistä saatiin yhä enemmän lähdeaineistoa ja uudempia artikkeleita. Myös menestystekijöitä käsittelevää tutkimusta kartoitettiin tutkimustietokannoista. Tutkimusongelmaa kartoittavassa kirjallisuudessa keskityttiin vain empiirisiin tutkimuksiin, joissa on operationalisoitu mahdollisia menestystekijöitä, minkä jälkeen näitä on tutkittu kvantitatiivisin menetelmin. Tällöin lähdeaineiston ulkopuolelle rajautuivat artikkelit, joissa ehdotetaan ketterän ohjelmistokehityksen parhaita käytänteitä

(engl. best practices), sillä niiden ei katsottu tarjoavan tarpeeksi vankkaa tieteellistä pohjaa tutkimusongelman tarkasteluun.

Tutkielman toisessa osassa pyritään vastaamaan varsinaiseen tutkimuskysymykseen empiirisellä tutkimuksella. Aineisto kerättiin laadullisilla teemahaastatteluilla, joissa haastateltiin yhteensä kuutta ketterää ohjelmistokehitystä tekevän yrityksen edustajaa. Toisessa osassa esitellään ensin valittu tutkimusmenetelmä, aineiston analysointimenetelmä sekä empiirisen tutkimuksen tulokset. Tämän jälkeen esitetään pohdintaa tutkielman empiiristen tulosten ja aikaisempien tuloksien eroista sekä yhteneväisyyksistä. Lopuksi tehdään yhteenveto tutkielman tuloksista.

Tutkielma on jäsennetty kahdeksaan lukuun. Luvussa 2 esitetään ketterän ohjelmistokehityksen määrittelyjä käyttäen pohjana sekä Ketterän ohjelmistokehityksen julistusta että akateemisessa kirjallisuudessa esitettyjä määritelmiä. Luvussa 3 kuvataan tarkemmin ketterät periaatteet, käytänteet ja neljä ketterää menetelmää: Scrum, XP, Lean-ohjelmistokehitys ja Kanban. Luvussa 4 tarkastellaan aikaisempia määrällisiä tutkimuksia ketterän ohjelmistokehityksen menestystekijöistä. Luvussa 5 kerrotaan laadullisen tutkimuksen toteutuksesta. Luvussa avataan tutkimusmenetelmän ja aineistonkeruumenetelmän valintaa sekä aineiston analysointia. Lisäksi luvussa arvioidaan tutkimuksen luotettavuutta. Luvussa 6 raportoidaan empiirisen tutkimuksen tulokset teemoittain. Luvussa 7 verrataan saatuja tuloksia aikaisempaan tutkimukseen aiheesta ja luvussa 8 esitetään tutkimuksen yhteenveto, arvioidaan tutkimuksen rajoitteita ja esitetään mahdollisia jatkotutkimusaiheita.

2 KETTERÄN OHJELMISTOKEHITYKSEN MÄÄRITTELY

Ketterä ohjelmistokehitys on ilmiönä hankalasti määriteltävissä, osan määritelmistä perustuessa ketteryyden taustalla olevaan filosofiaan ja osan perustuessa useimmille ketterille menetelmille yhteisiin käytänteisiin (Abbas, Gravell & Wills, 2008). Useat tutkijat ovat havainneet, että ketterän ohjelmistokehityksen tieteellinen tutkimus on epäselvää ja heikosti yhtenäistettävissä (Conboy, 2009; Lee & Xia, 2010; Dingsøyr ym., 2012). Osaltaan tämä voi liittyä ketterän ohjelmistokehityksen alkuperään. Conboyn (2009) mukaan ketterien menetelmien luomisesta ja niiden käytön edistämisestä ovat ilmiön alkuaikoina vastanneet lähinnä niitä työssään käyttäneet asiantuntijat ja konsultit. Ketterän ohjelmistokehityksen taustalla olevat arvot, periaatteet ja käytänteet ovat johdannaisia niitä käyttäneiden ammattilaisten omista kokemuksista (Lee & Xia, 2010), eikä niiden tehokkuutta ole pystytty todistamaan tieteellisesti (Dingsøyr ym., 2012).

Yleisesti ketteryyttä voidaan luonnehtia muun muassa kykynä olla vikkelä, notkea ja valpas (Erickson, Lyytinen & Siau, 2005). Ketteryys ei ole vain ohjelmistokehitykseen yhdistettävä konsepti ja vaikka ketteryyden konsepti ohjelmistokehityksen kontekstissa on verrattain tuore, on ketteryyttä tutkittu muilla tieteenaloilla jo pitkään – liiketoiminnan kirjallisuudessa termi esiintyi jo vuonna 1991. Itse asiassa ketteryyden konsepti syntyi tuotannon ja hallinnon tieteenalalla, jossa sitä on käytetty ja tutkittu kattavasti. (Conboy, 2009.) Seuraavissa alaluvuissa pohditaan ketterän ohjelmistokehityksen määritelmää yksityiskohtaisemmin.

2.1 Ketterän ohjelmistokehityksen julistus

Ketterän ohjelmistokehityksen julistus (engl. Manifesto for Agile Software Development) julkaistiin vuonna 2001 ketteriä menetelmiä käyttävien asiantuntijoiden työpajan tuotoksena (Agile Alliance, 2001). Vaikka julistusta kohtaan on osoitettu kritiikkiä (Laanti, Similä & Abrahamsson, 2013), on sen merkitys ket-

terälle ohjelmistokehitykselle suuri, minkä takia julistuksen käsittely tämän tutkielman näkökulmasta on oleellista. Jatkossa ketterän ohjelmistokehityksen julistuksesta käytetään kirjallisuudessa vakiintunutta termiä Agile Manifesto (Dybå & Dingsøyr, 2008; Conboy, 2009; Dingsøyr ym., 2012).

Agile Manifeston taustalla on pyrkimys tarjota vaihtoehto perinteisille ohjelmistokehityksen menetelmille, jotka ovat raskaita ja korostavat dokumentoinnin tärkeyttä (Agile Alliance, 2001). Näihin perinteisiin ohjelmistokehityksen menetelmiin viitataan kirjallisuudessa usein suunnitelmaperusteisina menetelminä (Dybå & Dingsøyr, 2008; Abrahamsson, Conboy, & Wang, 2009; Lee & Xia, 2010). Agile Manifesto määrittelee neljä arvoa ja 12 periaatetta, jotka yhdistetään parempaan ohjelmistokehitykseen. Agile Manifesto alkaa kirjoittajien toteamuksella: ”Löydämme parempia tapoja tehdä ohjelmistokehitystä, kun teemme sitä itse ja autamme muita siinä.” (Agile Alliance, 2001). Kirjoittajat arvostavat:

> Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja
>
> Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota
>
> Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja
>
> Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa (Agile Alliance, 2001)

Arvoissa on verrattu vasemmalla puolella olevia ketteriä arvoja oikealla puolella oleviin ketteryyden vastakohtiin. Agile Manifesto ei kuitenkaan pyri tekemään selvää rajanvetoa siihen, mitkä asiat pitäisi sisällyttää ketterään ohjelmistokehitykseen ja mitkä tulisi jättää pois. Arvojen yhteydessä kirjoittajat toteavat: ”Jälkimmäisilläkin asioilla on arvoa, mutta arvostamme ensiksi mainittuja enemmän.” (Agile Alliance, 2001). Toteamus on epäselvä, eikä se auta ymmärtämään miten paljon ketteriä menetelmiä hyödyntävien yritysten tulisi keskittyä ketteriin arvoihin verrattuna niiden vastakohtiin. Agile Manifesto ei myöskään ota kantaa ketterien arvojen keskinäisiin suhteisiin, eli esimerkiksi mikä näistä neljästä arvosta on ketteryyden näkökulmasta tärkein.

Arvot on jaettu edelleen kahteentoista periaatteeseen, jotka valottavat ketteryyttä hieman tarkemmin. Agile Manifeston kirjoittajat kertovat noudattavansa tiettyjä periaatteita, joihin Agile Manifesto itsessään pohjautuu (Agile Alliance, 2001). Ne kuvaavat tarkemmin, kuinka yrityksen, joka haluaa tehdä ketterää ohjelmistokehitystä, tulisi toimia. Ketteriä periaatteita tarkastelemalla voidaan myös tutkia ketterien arvojen keskinäisiä suhteita ja tehdä johtopäätöksiä siitä, mitkä Agile Manifeston arvot ovat tärkeimpiä kirjoittajiensa mielestä. Kinnunen (2015) on kuvannut Agile Manifeston ja ketterien menetelmien suhdetta toisiinsa. Taulukossa 1 on kuvattu, miten 12 ketterää periaatetta voidaan yhdistää Agile Manifeston neljään arvoon. Taulukosta huomataan, etteivät periaatteet jakaudu tasaisesti arvojen kesken – puolet periaatteista jakautuu yksilöihin ja kanssakäymiseen, kun muutokseen vastaamiseen viittaa vain yksi periaate.

TAULUKKO 1 Agile Manifeston arvojen ja ketterien periaatteiden vertailu (mukaillen Kinnunen, 2015, s. 18)

Agile Manifeston arvo	**Ketterän ohjelmistokehityksen periaate**
Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja	Liiketoiminnan edustajien ja ohjelmistokehittäjien tulee työskennellä yhdessä päivittäin koko projektin ajan.
	Rakennamme projektit motivoituneiden yksilöiden ympärille. Annamme heille puitteet ja tuen, jonka he tarvitsevat ja luotamme siihen, että he saavat työn tehtyä.
	Tehokkain ja toimivin tapa tiedon välittämiseksi kehitystiimille ja tiimin jäsenten kesken on kasvokkain käytävä keskustelu.
	Ketterät menetelmät kannustavat kestävään toimintatapaan. Hankkeen omistajien, kehittäjien ja ohjelmiston käyttäjien tulisi pystyä ylläpitämään työtahtinsa hamaan tulevaisuuteen.
	Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat syntyvät itseorganisoituvissa tiimeissä.
	Tiimi tarkastelee säännöllisesti, kuinka parantaa tehokkuuttaan, ja mukauttaa toimintaansa sen mukaisesti.
Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota	Toimiva ohjelmisto on edistymisen ensisijainen mittari.
	Teknisen laadun ja ohjelmiston hyvän rakenteen jatkuva huomiointi edesauttaa ketteryyttä.
	Yksinkertaisuus - tekemättä jätettävän työn maksimointi - on oleellista.
Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja	Tärkein tavoitteemme on tyydyttää asiakas toimittamalla tämän tarpeet täyttäviä versioita ohjelmistosta aikaisessa vaiheessa ja säännöllisesti.
	Toimitamme versioita toimivasta ohjelmistosta säännöllisesti, parin viikon tai kuukauden välein, ja suosimme lyhyempää aikaväliä.
Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa	Otamme vastaan muuttuvat vaatimukset myös kehityksen myöhäisessä vaiheessa. Ketterät menetelmät hyödyntävät muutosta asiakkaan kilpailukyvyn edistämiseksi.

Taulukkoa tulkitessa tulee huomioida, että osa periaatteista voi toimia pohjana useammalle arvolle. Esimerkiksi periaate ”Toimitamme versioita toimivasta ohjelmistosta säännöllisesti, parin viikon tai kuukauden välein, ja suosimme lyhyempää aikaväliä” viittaa taulukossa asiakasyhteistyön arvoon, mutta sen voisi yhdistää myös toimivan ohjelmiston arvoon. Vertailusta voidaan kuitenkin päätellä, että Agile Manifeston neljästä arvosta yksilöt ja kanssakäyminen

on kirjoittajien mielestä ketterän ohjelmistokehityksen näkökulmasta tärkein. Ketteriä periaatteita käsitellään lisää luvussa 3.

Julkaisunsa jälkeen Agile Manifesto on herättänyt paljon keskustelua eri tutkijoiden toimesta. Sitä käytetään yleisesti ketterän ohjelmistokehityksen määrittelyyn, mutta samalla sitä pidetään liian laveana tieteellisen tutkimuksen perustana käytettäväksi. (Laanti ym., 2013.) Conboy (2009) viittaa Agile Manifestoon pikemminkin mainosmateriaalina kuin tieteellisenä määrittelynä. Täten onkin selvää, ettei ketterää ohjelmistokehitystä voida uskottavasti ja tarpeeksi kattavasti määritellä vain Agile Manifeston pohjalta. Seuraavassa on esitetty muita ketterän ohjelmistokehityksen määrittelyjä.

2.2 Ketterän ohjelmistokehityksen määrittelyjä

Ericksonin ym. (2005) mukaan ketterän ohjelmistokehityksen ydin on poistaa mahdollisimman paljon suunnitelmaperustaisiin menetelmiin liittyvää raskautta, mikä mahdollistaa nopean reagoinnin muutoksiin erimerkiksi toimintaympäristössä, käyttäjävaatimuksissa ja projektin aikatauluissa. Taustalla on ajatus siitä, että suunnitelmaperustaiset menetelmät eivät mahdollista ympäristön muutoksiin vastaamista, sillä ne ovat liian vakiintuneita ja hitaita suunnanmuutoksiin. Ketteryyden katsotaan myös paremmin mahdollistavan muutoksien ennakoinnin (van Waardenburg & van Vliet, 2013).

Ketterässä ohjelmistokehityksessä ohjelmistoa kehitetään useammassa iteraatiossa, yksittäisen iteraation sisältäessä kaikki vaiheet suunnittelusta julkaisuun. Tällöin kehitettävään ohjelmistoon rakennetaan uusia toiminnallisuuksia inkrementaalisesti, ja palautetta uusista toiminnallisuuksista pyydetään jokaisen iteraation jälkeen. (Abbas ym., 2008.) Avainasemassa on myös iteraatioiden nopeus: kehitystiimit keskittyvät vain tärkeimpiin toiminnallisuuksiin ja julkaisevat ne nopeasti saadakseen niistä palautetta (Abrahamsson, Warsta, Siponen & Ronkainen, 2003).

Artikkelissaan, jossa Dingsøyr ym. (2012) pyrkivät selvittämään ketterän ohjelmistokehityksen tutkimuksen tilaa ja uraauurtavia tutkijoita, mainitsevat kirjoittajat Conboyn (2009) määritelmän ketterästä ohjelmistokehityksestä ”ylivoimaisesti kattavimpana” (Dingsøyr ym., 2012, s. 1214). Määritelmä on kattava, sillä se pohjautuu laajaan kirjallisuuskatsaukseen, jossa Conboy (2009) on tutkinut ketteryyden konseptia monesta eri näkökulmasta. Määritelmä on lopputulos päättelyketjussa, joka lähtee oletuksesta, että ketteryys pohjautuu kahteen konseptiin: joustavuus (engl. flexibility) ja turhan poistaminen (engl. leanness) (Conboy, 2009). Conboy (2009) on tutkinut systemaattisesti molempia konsepteja sekä peilannut niitä ketteryyteen. Päättelyketjun tuloksena Conboy esittää ohjelmistokehityksen ketteryyden tieteellisen määritelmän:

> Tietojärjestelmäkehityksen metodin jatkuva valmius luoda muutosta, proaktiivisesti tai reaktiivisesti vastaanottaa ja hyväksyä muutos ja oppia muutoksesta samalla lisä-

> ten asiakkaan kokemaa arvoa (taloudellinen, laadullinen ja yksinkertaisuus) hyödyntäen sen osia ja suhteita ympäristöön. (Conboy, 2009, s. 340)

Kuten Agile Manifeston sekä Ericksonin ym. (2005) näkemykset ketterästä ohjelmistokehityksestä, myös Conboyn (2009) määritelmä korostaa muutoksen merkitystä. Muutokseen ei tule ainoastaan vastata (Agile Alliance, 2001) tai reagoida (Erickson, 2005), vaan sitä tulee myös luoda ja siitä pitää oppia (Conboy, 2009). Conboyn (2009) määritelmän mukaan ketterä ohjelmistokehitys ei siis ole vain muutoksiin vastaamista ja niistä selviämistä, vaan ketterän ohjelmistokehityksen tulee myös mahdollistaa muutoksen luominen ja toiminnan parantaminen muutoksen tuloksena. Cockburn ja Williams (2003) ovat kuvanneet ohjelmistokehitystä epälineaarisena prosessina, jossa ei oletettavasti voida päätyä aina samaan lopputulokseen noudattamalla ennakkoon määritettyjä vaiheita, koska kehityksen aikana toimintaympäristössä tapahtuu liikaa muutoksia. Tämän takia on tärkeää, että ketterä ohjelmistokehitys antaa mahdollisuuden reagoida tunnistettuihin muutoksiin. Olennaista on myös se, että muutos voidaan nähdä myös mahdollisuutena: kaikki muutokset eivät ole uhkia, vaan ketteryyden ansiosta yritykset voivat paremmin tarttua ohjelmistokehityksen aikana tunnistettuihin uusiin liiketoiminta-mahdollisuuksiin. (Cockburn & Williams, 2003; Lyytinen & Rose, 2006.)

Kattavan määritelmän lisäksi Conboy (2009) on esittänyt myös ketteryyden luokittelun (kuvio 1). Luokittelu perustuu Conboyn esittämään ketteryyden määritelmään ja määritelmän tärkeimpiin komponentteihin. Luokittelu kuvaa ne tavoitteet, jotka tietojärjestelmäkehityksen menetelmän tai sen osan tulee saavuttaa ollakseen ketterä. Ensinnäkin menetelmän osan tulee jollakin tapaa edistää muutosta: se voi auttaa muutoksen luomisessa, sen ennakoinnissa, siihen reagoinnissa tai siitä oppimisessa. Toiseksi menetelmän osan tulee edistää havaittua taloudellisuutta, havaittua laatua tai havaittua yksinkertaisuutta siten, ettei se vähennä niistä yhtäkään. Tällöin 400-sivuinen vaatimusmäärittelyn dokumentti saattaa lisätä laatua ja yksinkertaisuutta, mutta ei silti ehkä ole ketterä, jos se vähentää taloudellisuutta. Kolmanneksi menetelmän osan jatkuva käyttövalmius on sen ketteryyden edellytys. Esimerkiksi testaus lisää ketteryyttä joltakin osin, mutta jos testien valmistelu kestää tunteja, on sen merkitys ketteryydelle epäselvä. (Conboy, 2009.)

Myös Lee ja Xia (2010) pitävät ketterän ohjelmistokehityksen perustana olevana teemana muutoksen hyväksymistä ja siihen vastaamista. Ketteryys saavutetaan lyhyiden, inkrementaalisten ja iteratiivisten aika-ikkunaan sidottujen kehityskierrosten, itseorganisoituvien tiimien, sidosryhmien aktiivisen osallistumisen ja toimivan ohjelmiston jatkuvan toimittamisen avulla (Lee & Xia, 2010). He tutkivat ohjelmistokehityksen ketteryyttä kehitystiimin näkökulmasta ja määrittelevät ketteryyden ”kehitystiimin kykynä tehokkaasti vastata muutoksen ja sisällyttää käyttäjävaatimusten muutokset projektin elinkaaren aikana” (Lee & Xia, 2010, s. 90). Ketteryyttä on kuvattu myös ihmiskeskeisyyden avulla: ohjelmistokehityksen ollessa arvaamatonta, onnistuminen vaatii luovia ja lahjakkaita ihmisiä. Yksi ketterän ohjelmistokehityksen tehtävistä on tukea kehitystiimiä määrittämään paras tapa tehdä työnsä. (Abbas ym., 2008.)

1. Ollakseen ketterä, tietojärjestelmäkehityksen menetelmän osan* tulee edistää yhtä tai useampaa seuraavista:
   i. muutoksen luominen
   ii. muutoksen ennakointi
   iii. muutokseen reagointi
   iv. muutoksesta oppiminen

2. Ollakseen ketterä tietojärjestelmäkehityksen menetelmän osan tulee edistää yhtä tai useampaa seuraavista, vähentämättä niistä ainuttakaan:
   i. havaittu taloudellisuus
   ii. havaittu laatu
   iii. havaittu yksinkertaisuus

3. Ollakseen ketterä tietojärjestelmäkehityksen menetelmän osan tulee olla jatkuvassa valmiudessa. Toisin sanoen minimaalinen aika ja kustannus osan käytön valmistelussa.

* Tietojärjestelmäkehityksen menetelmän mikä tahansa tietty osa

KUVIO 1 Tietojärjestelmäkehityksen ketteryyden luokittelu (Conboy, 2009, s. 341)

2.3 Yhteenveto ketterän ohjelmistokehityksen määrittelystä

Yhteenvetona voidaan todeta, ettei tieteellisessä kirjallisuudessa ole täysin vakiintunutta määritelmää ketterälle ohjelmistokehitykselle. Yhtenä merkittävänä syynä voi olla se, että konsepti on vielä verrattain tuore ohjelmistokehityksen kontekstissa ja sen edistämisestä ovat alkuaikoina vastanneet pääosin ketteriä menetelmiä työkseen käyttäneet asiantuntijat ja konsultit. Tällä on ollut myös vaikutuksensa tieteelliseen tutkimukseen; suuri osa ketterää ohjelmistokehitystä käsittelevästä kirjallisuudesta ei perustu teorialle (Dingsøyr ym., 2012). Agile Manifesto on usean tutkijan mielestä huono lähtökohta tieteellisen työn pohjaksi ja vaikka sen merkityksestä akateemiselle tutkimukselle on esitetty kritiikkiä (Conboy, 2009; Laanti ym., 2013), sitä käytetään silti akateemisessa kirjallisuudessa monen artikkelin lähtökohtana ketteryyden määrittelyyn.

Monissa edellä esitetyissä ketterän ohjelmistokehityksen määrittelyissä otetaan kantaa muutokseen. Täten voidaankin olettaa, että ketterässä ohjelmistokehityksessä oleellista on mahdollisuus muutokseen; joko sen luominen, sen ennakoiminen, siihen reagoiminen tai siitä oppiminen. Muita ketterään ohjelmistokehitykseen yhdistettäviä konsepteja ovat muun muassa keveys, inkrementaalisuus ja iteratiivisuus sekä Agile Manifestossa esiintyvät yksilöt ja kanssakäyminen, toimiva ohjelmisto ja asiakasyhteistyö. Tärkeä huomio on myös Conboyn (2009) määritelmässä eksplisiittisesti esiintyvä liiketoiminnallinen näkökulma – ketterä ohjelmistokehitys on kuitenkin pohjimmiltaan keino yrityksille hankkia kilpailuetua, tehdä ohjelmistokehitystä taloudellisemmin ja tarjota

asiakkailleen liiketoiminnallista arvoa. Ennen kaikkea ketteryys on tärkeää siksi, että se edistää näitä tekijöitä.

3 KETTERÄN OHJELMISTOKEHITYKSEN PERIAATTEET JA MENETELMÄT

Ketteryyttä saavuttaakseen yritys voi toimia ketterien periaatteiden mukaisesti tai hyödyntää ketteriä menetelmiä tai käytänteitä. Tässä yhteydessä ketterät periaatteet ovat Agile Manifestossa esitetyt 12 periaatetta (Agile Alliance, 2001). Ketterät menetelmät, joita ovat muun muassa Extreme Programming (XP), Scrum, Lean-ohjelmistokehitys (engl. Lean Software Development), Feature-Driven Development (FDD), Crystal, Dynamic Systems Development Method (DSDM), Agile Modeling ja näiden muunnokset (Dybå & Dingsøyr, 2008; Conboy, 2009; Lee & Xia, 2010; Dingsøyr ym., 2012), perustuvat pääsääntöisesti ketteriin periaatteisiin. Jos yritys ei käytä yhtä menetelmää tai niiden yhdistelmää, se voi kuitenkin hyödyntää valitsemiaan ketteriä käytänteitä toiminnassaan. Seuraavissa alaluvuissa kuvataan ketterät periaatteet, yleisimmin käytetyt ketterät menetelmät ja käytänteet.

Ketteriä menetelmiä kuvatessa tulee kiinnittää huomiota siihen, mitkä ketterät menetelmät kannattaa valita lähempään tarkasteluun, sillä pro gradu -tutkielman rajallisen laajuuden vuoksi kaikkia menetelmiä ei voida tarkastella. Yleisesti käytössä olevia ketteriä menetelmiä on kartoittanut yhdysvaltalainen teknologiayhtiö VersionOne (2015). Tutkimuksen mukaan ylivoimaisesti yleisin ketterä menetelmä on Scrum; 56 % vastaajista raportoi käyttävänsä sitä. Seuraavaksi yleisimpiä varsinaisia menetelmiä olivat Kanban (5 %) ja Lean-ohjelmistokehitys (2 %). Näitä suositumpia olivat kuitenkin eri menetelmien hybridit: Scrum/XP (10 %), yrityksen tarpeisiin mukautettu hybridi (8 %) ja Scrumban (6 %). (VersionOne, 2015.)

Suomessa ketterien menetelmien käyttöä ovat tutkineet Rodríguez, Markkula, Oivo, ja Turula (2012). Heidän vuonna 2011 järjestämäänsä kyselyyn vastasi 408 Suomessa eri rooleissa toimivaa asiantuntijaa. Tutkimuksen tuloksien mukaan viisi yleisintä ketterää menetelmää suomalaisissa ohjelmistoyrityksissä olivat Scrum, Extreme Programming, Agile Modeling, Feature-Driven Development ja Kanban; Scrumin ollessa tässäkin tapauksessa ylivoimaisesti käytetyin menetelmä. (Rodríguez ym., 2012.) Tutkimuksen tuloksia tulkitessa tulee ottaa huomioon, että tutkimuksen tuloksista on tämän pro gradu

-tutkielman kirjoittamisen aikaan kulunut jo viisi vuotta. Tämä on pitkä aika, kun otetaan huomioon, miten tuore ilmiö ketterä ohjelmistokehitys kokonaisuudessaan on. On siis erittäin todennäköistä, että ketterien menetelmien käytössä suomalaisten ohjelmistoyrityksien kontekstissa on tapahtunut muutoksia tutkimuksen toteutuksen jälkeen.

Näiden tietojen pohjalta tarkemmin tutkittaviksi ketteriksi menetelmiksi valittiin Scrum, Extreme Programming, Kanban ja Lean-ohjelmistokehitys. Scrum ja Extreme Programming valittiin niiden suhteellisen suosion vuoksi: Scrum on ylivoimaisesti suosituin käytössä olevista ketteristä menetelmistä, kun taas Extreme Programming oli toiseksi suosituin Rodríguezin ym. (2012) tutkimuksessa ja Scrum/XP -hybridi oli toiseksi suosituin VersionOnen (2015) tutkimuksessa. Kanban ja Lean-ohjelmistokehitys valittiin, sillä ne olivat seuraavaksi käytetyimmiksi raportoidut menetelmät VersionOnen (2015) tutkimuksessa.

3.1 Ketterät periaatteet

Kuten luvussa 2 todettiin Agile Manifeston kirjoittajat kertovat noudattavansa tiettyjä periaatteita, joihin Agile Manifesto itsessään pohjautuu (Agile Alliance, 2001). Nämä ketterät periaatteet eivät ole virallinen määritelmä ketteryydelle vaan pikemminkin ne tarjoavat suuntaviivoja siihen, miten laadukasta ohjelmistoa voidaan tuottaa ketterällä tavalla (Dingsøyr ym., 2012). Luvussa 2 tarkasteltiin periaatteiden merkitystä Agile Manifeston arvoille näiden priorisoinnin näkökulmasta. Koska periaatteet ovat vaikuttaneet lukuisten ketterien menetelmien ja käytänteiden kehittymiseen, on niiden tarkempi käsittely tämän tutkielman kontekstissa perusteltua.

Tiivistettynä periaatteiden taustalla on ajatus itseorganisoituvista tiimeistä, jotka työskentelevät yhdessä pyrkien säilyttämään sellaisen tahdin, joka mahdollistaa luovuuden ja tuottavuuden. Periaatteet korostavat muutoksen tärkeyttä – muutokset käyttäjävaatimuksiin tulisi voida hyväksyä missä tahansa kehitysvaiheessa. Lisäksi asiakkaat (tai heidän edustajansa) ovat aktiivisesti osana kehitysprosessia antaen palautetta ja esittävät ajatuksiaan, mikä mahdollistaa paremman lopputuloksen. (Dingsøyr ym., 2012.)

Ketteriä periaatteita ovat tutkineet Laanti ym. (2013). He ovat analysoineet periaatteet sen mukaan, mitä ketteryyteen yhdistettävää konseptia eri periaatteet painottavat (taulukko 2). Taulukosta huomataan, että verrattuna Agile Manifeston arvoihin, ketterät periaatteet antavat tarkempia ohjeita siitä, miten yritys, joka haluaa tehdä ketterää ohjelmistokehitystä, voi toimia ketterästi – esimerkiksi toteamus: ”Liiketoiminnan edustajien ja ohjelmistokehittäjien tulee työskennellä yhdessä päivittäin koko projektin ajan” on tarkempi kuin Agile Manifeston arvo ”Kokemuksemme perusteella arvostamme: Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja”. Periaatteet jättävät käytännön implementoinnin kuitenkin ketteryyttä tavoittelevan yrityksen vastuulle:

ne eivät ota kantaa esimerkiksi siihen, millä tavoin liiketoiminnan edustajien ja ohjelmistokehittäjien kannattaisi käytännössä työskennellä yhdessä päivittäin.

TAULUKKO 2 Ketterät periaatteet ja mitä ne painottavat (Laanti ym., 2013, s. 249)

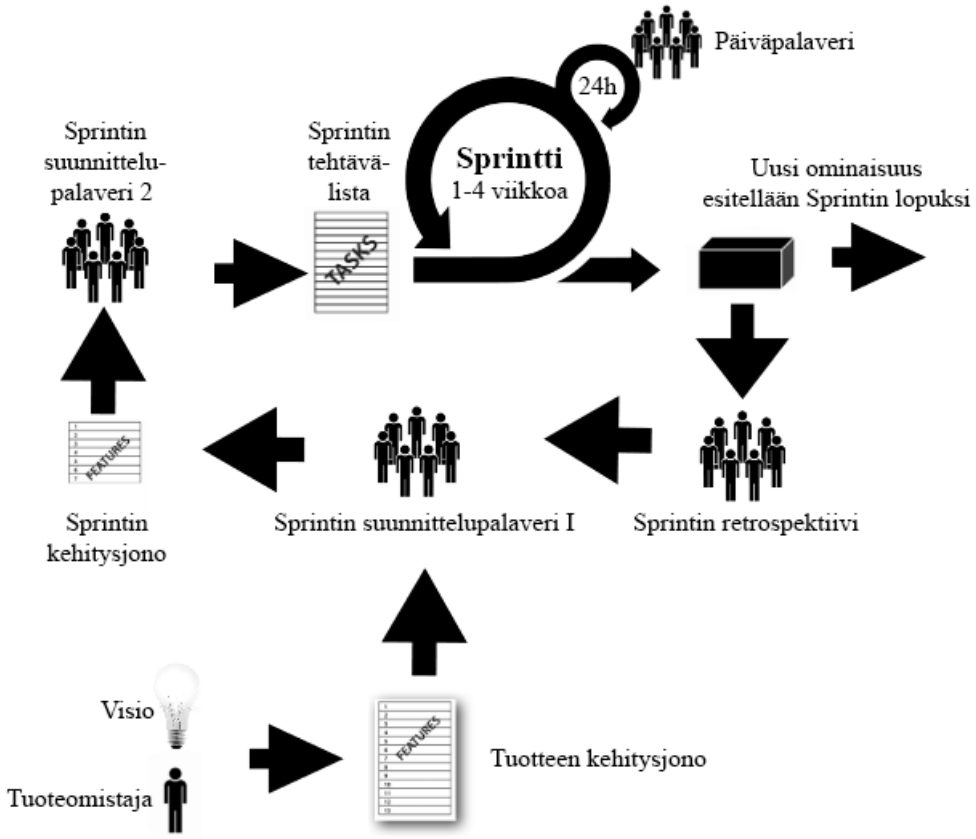
Ketterä periaate	**Painotus**
Tärkein tavoitteemme on tyydyttää asiakas toimittamalla tämän tarpeet täyttäviä versioita ohjelmistosta aikaisessa vaiheessa ja säännöllisesti.	Asiakastyytyväisyys, Jatkuva toimittaminen, Aikaiset toimitukset
Otamme vastaan muuttuvat vaatimukset myös kehityksen myöhäisessä vaiheessa. Ketterät menetelmät hyödyntävät muutosta asiakkaan kilpailukyvyn edistämiseksi.	Mukautumiskyky, Kilpailukyky, Asiakashyöty
Toimitamme versioita toimivasta ohjelmistosta säännöllisesti, parin viikon tai kuukauden välein, ja suosimme lyhyempää aikaväliä.	Usein tapahtuvat toimitukset
Liiketoiminnan edustajien ja ohjelmistokehittäjien tulee työskennellä yhdessä päivittäin koko projektin ajan.	Yhteistyö
Rakennamme projektit motivoituneiden yksilöiden ympärille. Annamme heille puitteet ja tuen, jonka he tarvitsevat ja luotamme siihen, että he saavat työn tehtyä.	Motivoituneet yksilöt, Hyvä ympäristö, Tuki, Luottamus
Tehokkain ja toimivin tapa tiedon välittämiseksi kehitystiimille ja tiimin jäsenten kesken on kasvokkain käytävä keskustelu.	Tehokkuus, Kommunikointi
Toimiva ohjelmisto on edistymisen ensisijainen mittari.	Edistymisen mittaaminen toimitettavan tuotteen perusteella
Ketterät menetelmät kannustavat kestävään toimintatapaan. Hankkeen omistajien, kehittäjien ja ohjelmiston käyttäjien tulisi pystyä ylläpitämään työtahtinsa hamaan tulevaisuuteen.	Pysyvyys, Ihmiset
Teknisen laadun ja ohjelmiston hyvän rakenteen jatkuva huomiointi edesauttaa ketteryyttä.	Keskittyminen tekniseen erinomaisuuteen
Yksinkertaisuus - tekemättä jätettävän työn maksimointi - on oleellista.	Yksinkertaisuus, Työn optimointi
Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat syntyvät itseorganisoituvissa tiimeissä.	Itseorganisoituvuus
Tiimi tarkastelee säännöllisesti, kuinka parantaa tehokkuuttaan, ja mukauttaa toimintaansa sen mukaisesti.	Sisäänrakennettu tehokkuuden ja toiminnan parantaminen

Ketterien periaatteiden analyysi, jonka Laanti ym. (2013) ovat esittäneet, laajentaa Agile Manifeston määritelmää ketteryydestä. Kun tarkastellaan eri periaatteiden painotuksia, huomataan, että periaatteisiin voidaan yhdistää 22 yksilöllistä konseptia, joilla periaatteita voidaan kuvata – vaikkakin osa konsepteista eroaa vain vähän toisistaan. Esimerkiksi Jatkuva toimittaminen, Aikaiset toimi-

tukset ja Usein tapahtuvat toimitukset kuvaavat kaikki samaa asiaa: ohjelmistoa tulisi toimittaa asiakkaalle usein ja mahdollisimman nopeasti kehitysprojektin alusta alkaen.

3.2 Scrum

Kuten edellä todettiin, Scrum on ketteristä menetelmistä käytetyin. Menetelmän käyttö ja suosio on niin laajaa, että joissakin tapauksissa sen jopa ajatellaan olevan yhtä kuin ketteryys (Hossain, Babar & Paik, 2009; Poppendieck & Cusumano, 2012). Se ei kuitenkaan ole puhtaasti ohjelmistokehitysmenetelmä, vaikkakin menetelmän suosiota selittää suurilta osin sen menestyksekäs käyttö juuri ohjelmistokehityksen kontekstissa. Itse asiassa, Scrumin luojien, Ken Schwaberin ja Jeff Sutherlandin (2013), mukaan Scrum ei ole prosessi tai tekniikka tuotteiden rakentamiseen – sen sijaan sen on enemmänkin viitekehys, jossa voidaan hyödyntää lukuisia prosesseja ja tekniikoita. Scrum muodostuu rooleista, tapahtumista ja tuotoksista (kuvio 2). (Schwaber & Sutherland, 2013.) Seuraavaksi kuvataan nämä tarkemmin.



KUVIO 2 Scrumin roolit, tapahtumat ja tuotokset (Sutherland, 2010, s. 11)

3.2.1 Scrum roolit

Scrum sisältää kolme roolia, jotka kaikki ovat osa Scrum-tiimiä: tuoteomistaja, kehitystiimi ja Scrummaster. Scrum-tiimin autonomisuus on yksi Scrumia määrittelevistä tekijöistä, jolloin päätöksenteko tapahtuu operatiivisella tasolla, eikä perinteisen ”ylhäältä alas” tulevan käskyn kautta (Moe, Dingsøyr & Dybå, 2010). Tuoteomistaja vastaa kehitettävän tuotteen ja kehitystiimin työn arvon maksimoinnista. Samalla tuoteomistaja myös vastaa tuotteen kehitysjonon hallinnasta. (Schwaber & Sutherland, 2013.) Tuoteomistajan rooli on erittäin oleellinen kehitysprojektin onnistumisen kannalta: hän on vastuussa siitä, että oikeat asiat tulevat tehdyksi, oikeaan aikaan. Schwaber ja Sutherland (2013) ohjeistavat, että koko organisaation tulee kunnioittaa tuoteomistajan päätöksiä, jotta hän voi onnistua työssään. He myös teroittavat, että tuoteomistajan tulee olla yksi henkilö, ei komitea.

Kehitystiimin jäsenet vastaavat tuoteversion kehityksestä. Kehitystiimillä tulee olla valta organisoida omaa työtään autonomisesti, jotta tästä muodostuvat synergiaedut voisivat parantaa kehitystiimin suorituskykyä ja tuottavuutta. Kehitystiimiä kuvataan muun muassa seuraavasti: kehitystiimit ovat itseohjautuvia ja monitaitoisia, kaikkien jäsenten titteli on kehittäjä, vastuualueesta riippumatta, kehitystiimi on yhteisesti vastuussa kehittämisestä ja kehitystiimien ei tule sisältää alitiimejä. (Schwaber & Sutherland, 2013.)

Scrummaster vastaa, että Scrum-tiimin jäsenet ymmärtävät ja käyttävät Scrumia. Scrummasterin tehtävänä on varmistaa, että kaikki pitäytyvät Scrumin teoriassa, käytännöissä ja säännöissä. Kaikki Scrummasterin tehtävät eivät liity Scrum-tiimin sisäiseen toimintaan – hän auttaa myös ulkopuolisia ymmärtämään, mitkä heidän toimintatavoistaan edistävät Scrum-tiimin toimintaa ja mitkä eivät. (Schwaber & Sutherland, 2013.)

3.2.2 Scrum tapahtumat

Scrumin tapahtumat ovat ennalta sovittuja ja aikarajattuja. Ne on suunniteltu lisäämään läpinäkyvyyttä ja mahdollistamaan tarkastelu projektin aikana. (Schwaber & Sutherland, 2013.)

Sprintti on Scrumin ydin. Se on rajattu enintään kuukauden pituiseksi ja sen aikana Scrum-tiimi tuottaa käyttökelpoisen ja potentiaalisesti julkaisukelpoisen tuoteversion kehitettävästä tuotteesta. Uusi sprintti seuraa edellistä ja yksi sprintti koostuu suunnittelupalaverista, päiväpalavereista, kehitystyöstä, sprintin katselmoinnista ja sprintin retrospektiivistä. (Schwaber & Sutherland, 2013.)

Sprintin suunnittelupalaverissa suunnitellaan seuraavan sprintin aikana tehtävä työ. Koko Scrum-tiimi osallistuu suunnitteluun ja pyrkii vastaamaan kahteen kysymykseen: ”Mitä seuraavassa sprintissä tehdään?” ja ”Miten valittu työ toteutetaan?”. (Schwaber & Sutherland, 2013.)

Päiväpalaveri pyrkii tekemään kehitystiimin työtä läpinäkyväksi. Siinä kehitystiimi kokoontuu korkeintaan 15 minuutin ajaksi ja tarkastelee edellisen

päiväpalaverin jälkeen tehtyä työtä, ja ennustaa mitä toteutetaan seuraavaan päiväpalaveriin mennessä. Palaverissa jokainen kehitystiimin jäsen vastaa vuorollaan seuraaviin kysymyksiin:

- Mitä tein eilen auttaakseni kehitystiimiä saavuttamaan sprintin tavoitteen?
- Mitä aion tehdä tänään auttaakseni kehitystiimiä saavuttamaan sprintin tavoitteen?
- Onko mitään estettä, joka estää minua tai kehitystiimiä saavuttamasta sprintin tavoitetta? (Schwaber & Sutherland, 2013, s. 10)

Sprintin katselmointi pidetään sprintin lopussa. Tällöin tarkastellaan kehitettyä tuoteversioita ja tehdään tarvittaessa muutoksia tuotteen kehitysjonoon. Scrum-tiimi esittelee tehdyn työn sidosryhmille, jotka antavat palautetta tuoteversiosta ja antavat lisätietoa liiketoiminnasta, markkinoista sekä tarvittavasta teknologiasta. (Hossain ym., 2009.) Tämän tiedon pohjalta osallistujat keskustelevat siitä, mitä seuraavaksi tulisi kehittää arvon optimoimisen näkökulmasta. (Schwaber & Sutherland, 2013.)

Sprintin retrospektiivi on varattu Scrum-tiimin sisäisen toiminnan tarkasteluun. Se pidetään edellisen sprintin katselmoinnin ja seuraavan sprintin suunnittelupalaverin välissä ja sen tarkoituksena on tarkastella muun muassa edellisen sprintin kehitysprosessia, yhteistyötä, työkaluja sekä hyvin ja huonosti sujuneita asioita. Retrospektiivissä tulisi myös määritellä tärkeimmät parannukset sekä luoda suunnitelma prosessin parantamiseen seuraavaa sprinttiä varten. (Hossain ym., 2009; Schwaber & Sutherland, 2013.)

3.2.3 Scrum tuotokset

Scrum sisältää kolme oleellista tuotosta: tuotteen kehitysjonon, sprintin kehitysjonon ja tuoteversion. Tuotteen kehitysjono kuvaa kehitettävän tuotteen sen hetkisiä vaatimuksia. Se siis sisältää kaikki ominaisuudet, toiminnot, vaatimukset, parannukset ja korjaukset, jotka sillä hetkellä tiedetään tarpeelliseksi. Tärkeä huomio on, ettei tuotteen kehitysjono ole valmis, vaan se kehittyy jatkuvasti tuotteen kehittyessä sekä palautteen ja ympäristön muutosten perusteella. (Schwaber & Sutherland, 2013.)

Sprinttiin valittavat tuotteen kehitysjonon kohdat muodostavat sprintin kehitysjonon. Vaatimuksien lisäksi sprintin kehitysjono sisältää myös suunnitelman niiden toteuttamiseksi. Sprintin kehitysjonossa tehtävien tulee olla yksityiskohtaisempia kuin tuotteen kehitysjonossa, jotta tarvittava työmäärä voidaan arvioida riittävällä tarkkuudella. Myös sprintin kehitysjonoon voidaan kuitenkin tehdä muutoksia, kun ymmärrys siitä, mitä sprintin tavoitteen saavuttamiseksi tarvitaan, kasvaa. (Schwaber & Sutherland, 2013.)

Tuoteversio on summa kaikista sen hetkisen tuotteen kehitysversion kohdista, jotka on saatu valmiiksi. Tuoteversion tulee olla käyttökelpoisessa kunnossa ja sen tulee täyttää Scrum-tiimin ”valmiin määritelmä”. Tällä tarkoitetaan

koko tiimin yhteistä ymmärrystä siitä, mitä tarkoittaa, kun tuoteversion sanotaan olevan valmis. (Schwaber & Sutherland, 2013.)

3.3 Extreme Programming

Extreme Programming -menetelmän (XP) teki kuuluisaksi Kent Beckin (1999a) teos Extreme Programming Explained: Embrace Change (Cohen ym., 2004). Ajatus XP:n taustalla on, että ohjelmistokehittäjät keskittyvät vain tiedostettujen vaatimuksien ja ominaisuuksien kehittämiseen. Kaikkia käyttäjien vaatimuksia ei ole tarpeen selvittää etukäteen, vaan muutokset otetaan vastaan myös kehityksen aikana. (Erickson ym., 2005.) Näin XP:tä hyödyntämällä pyritään toimittamaan paras mahdollinen lisäarvo asiakkaalle, nopeimmalla mahdollisella tavalla. Menetelmän nimi juontaa juurensa ajatukseen viedä ohjelmistokehityksen käytänteet ”äärimmilleen” (Agarwal & Umphress, 2008, s. 82). Beckin (1999a) mukaan XP sisältää neljä arvoa sekä neljä aktiviteettia: kommunikaatio, yksinkertaisuus, palaute ja rohkeus sekä ohjelmointi, testaus, kuuntelu ja virheiden poistaminen. Nämä johtavat XP:n 12 käytänteeseen, jotka kuvataan seuraavaksi.

Suunnittelupeli tapahtuu jokaisen kehitysiteraation alussa. Siinä ohjelmistokehittäjät ja sidosryhmät neuvottelevat seuraavaksi toteutettavista vaatimuksista. Vaatimuksia kutsutaan käyttäjätarinoiksi ja ne kirjataan ylös siten, että kaikki ymmärtävät niiden merkityksen. (Cohen ym., 2004.) Pyrkimyksenä on löytää ne vaatimukset, jotka tuottavat eniten liiketoiminnallista arvoa (Agarwal & Umphress, 2008).

Pienillä julkaisuilla tarkoitetaan, että ohjelmistoa kehitetään pienissä osissa, joita voidaan lisätä jo valmistuneeseen ohjelmistoon. Uusia julkaisuja tehdään muutaman päivän tai muutaman viikon välein. (Cohen ym., 2004.)

Ohjelmistokehittäjät ja sidosryhmät sopivat yhteisestä *vertauskuvasta* tai joukosta vertauskuvia, jota käytetään kuvaaman ohjelmistoa. (Cohen ym., 2004.) Tällä pyritään helpottamaan kommunikaatiota kehitystyön aikana (Agarwal & Umphress, 2008).

Yksinkertainen suunnittelu tarkoittaa, että kehittäjien tulee pyrkiä pitämään ohjelmiston suunnittelu mahdollisimman yksinkertaisena (Cohen ym., 2004). Beckin mukaan tämän voi tiivistää toteamalla ”Kerro kaikki kerran ja vain kerran” (Beck, 1999b, s. 71).

Ohjelmistokehittäjien tulisi tehdä työtään *testausvetoisesti*: ennen kuin ohjelmistokoodia kirjoitetaan, sille pitää kirjoittaa testi. Myös asiakkaat kirjoittavat toiminnallisia testejä ja jokaisen iteraation jälkeen, ohjelmiston tulee läpäistä kaikki testit. (Cohen ym., 2004.)

Refaktoroinnilla varmistetaan ohjelmistokoodin yksinkertaisuus. Koodia parannetaan jatkuvasti, jotta se läpäisee määritellyt testit. (Beck, 1999b; Cohen ym., 2004.)

Pariohjelmoinnissa kaksi ohjelmistokehittäjää jakavat yhden tietokoneen ja käyttävät sitä ohjelmistokoodin kirjoittamiseen (Cohen ym., 2004). Näin ohjel-

mistokoodia arvioidaan jatkuvasti, mikä parantaa koodin laatua (Agarwal & Umphress, 2008).

Jatkuva integraatio tarkoittaa, että valmistuvaa ohjelmistokoodia integroidaan nopeasti olemassa olevaan ohjelmistoon. Tällöin uuden ohjelmistoversion tulee läpäistä kaikki testit, myös edelliset, tai muutokset mitätöidään. (Cohen ym., 2004.)

Yhteisessä omistajuudessa kaikki tuotettu ohjelmistokoodi kuuluu kaikille kehitystiimin jäsenelle ja jokainen voi tehdä muutoksia mihin vain ohjelmiston osaan halutessaan. Tämä kannustaa kaikkia kehitystiimin jäseniä tekemään parannuksia kaikkiin ohjelmiston osiin. (Cohen ym., 2004; Agarwal & Umphress, 2008.)

Läsnä oleva asiakas työskentelee kokoaikaisesti kehitystiimin kanssa vastatakseen kysymyksiin, suorittaakseen toiminnallisia testejä ja varmistaakseen, että kehitystyö sujuu asiakkaan tavoitteiden mukaisesti. Paremman kommunikaation ansiosta dokumentaatiota ei tarvitse tehdä yhtä kattavasti. (Cohen ym., 2004; Agarwal & Umphress, 2008.)

40-tuntisella työviikolla varmistetaan, ettei kehitystiimin jäsenten tarvitse työskennellä ylipitkiä työviikkoja. Vaatimukset jokaiseen iteraatioon tulisi valita niin, että ylityöt voidaan välttää. (Cohen ym., 2004.) Jatkuvat ylityöt ovat merkki syvemmistä ongelmista, joihin tulee puuttua (Beck, 1999b).

Kehitystiimi työskentelee *jaetussa työtilassa*, jossa työpisteet ovat toistensa läheisyydessä ja pariohjelmointiin käytettävät yhteiset työpisteet ovat tilan keskellä. (Beck, 1999b; Cohen ym., 2004.)

3.4 Lean-ohjelmistokehitys

Termiä ”lean” käytettiin ensimmäisen kerran tuotannonjohtamisessa ja sen teki kuuluisaksi erityisesti japanilainen autovalmistaja Toyota (Poppendieck & Cusumano, 2012). Ohjelmistokehityksen kontekstiin lean-ajattelua ovat soveltaneet Poppendieck ja Poppendieck (2003) kirjassaan Lean Software Development: An Agile Toolkit. Petersenin ja Wohlinin (2011) mukaan Lean-ohjelmistokehitykselle on ominaista ajatusmalli, jossa keskitytään kokonaisvaltaisesti niihin ohjelmistokehityksen osiin, jotka luovat asiakasarvoa. Tämän tulee ohjata kehitystyötä ”alusta loppuun” eli ensimmäisistä ideoista ja luonnoksista aina ohjelmiston julkaisuun saakka. Työ tulisi pystyä jakamaan kolmeen kategoriaan: arvoa tuottavaan työhön, pakolliseen työhön joka ei lisää arvoa sekä arvoa tuottamattomaan työhön (Wang, Conboy & Cawley, 2012). Lean perustuu seitsemään periaatteeseen, joiden käytäntöön siirtämiseen Poppendieck ja Poppendieck (2003) tarjoavat kirjassaan yhteensä 22 työkalua. Seuraavaksi esitellään Lean-ohjelmistokehityksen 7 periaatetta.

Optimoi kokonaisuus tarkoittaa, että kehitettävä ohjelmisto tulisi nähdä osana suurempaa kokonaisuutta. Asiakkaan näkökulmasta ohjelmiston arvo liittyy usein laajempaan kokonaisuuteen kuten mahdollisuuteen ostaa tuote verkkosi-

vuilta. Sen sijaan, että keskityttäisiin vain ohjelmistokoodiin, tulisi asiakkaan näkökulmaa ajatella laajemmin. (Poppendieck & Cusumano, 2012.)

Eliminoi hukka on turhan työn välttämistä. Hukkaa on kaikki, mikä ei tuota asiakkaalle arvoa tai mikä ei lisää ymmärrystä siitä, kuinka arvoa voitaisiin tuottaa tehokkaammin. Ohjelmistokehityksessä suurimpia hukan aiheuttajia ovat turhat ominaisuudet, menetetty tietämys, osittain tehty työ, tehtävien siirtely sekä virheiden etsiminen ja korjaaminen. (Poppendieck & Cusumano, 2012.)

Sisäisen laadun kehittäminen on tärkeää ohjelmistokehityksen kontekstissa, sillä ohjelmiston odotetaan säilyttävän käytettävyytensä ajan saatossa. Laadukas ohjelmisto sisältää yhdenmukaisen arkkitehtuurin, on käyttäjiensä mielestä käytettävyydeltään erinomainen, sopii tarkoitukseensa ja on ylläpidettävä, mukautuva sekä laajennettava. (Poppendieck, & Poppendieck, 2003.)

Jatkuvalla oppimisella viitataan ohjelmistokehityksen aikana tapahtuvaan oppimiseen. Ohjelmistokehittäjien tulisi kartoittaa eri vaihtoehtoja ja päättää niiden välillä mahdollisimman myöhään – kun he ovat oppineet mahdollisimman paljon eri vaihtoehtojen hyvistä ja huonoista puolista. Oppimista tulisi tapahtua myös tuotteen ominaisuuksien suhteen: kehittämisen voi aloittaa vähimmäisvaatimuksilla, joita sitten laajennetaan asiakaspalautteista oppimalla. (Poppendieck & Cusumano, 2012.)

Toimita nopeasti tarkoittaa, että ohjelmistoa tulisi toimittaa viikoittain, päivittäin tai jopa reaaliajassa. Tällöin yksittäisen ohjelmiston kehitystä ei tulisi ajatella projektina, vaan järjestelmävirtauksena, jossa ohjelmistoa suunnitellaan, kehitetään ja julkaistaan jatkuvana virtauksena pienin muutoksin. (Poppendieck & Cusumano, 2012.)

Sitouta kaikki on edellytys sille, että ohjelmistoa voidaan kehittää virtauksena projektin sijaan. Tällöin ohjelmistokehitystä ei voida nähdä muusta liiketoiminnasta erillisenä yksikkönä, vaan sen tulee sisältää myös muita liiketoimintoja. Itse asiassa, se saattaa muistuttaa ”organisaation pienoismallia” (Poppendieck & Cusumano, 2012, s. 29). Ohjelmistokehityksessä tulisi olla mukana asiakkaan edustajia, suunnittelijoita, kehittäjiä, testaajia, eri toimintojen ja tukipalveluiden edustajia sekä ehkä myös talousvastaavia. (Poppendieck & Cusumano, 2012.)

Jatkuva parantaminen tarkoittaa, että käytänteitä tulisi pyrkiä parantamaan koko ajan. Kehitettävät ohjelmistot ovat usein erittäin monimutkaisia ja kehitys tapahtuu muuttuvassa ympäristössä – jos jokin on toiminut jossain tilanteessa, se ei välttämättä ole paras ratkaisu kaikkiin ongelmiin. Menetelmät ja käytänteet tulisi nähdä lähtökohtana, josta niitä käyttävät ihmiset voivat kehittää niitä paremmiksi. (Poppendieck, & Poppendieck, 2003; Poppendieck & Cusumano, 2012.)

3.5 Kanban

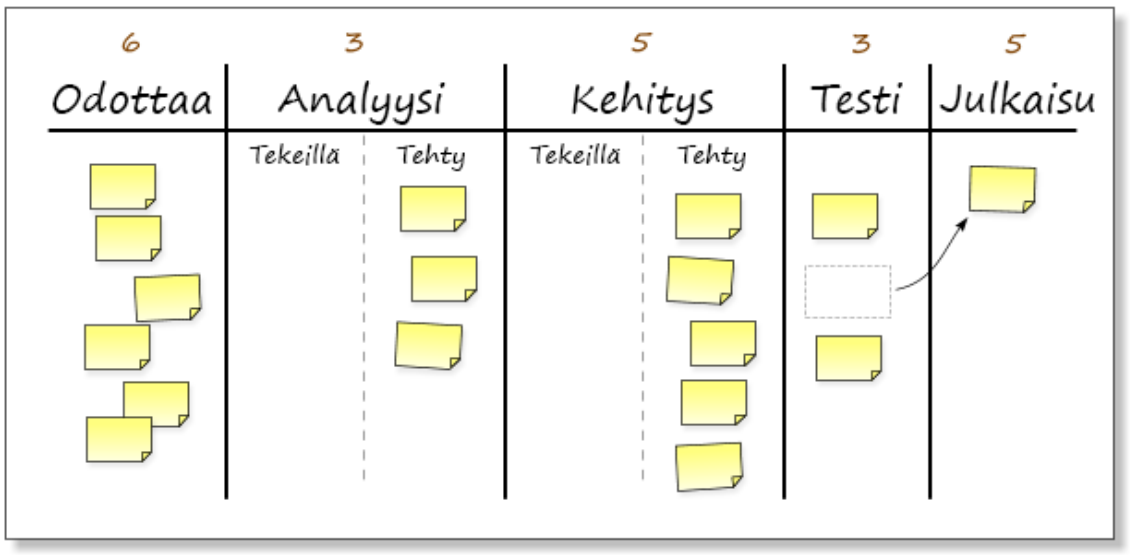
Kanban on menetelmä, joka toteuttaa Lean-ajattelua käytännössä (Ikonen, Pirinen, Fagerholm, Kettunen & Abrahamsson, 2011) ja sen käytöstä ohjelmistoke-

hityksen kontekstissa on viime vuosina tullut yhä suositumpaa (Ahmad, Markkula & Oivo, 2013). Kanban perustuu kolmeen keskeiseen periaatteeseen: visualisoi työvirta, rajoita samanaikainen työmäärä ja mittaa työn läpimenoaikaa (Kniberg & Skarin, 2010). Työvirran visualisoinnin tarkoituksena on maksimoida arvovirtaus ja tehdä näkyväksi mahdolliset pullonkaulat sekä ongelmat. Tärkeimpänä periaatteena on optimoida ohjelmistokehitys kokonaisuutena ja välttää yksittäisen vaiheen yli-optimointia. (Anderson, Concas, Lunesu & Marchesi, 2011.)

Kanbanissa arvovirtaus tehdään näkyväksi taululle (kuvio 3), jossa on oma sarakkeensa jokaiselle arvovirtauksen vaiheelle. Kehitettävän ohjelmiston ominaisuudet ja niihin liittyvät tehtävät priorisoidaan Scrumin tapaan tuotteen kehitysjonoon. Toteutettavat tehtävät asetetaan taululle ja tehtävän siirtyessä seuraavaan vaiheeseen, se siirretään myös taululla. Tämä edellyttää, että tehty työ täyttää sovitut vaatimukset valmiin määrittelystä. Visualisoinnin ansiosta työn alla olevat tehtävät (engl. work in process, WIP) on koko ajan näkyvillä sekä ohjelmistokehitystiimille että muille sidosryhmille. (Anderson ym., 2011; Poppendieck & Cusumano, 2012.) Kanbanissa ohjelmistokehitystiimi ei useimmiten työskentele aikarajattuja tapahtumia noudattaen, vaan työn kulku (engl. flow) määräytyy tehtävien liikkuessa vaiheesta toiseen (Anderson ym., 2011).

Samanaikainen työmäärä on rajoitettu sekä kokonaisuutena että jokaisen arvoa lisäävän vaiheen osalta (Poppendieck & Cusumano, 2012). Jokaisessa vaiheessa voi siis olla yhtäaikaisesti vain rajattu määrä tehtäviä. Tämä tehdään näkyväksi merkitsemällä raja-arvot taulun sarakkeisiin (Kniberg & Skarin, 2010). Näin ohjelmistokehitystiimillä on yhtäaikaisesti vain vähän ominaisuuksia toteutettavanaan ja uusia toiminnallisuuksia pyritään julkaisemaan jatkuvasti arvovirtausta ja työn kulkua optimoiden (Anderson ym., 2011).

Työn läpimenoaika kuvaa sitä, kuinka kauan tehtävällä kestää kulkea Kanban-taulun poikki kaikkien vaiheiden läpi. Optimoimalla prosessia voidaan välttää pullonkaulat kehityksen aikana ja pienentää läpimenoaikaa. (Kniberg & Skarin, 2010.)



KUVIO 3 Esimerkki Kanban-taulusta (Peterson, 2015)

3.6 Ketterät käytänteet

Yritys voi tavoitella ketteryyttä myös implementoimalla ketteriä käytänteitä toimintaansa, ilman, että se noudattaisi tarkasti jotakin ketterää menetelmää. Yang, Liang ja Avgeriou (2016) ovat tehneet yhteenvedon 20 suosituimmasta käytänteestä (taulukko 3). Kirjoittajien mukaan lista on ensimmäinen laatuaan, sillä tieteellisestä kirjallisuudesta ei löydy vastaavaa yhteenvetoa olemassa olevista ketteristä käytänteistä. Yhteensä he löysivät yli sata ketterää käytännettä, joiden jaottelu on tehty perustuen siihen, miten usein eri käytänteisiin on viitattu tutkituissa artikkeleissa. (Yang ym., 2016.)

Lista sisältää paljon edellä esiteltyjen XP:n ja Scrumin käytänteitä: XP:n 12 käytänteestä vain 40-tuntinen työviikko ei esiinny listassa ja Scrumiin viittaavia käytänteitä listassa on viisi. On tärkeää huomioida, että taulukossa mainitut käytänteet on luokiteltu suosituimmiksi kirjallisuudessa esiintyvien viittauksien perusteella. Tämän takia ne eivät välttämättä ole käytetyimpiä ketteriä käytänteitä. Itse asiassa VersionOne (2015) raportoi suosituimmiksi viideksi ketteräksi käytänteeksi päiväpalaverit, lyhyet iteraatiot, priorisoidut kehitysjonot, iteraation suunnittelun ja retrospektiivit. Myös kirjoittajat itse esittävät lievää kritiikkiä listaa kohtaan: he tiedostavat, ettei lista ole kaiken kattava ja koska artikkelin varsinainen aihe käsittelee ohjelmiston arkkitehtuuria, sisältää lista vain käytänteitä, jotka ovat löytyneet ohjelmistoarkkitehtuuria käsittelevistä artikkeleista (Yang ym., 2016).

TAULUKKO 3 20 suosituinta ketterää käytännettä (mukaillen Yang ym., 2016, s. 159)

• Testivetoinen kehitys	• Käyttäjätarinat
• Pariohjelmointi	• Suunnittelupeli
• Jatkuva integraatio	• Iteraation suunnittelu
• Päiväpalaveri	• Iteratiivinen kehittäminen
• Refaktorointi	• Avoin työtila
• Läsnä oleva asiakas	• Yksinkertainen suunnittelu
• Tuotteen kehitysjono	• Yksikkötestaus
• Koodin yhteinen omistajuus	• Iteraation katselmointi
• Retrospektiivi	• Järjestelmän vertauskuva
• Ohjelmointistandardit	• Pienet julkaisut

3.7 Yhteenveto ketteristä periaatteista ja menetelmistä

Ketteryyttä saavuttaakseen yritys voi toimia ketterien periaatteiden mukaisesti, hyödyntää toiminnassaan ketteriä menetelmiä tai osaa menetelmien käytänteistä. Verrattuna Agile Manifeston arvoihin, ketterät periaatteet tarjoavat tarkem-

pia ohjeita siitä, miten ketteryyttä voidaan saavuttaa. Ne jättävät kuitenkin käytännön implementoinnin ketteryyttä tavoittelevan yrityksen vastuulle, jolloin yritys voi hyödyntää ketteriä periaatteita sille parhaiten soveltuvalla tavalla. Ketterät periaatteet ovat toimineet pohjana myös usealle ketterälle menetelmälle. Suosituimmat ketterät menetelmät ovat Scrum, Extreme Programming, Lean-ohjelmistokehitys ja Kanban. Nämä menetelmät antavat yritykselle tarkkoja ohjeita käytänteistä, rooleista, tuotoksista ja toimintamalleista, joilla ketteryyttä voidaan saavuttaa. Jos yritys ei halua käyttää ketterää menetelmää, se voi kuitenkin implementoida tarpeelliseksi katsomiaan ketteriä käytänteitä omaan toimintaansa. Akateemisessa kirjallisuudessa raportoiduista suosituimmista ketteristä käytänteistä suurin osa ovat XP:hen tai Scrumiin yhdistettäviä käytänteitä.

4 AIKAISEMMAT TUTKIMUKSET KETTERÄN OHJELMISTOKEHITYKSEN MENESTYSTEKIJÖISTÄ

Tutkielman tutkimuskysymykseen vastaaminen vaatii tieteellisessä kirjallisuudessa raportoitujen ketterän ohjelmistokehityksen menestystekijöiden tarkastelua. Seuraavassa tarkastellaan lähemmin aiheesta tehtyjä kvantitatiivisia tutkimuksia sekä vertaillaan tutkimuksien tuloksia toisiinsa.

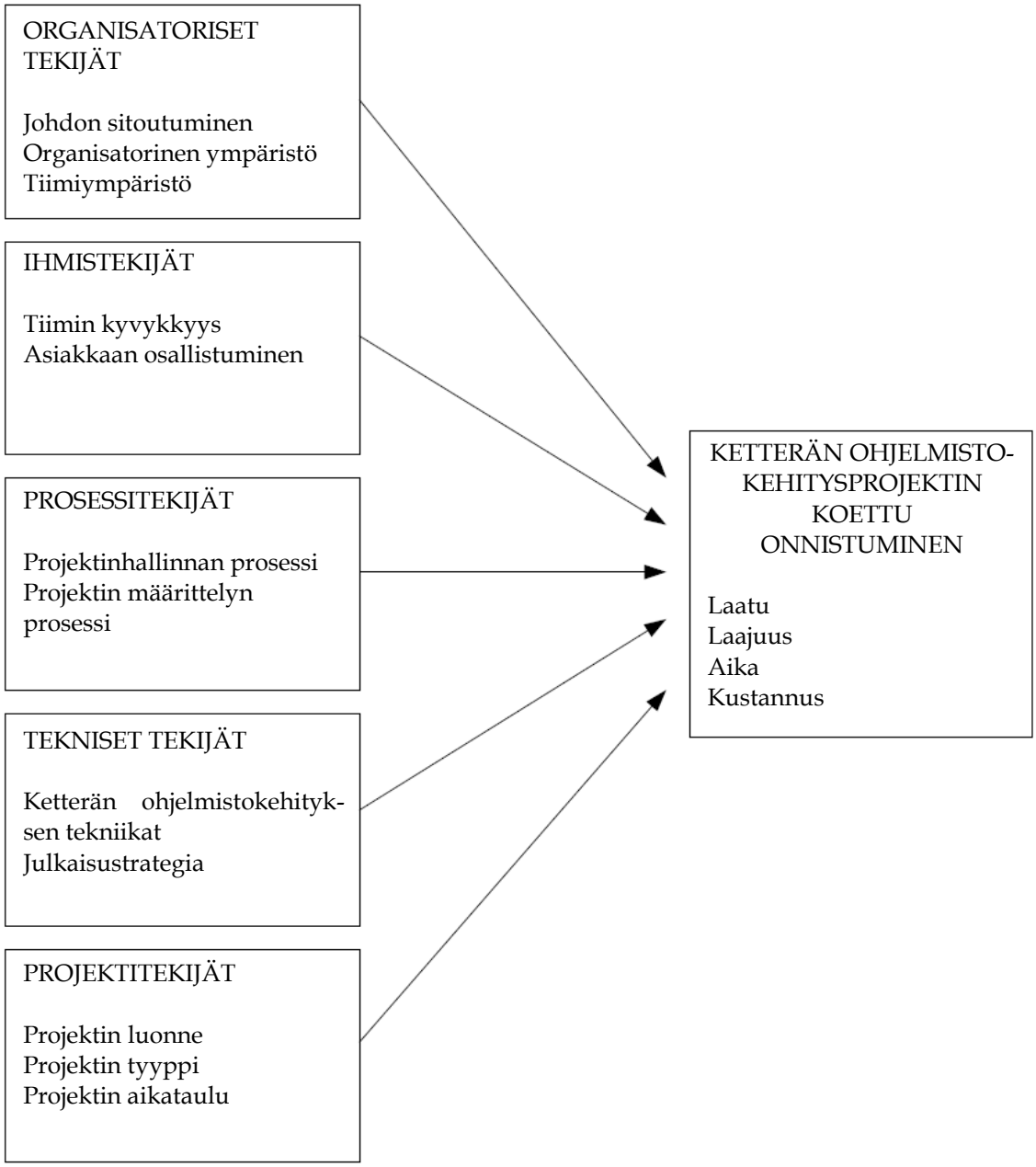
4.1 Aikaisempien tutkimusasettelujen esittely

Chow ja Cao (2008) ovat tutkineet ketterän ohjelmistokehityksen kriittisiä menestystekijöitä hyödyntäen määrällisiä menetelmiä. Heidän mukaansa virallista tutkimusta kriittisistä menestystekijöistä tässä kontekstissa ei ole aikaisemmin tehty – ketterän ohjelmistokehityksen menestyksen tutkimus on perustunut pääosin tapaustutkimuksiin sekä kokoelmiin eri ketterien projektien havainnoineista (Chow & Cao, 2008). Kyseinen tutkimus on noussut suosituksi artikkeliksi tieteellisessä kirjallisuudessa: Chuang, Luor ja Lu (2014) ovat listanneet artikkelin yhdeksi viitatuimmista vuosien 2001–2012 välillä julkaistuista ketterää ohjelmistokehitystä käsittelevistä artikkeleista, vaikka se onkin julkaistu vertailuajankohdan loppupuolella.

Tutkimuksessaan Chow ja Cao (2008) määrittelevät kriittiset menestystekijät sellaiseksi tekijöiksi, joiden tulee sisältyä toimintaan, jotta ketterä projekti olisi onnistunut. Aiempaan tutkimukseen perustuen he luokittelevat menestystekijät viiteen kategoriaan: organisaatioon liittyvät, ihmisiin liittyvät, prosesseihin liittyvät, tekniikoihin liittyvät ja projektiin liittyvät. Projektin menestyksen attribuutteina he pitävät laatua, laajuutta, aikaa ja kustannuksia. (Chow & Cao, 2008.)

Tutkimusasetelmassaan Chow ja Cao (2008) ovat nimenneet 12 lopullista tekijää luokiteltuna viiteen edellä mainittuun luokkaan, jotka linkittyvät ketterän ohjelmistokehitysprojektin menestyksen neljään attribuuttiin. Tämän poh-

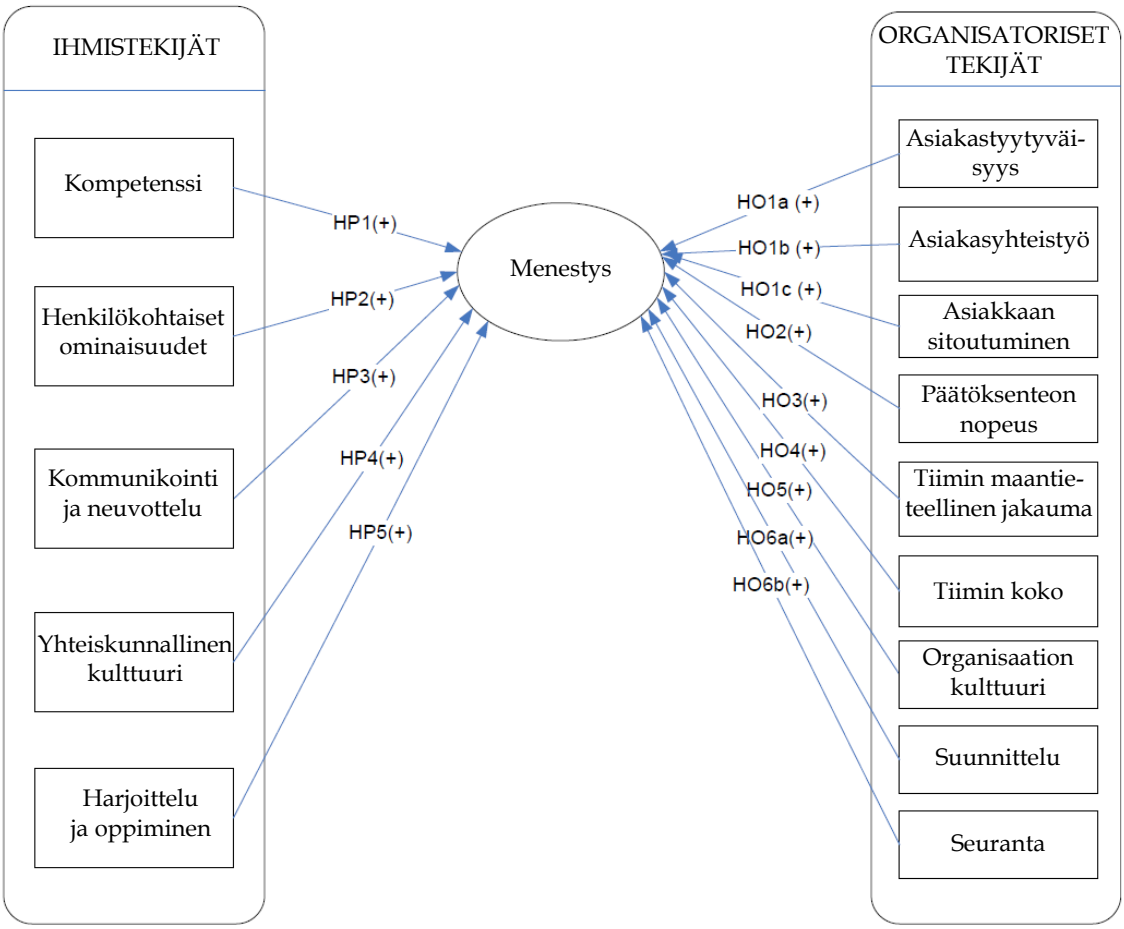
jalta tutkimuksessa pyrittiin etsimään vastaus yhteensä 48 hypoteesiin. Tutkimusasetelma on kuvattu tarkemmin kuviossa 4.



KUVIO 4 Chow'n ja Caon tutkimuksen tutkimusasetelma (Chow & Cao, 2008, s. 964)

Myös Misra ym. (2009) ovat tutkineet kriittisiä menestystekijöitä määrällisin menetelmin. Tutkimuksessa menestys määritellään lyhentyneenä aikana, pienempinä kuluina ja parantunteena laatuna. Menestystä arvioidaan viiden kriteerin avulla: lyhentyneinä julkaisuaikatauluina, kasvaneena sijoitetun pääoman tuottoprosenttina (ROI), lisääntyneenä kykynä täyttää nykyiset asiakasvaatimukset sekä vastata muuttuviin vaatimuksiin ja parantuneina liiketoimintaprosesseina. Kirjoittajat ovat esittäneet 14 hypoteettista menestystekijää, jotka on jaoteltu kahteen luokkaan: ihmisiin liittyviin ja organisaatioon liittyviin tekijöihin. Osa tekijöistä perustuu aikaisempaan

tieteelliseen kirjallisuuteen, mutta osa on kirjoittajien itsensä esittämiä. (Misra ym., 2009.) Kuviossa 5 on kuvattu tutkimuksen tutkimusasetelma.



KUVIO 5 Misran ym. esittämät hypoteettiset menestystekijät (Misra ym., 2009, s. 1873)

Sekä França ym. (2010) että Stankovic ym. (2013) ovat pyrkineet toistamaan Chow'n ja Caon (2008) tutkimuksen ja vahvistamaan raportoidut menestystekijät omissa tutkimuksissaan. Stankovic ym. (2013) ovat käyttäneet täsmälleen samaa, kuviossa 4 esitettyä, tutkimusasetelmaa kuin Chow ja Cao (2008) viisi vuotta aikaisemmin: 12 oletettua menestystekijää ja 4 menestyksen attribuuttia; yhteensä 48 hypoteesia. França ym. (2010) taas ovat tutkineet Chow'n ja Caon (2008) alkuperäisiä menestystekijöitä, mutta ovat tutkimuksessaan määritelleet projektin onnistumisen eri tavalla kuin alkuperäisessä.

4.2 Aikaisempien tutkimustuloksien esittely

Chow'n ja Caon (2008) tutkimuksessa kahdestatoista tekijästä vain kuuden todettiin olevan mahdollisia kriittisiä menestystekijöitä. Tiimiympäristö, tiimin kyvykkyys, asiakkaan osallistuminen, projektinhallinnan prosessi, ketterän oh-

jelmistokehityksen tekniikat ja julkaisustrategia olivat tekijöitä, jotka vaikuttavat positiivisesti projektin onnistumiseen.

Kolme tärkeintä tekijää olivat julkaisustrategia, joka vaikutti projektin laajuuteen, aikataulussa pysymiseen ja kustannuksiin, ketterän ohjelmistokehityksen tekniikat, joka vaikutti laatuun ja laajuuteen sekä tiimin kyvykkyys, joka vaikutti aikataulussa pysymiseen ja kustannuksiin. Täten esitetystä viidestä kategoriasta tekniset tekijät on tärkein, sillä sen kattaessa sekä julkaisustrategian että ketterän ohjelmistokehityksen tekniikat, vaikuttaa se positiivisesti kaikkiin neljään projektin menestyksen attribuuttiin. Projektitekijöiden todettiin kokonaisuudessaan olevan merkityksettömiä ketterän ohjelmistokehitysprojektin menestymisen näkökulmasta (Chow & Cao, 2008.)

Tutkimus vahvistaa monia ketterään ohjelmistokehitykseen yhdistettyjä uskomuksia ja raportoidut menestystekijät ovat osaltaan linjassa ketterien periaatteiden kanssa. Esimerkiksi julkaisustrategia vastaa ensimmäiseen ja kolmanteen Agile Manifeston ketterään periaatteeseen, ketterän ohjelmistokehityksen tekniikat voidaan yhdistää yhdeksänteen ja kymmenenteen periaatteeseen, ja tiimin kyvykkyys on linjassa viidennen periaatteen kanssa. (Chow & Cao, 2008.)

Chow ja Cao (2008) uskovat, että tulokset osoittavat myös jotkin ketterän ohjelmistokehityksen olettamukset vääriksi. Vahva liikkeenjohdon tuki ei näytä olevan kriittinen tekijä ketterässä projektissa, eivätkä ketterissä menetelmissä ohjeistetut työympäristöt ja –tavat ole kriittisiä projektin menestyksen kannalta. Esimerkiksi yhteiset työtilat ja pariohjelmoinnin mahdollistavat työpisteet eivät tuloksien mukaan ole välttämättömiä. Tällöin tiimin ei tarvitse noudattaa menetelmien ohjeita kirjaimellisesti, vaan he voivat tarvittaessa soveltaa niitä tarpeisiinsa. (Chow & Cao, 2008.)

Tulokset ovat pääosin uskottavia, mutta osaa niistä on tulkittava varauksella. Tutkimuksen perusteella voidaan kärjistetysti todeta, että kunhan ketterää ohjelmistokehitystä tekee kyvykäs tiimi, joka osaa hyödyntää ketterän ohjelmistokehityksen tekniikoita oikein ja julkaisee ohjelmistoa ketterien periaatteiden mukaisesti, tulee projekti onnistumaan.

On epäintuitiivista, että tuloksien mukaan projektiin liittyvillä tekijöillä ei olisi vaikutusta projektin onnistumiseen. Myös Chow ja Cao (2008) ovat esittäneet joitakin puutteita tutkimuksessaan. Tutkittu joukko oli suhteellisen pieni ja ketteristä menetelmistä XP oli yliedustettuna 53 prosentin osuudellaan. He ehdottavatkin, että tutkimus tulisi toistaa viiden ja kymmenen vuoden kuluttua alkuperäisen ilmestymisestä, jotta voitaisiin kartoittaa mahdollisia uusia menestystekijöitä ja tutkia, väheneekö joidenkin tekijöiden merkitys nykyisestä.

Françan ym. (2010) tutkimustulokset tukevat Chow'n ja Caon (2008) alkuperäisiä tuloksia. Vaikka tutkimustuloksien perusteella menestystekijät saavatkin hieman erilaisen prioriteetti-järjestyksen, raportoivat França ym. (2010) kaikki samat tekijät joko kriittisinä menestystekijöinä tai tärkeinä tekijöinä kuin alkuperäinen Chow'n ja Caon (2008) tutkimus: Françan ym. (2010) mukaan julkaisustrategia sekä projektin hallinnan prosessi ovat kriittisiä menestystekijöitä ja asiakkaan osallistuminen, ketterän ohjelmistokehityksen tekniikat, tiimin kyvykkyys sekä tiimiympäristö ovat tärkeitä tekijöitä projektin onnistumiselle.

Näin ollen tulokset vahvistavat ennen kaikkea julkaisustrategian merkitystä ketterän ohjelmistokehityksen menestystekijänä. Samalla se osoittaa myös samoja ketterän ohjelmistokehityksen olettamuksia vääräksi kuin alkuperäinen tutkimus.

Yllättävää kyllä, Stankovic ym. (2013) eivät pystyneet vahvistamaan yhtäkään Chow'n ja Caon (2008) alkuperäisen tutkimuksen tekijää kriittisenä menestystekijänä. Sen sijaan heidän tulostensa perusteella kolme uutta tekijää olisivat mahdollisia kriittisiä menestystekijöitä ketterässä ohjelmistokehityksessä: projektin määrittelyn prosessi, projektin luonne ja projektin aikataulu. Projektin hallinnan prosessin todettiin myös tämän tutkimuksen tulosten perusteella olevan tärkeä tekijä, mutta eri syystä kuin alkuperäisessä tutkimuksessa: Chow ja Cao (2008) raportoivat, että se vaikuttaisi laatuun, kun taas Stankovic ym. (2013) raportoivat sen vaikuttavan kustannuksiin. Kuten alkuperäinen sekä Françan ym. (2010) toistama tutkimus, myös Stankovicin ym. (2013) tutkimuksen tulokset osoittavat joitakin ketterän ohjelmistokehityksen olettamuksia vääriksi. Tämänkään tutkimuksen tuloksien valossa johdon sitoutumisella, organisatorisella ympäristöllä tai projektin tyypillä ei näyttäisi olevan merkitystä menestyksen näkökulmasta.

Kun erot tutkimuksen välillä ovat näin suuria, on syitä etsittävä otannasta, eikä mittausvirheen mahdollisuuttakaan voi poistaa. Stankovicin ym. (2013) tutkimuksessa vastaajista suurin osa käytti Scrumia, mikä on linjassa ketterien menetelmien käytön yleisen kehityksen kanssa. Alkuperäisessä tutkimuksessa suurin osa vastaajista edusti XP:tä (Chow & Cao, 2008), mikä saa pohtimaan, olisiko tällä vaikutusta tuloksien merkittävään eroon. Kirjoittajien suorittaman varianssianalyysin avulla tämä mahdollisuus on kuitenkin poistettu (Stankovic ym., 2013). Kirjoittajat tekevät kuitenkin tärkeän huomion tutkimukseen osallistuneista yrityksistä: koska Länsi-Balkanin alueella, jossa tutkimus toteutettiin, on paljon yrityksiä, joita käytetään ulkoistettuna palveluna, johtaa tämä tilanteisiin, jossa kehitystiimi muodostuu eri maassa olevasta ”varsinaisesta tiimistä” ja Balkanissa toimivasta ”vahvistustiimistä”. Kirjoittajat päättelevätkin, että ero tuloksissa selittyy näiden ”sekoitettujen” tiimien vaikutuksella. (Stankovic ym., 2013.) Vaikka ilmiö varmasti vaikuttaakin tuloksiin, on silti vaikea uskoa, että vaikutus olisi niin suuri, ettei yksikään alkuperäisessä tutkimuksessa esitetty menestystekijä olisi pätevä tässä tilanteessa. Selvimpänä rajoitteena tutkimukselleen kirjoittajat näkevät otoskoon: vastaajia oli yhteensä 23 (Stankovic ym., 2013), mikä on vain joitakin prosentteja alkuperäisen tutkimuksen vastaajamäärästä. Tutkimuksien eroja on vertailtu lisää seuraavassa luvussa.

Misran ym. (2009) tutkimuksen tuloksien mukaan yhdeksän tekijää liittyvät merkittävästi menestykseen: asiakastyytyväisyys, asiakasyhteistyö, asiakkaan sitoutuminen, päätöksenteon nopeus, organisaation kulttuuri, seuranta, henkilökohtaiset ominaisuudet, yhteiskunnallinen kulttuuri sekä harjoittelu ja oppiminen. Tuloksien analyysin perusteella edellä mainituista tekijöistä seitsemän vaikutti erittäin merkittävästi menestykseen – vain asiakastyytyväisyys ja henkilökohtaiset ominaisuudet olivat lähellä tilastollisesti melkein merkitsevää arvoa 0.05. (Misra ym., 2009.)

Tutkimuksen tulokset ovat suurimmalta osin uskottavia. On järkevää, että päätöksenteon nopeutta ja asiakkaaseen liittyviä tekijöitä pidetään tärkeinä, sillä onhan ketterän ohjelmistokehityksen taustalla ajatus luoda ohjelmistoa nopeasti siten, että se täyttää asiakkaan nykyiset ja tulevat vaatimukset, mikä edellyttää asiakkaan osallistumista kehitystyöhön. Toisaalta osa tuloksien perusteella hylätyistä tekijöistä ja menestykseen vaikuttavien tekijöiden keskinäinen tärkeysjärjestys on vaikea hyväksyä. Tuloksien mukaan henkilön kompetenssilla ei olisi vaikutusta menestykseen (Misra ym., 2009), mikä ei ole uskottavaa. Se on Chow'n ja Caon (2008) sekä Françan ym. (2010) tutkimustuloksien vastainen tulos ja myös Misra ym. (2009) esittävät epäilyksensä tuloksen luotettavuudesta.

Korkein korrelaatio menestyksen kanssa näyttäisi tuloksien mukaan olevan yhteiskunnallisella kulttuurilla (Misra ym., 2009). Tämä tarkoittaa käytännössä, että tiimit tulisi muodostaa henkilöistä, jotka tulevat kulttuureista, joissa arvostetaan avoimuutta, dynaamisuutta ja eteenpäin suuntautuneisuutta. Lisäksi on hyödyllistä, jos tiimin jäsenet edustavat yhtä kulttuuria. (Misra ym., 2009.) IT-ala on tunnettu monikulttuurisista työyhteisöistä ja on vaikea uskoa, että kulttuurin merkitys olisi näin suuri, ottaen huomioon, että ohjelmistokehitystä on tehty jo hyvin pitkään monikulttuurisessa ympäristössä. Jos yhteiskunnallinen kulttuuri olisi kriittinen menestystekijä, voisi olettaa, että myös Chow ja Cao (2008) olisivat tutkineet sitä. Lisää tutkimustuloksien vertailua on esitetty luvussa 4.3.

4.3 Aikaisempien tutkimustuloksien vertailu ja yhteenveto

Edellä esiteltyjen tutkimuksien tuloksia tarkastellessa voidaan todeta, että kuten ketterän ohjelmistokehityksen tutkimus kokonaisuutena, myös sen kriittisten menestystekijöiden tutkimus on hankalasti yleistettävissä. Taulukkoon 4 on koostettu edellä esiteltyjen tutkimusten tulokset. Taulukkoa tulkitessa tulee huomioida, että vain Chow ja Cao (2008) raportoivat osan vahvistetuista tekijöistä olevan kriittisiä menestystekijöitä ja osan olevan mahdollisesti tärkeitä tekijöitä menestyksen näkökulmasta. Muissa tutkimuksissa tutkijat ovat raportoineet tuloksissaan yksinkertaisesti menestystekijät. Näiden osalta menestystekijät on kuitenkin taulukossa jaettu kriittisiin ja tärkeisiin tekijöihin analysoimalla tutkimuksien dataa, jotta tekijöiden priorisointi olisi näkyvämpää.

Ketterän ohjelmistokehityksen kriittisistä menestystekijöistä on vaikea saada kokonaisvaltaista kuvaa. Oman haasteensa tuloksien tulkinnalle asettaa se, että Misran ym. (2009) tutkimuksessa tekijöiden määrittely eroaa Chow'n ja Caon (2008), Françan ym. (2010) sekä Stankovicin ym. (2013) tutkimuksien määrittelystä, eikä Misran ym. (2009) tutkimuksessa käsitellä kaikkia samoja tekijöitä kuin muissa kolmessa tutkimuksessa. Mutta vaikka tuloksia tulkitsisi vain Chow'n ja Caon (2008), Françan ym. (2010) sekä Stankovicin ym. (2013) tutkimuksen välillä, ei kiistattomia yhteneväisyyksiä menestystekijöistä löydy: kaikki Chow'n ja Caon (2008) raportoimat kriittiset menestystekijät ovat Stankovicin

ym. (2013) tutkimuksessa tekijöitä, joilla ei ole vaikutusta menestykseen – ja sama ilmiö pätee toisinpäin. Menestystekijöiden osalta näiden tutkimuksien tulokset näyttäisivät osoittavan yksiselitteisesti vain, että projektin hallinnan prosessi on tärkeä (Chow & Cao, 2008; Stankovic ym., 2013) tai jopa kriittinen (França ym., 2010) menestyksen näkökulmasta.

TAULUKKO 4 Aikaisemmissa tutkimuksissa esitettyjen menestystekijöiden vertailu

	Chow & Cao, 2008	**Misra ym., 2009**	**França ym., 2010**	**Stankovic ym., 2013**
Kriittiset menestystekijät	Julkaisustrategia Ketterän ohjelmistokehityksen tekniikat Tiimin kyvykkyys	Asiakasyhteistyö Asiakkaan sitoutuminen Päätöksenteon nopeus Organisaation kulttuuri Seuranta Yhteiskunnallinen kulttuuri Harjoittelu ja oppiminen	Julkaisustrategia Projektin hallinnan prosessi	Projektin luonne
Tärkeät tekijät	Asiakkaan osallistuminen Projektin hallinnan prosessi Tiimiympäristö	Asiakastyytyväisyys Henkilökohtaiset ominaisuudet	Asiakkaan osallistuminen Ketterän ohjelmistokehityksen tekniikat Tiimin kyvykkyys Tiimiympäristö	Projektin aikataulu Projektin hallinnan prosessi Projektin määrittelyn prosessi
Tekijät, joilla ei vaikutusta	Johdon sitoutuminen Organisatorinen ympäristö Projektin määrittelyn prosessi Projektin luonne Projektin tyyppi Projektin aikataulu	Tiimin maantieteellinen jakauma Tiimin koko Suunnittelu Kompetenssi Kommunikointi ja neuvottelu	Johdon sitoutuminen Organisatorinen ympäristö Projektin määrittelyn prosessi Projektin luonne Projektin tyyppi Projektin aikataulu	Asiakkaan osallistuminen Johdon sitoutuminen Julkaisustrategia Ketterän ohjelmistokehityksen tekniikat Organisatorinen ympäristö Projektin tyyppi Tiimiympäristö Tiimin kyvykkyys

Taulukko osoittaa ristiriitoja eri tutkimuksissa raportoitujen menestystekijöiden välillä. Toisaalta tuloksia tarkemmin tulkitessa voidaan raportoiduista tekijöistä löytää myös joitakin yhteneväisyyksiä. Ehkäpä selvin epäkohta, johon viitattiin jo luvussa 4.2, on yhteiskunnallisen kulttuurin merkitys eri tutkimuksissa. Misran ym. (2009) tutkimuksen tuloksien mukaan tämä olisi kriittisin menestysteki-

jä ketterässä ohjelmistokehityksessä. Kuten aikaisemminkin todettiin, vaikuttaa väite epäuskottavalta. Jos tekijä todella olisi kriittinen menestystekijä, voisi olettaa, että ainakin myöhemmin julkaistut Françan ym. (2010) tai Stankovicin ym. (2013) tutkimukset olisivat ottaneet kantaa aiheeseen.

Organisaation kulttuurin merkitys on myös melko selvästi ristiriitainen tutkimuksien välillä. Misra ym. (2009) raportoivat sen olevan kriittinen menestystekijä, kun kolmessa muussa tutkimuksessa organisatorisen ympäristön, joka mittaa samoja attribuutteja, on todettu olevan tekijä, jolla ei ole vaikutusta projektin menestykseen. Toisaalta organisaation kulttuuri on esimerkki siitä, miten tuloksien tulkinta on hankalaa, koska eri tekijöiden määrittelyt eroavat tutkimuksien välillä: Misran ym. (2009) määritelmä organisaation kulttuurista sisältää samoja attribuutteja kuin Chow'n ja Caon (2008) määritelmä tiimin kyvykkyydestä, jonka he ovat todenneet olevan kriittinen menestystekijä ja França ym. (2010) ovat todenneet olevan tärkeä tekijä. Näiden tutkimuksien tuloksien perusteella näyttäisikin siltä, ettei organisaation kulttuurilla ja ketteryyttä tukevilla organisaatiorakenteilla ole merkitystä menestyksen näkökulmasta, kunhan tiimin esimiehet tukevat ketteryyttä ja osaavat hyödyntää ketteriä menetelmiä.

Toinen tiimin kyvykkyyteen liittyvä epäkohta on sen ristiriita Misran ym. (2009) tuloksien kompetenssi-tekijän kanssa: Chow ja Cao (2008) sekä França ym. (2010) pitävät kriittisenä tekijänä tiimin jäseniä, joilla on korkea kompetenssi ja osaaminen, kun taas Misran ym. (2009) tuloksien mukaan tällä ei olisi merkitystä. Kirjoittajat kuitenkin osoittivat epäilyksensä tulosta kohtaan ja uskoivat, että asiaa pitäisi tutkia tarkemmin (Misra ym., 2009). Toisaalta Stankovicin ym. (2013) tulokset eivät vahvista kompetenssin merkitystä menestykselle. Tämä vaikuttaa epäuskottavalta. Ohjelmistokehitys on monilta osin luonteeltaan korkeaa substanssiosaamista vaativaa työtä, joten on oletettavaa, että parempia tuloksia saadaan, jos kehitystyöhön osallistuvat henkilöt ovat osaavia.

Chow'n ja Caon (2008) tiimin kyvykkyys sisältää myös joitakin yhtenäistettäviä attribuutteja Misran ym. (2009) tuloksista. Tiimin kyvykkyyttä määritellään tiimin jäsenten korkean motivaation ja sopivan teknisen kouluttamisen avulla. Misran ym. (2009) tuloksissa menestystekijöinä on raportoitu henkilökohtaiset ominaisuudet, jonka yhtenä attribuuttina on korkea motivaatio. Lisäksi yhtenä menestystekijänä on raportoitu harjoittelu ja oppiminen. Joten ainakin näiden tekijöiden merkitys menestystekijöinä on yhtenäinen Chow'n ja Caon (2008), Misran ym. (2009) sekä Françan ym. (2010) tutkimuksien välillä.

Tiimiympäristön osalta Chow ja Cao (2008) ovat todenneet, että jos kehitystyötä tehdään pienissä tiimeissä, jotka työskentelevät samassa paikassa, on menestyminen todennäköisempää. Misran ym. (2009), Françan ym. (2010) ja Stankovicin ym. (2013) tulokset eivät kuitenkaan tue tätä väitettä. Aiheesta on vaikea tehdä johtopäätöksiä. Eri ketterien menetelmien ohjeissa otetaan kantaa tiimin kokoon ja samoissa tiloissa toimimiseen (esimerkiksi XP ja Scrum), joten voisi olettaa, että näillä tekijöillä olisi merkitystä. Toisaalta teknologian kehittyminen on helpottanut maantieteellisesti jakautuneiden tiimien työskentelyä, joka on voinut vähentää tekijöiden merkitystä.

Asiakkaaseen liittyvät tekijät on helppo hyväksyä kriittisenä menestystekijänä. Chow ja Cao (2008) sekä França ym. (2010) raportoivat asiakkaan osallistumisen yhtenä tärkeistä tekijöistä ja Misra ym. (2009) ovat raportoineet asiakasyhteistyön, asiakkaan sitoutumisen ja asiakkaan tyytyväisyyden menestystekijöinä. Vaikka Stankovicin ym. (2013) tulokset eivät vahvista Chow'n ja Caon (2008) sekä Françan ym. (2010) tuloksia, on silti oletettavaa, että asiakkaan osallistumisella on suuri vaikutus menestymisen näkökulmasta. Asiakasyhteistyö on myös tekijä, jonka merkitys väistämättä vähenee, jos kehitystyötä tehdään ympäristössä, jossa suuri osa Stankovicin ym. (2013) tutkimukseen vastanneista toimi: kun kehitystiimin rooli on enemmänkin ”vahvistustiimi”, jolle ”varsinainen tiimi” (joka hoitaa kommunikaation asiakkaan kanssa) delegoi tehtäviä, ei asiakasyhteistyö ole yhtä tärkeää. Tämä saattaa selittää erot tutkimustuloksissa.

Yhteenvetona on todettava, että ketterän ohjelmistokehityksen kriittisistä menestystekijöistä on vaikea tehdä yleistettäviä johtopäätöksiä. Kun tekijöitä tulkitaan yksityiskohtaisemmin, voidaan tutkimuksien välillä löytää joitakin yhteneväisyyksiä. Tämä edellyttää kuitenkin Stankovicin ym. (2013) tutkimustulosten osittaista sivuuttamista, jolloin ei voida esittää aidosti empiiriseen tutkimukseen perustuvia, luotettavia johtopäätöksiä. Tämä huomioiden voidaan todeta, että aikaisempien tutkimuksien perusteella ketterän ohjelmistokehityksen menestystekijöitä ovat ketterän ohjelmistokehityksen mukainen julkaisustrategia, projektin hallinnan prosessi, asiakkaan osallistuminen sekä motivoituneet, osaavat ja sopivaa koulutusta saavat tiimin jäsenet.

5 EMPIIRISEN TUTKIMUKSEN MENETELMÄ

Tässä luvussa käydään läpi empiiriseen tutkimukseen valittu tutkimusmenetelmä. Aluksi esitetään perusteet valitulle tutkimusmenetelmälle, minkä jälkeen esitellään tutkimuksen tiedonkeruumenetelmä. Tämän jälkeen esitellään haastattelurunko ja perustellaan rungon suhteen tehtyjä valintoja. Lopuksi kerrotaan haastateltavien valinnasta, haastattelujen kulusta, aineiston analysointimenetelmä sekä arvioidaan tutkimuksen luotettavuutta.

5.1 Laadullinen tutkimus

Tämän tutkielman tutkimusmenetelmäksi valittiin kvalitatiivinen eli laadullinen tutkimus. Laadullisen tutkimuksen lähtökohtana on todellisen elämän kuvaaminen ja pyrkimys tutkittavan ilmiön mahdollisimman kokonaisvaltaiseen tutkimiseen (Hirsjärvi, Remes & Sajavaara, 2004). Sen sijaan, että tutkimuksessa pyrittäisiin todentamaan ennalta-asetettuja väittämiä tai saavuttamaan tilastollista yleistettävyyttä, on laadullisen tutkimuksen tarkoitus pikemminkin kuvata, ymmärtää, tulkita ja paljastaa tosiasioita tutkittavasta kohteesta (Hirsjärvi ym., 2004; Eskola & Suoranta, 2008). Tunnusomaista laadulliselle tutkimukselle on, että tutkijan pyrkimyksenä on tuoda esiin tutkittavien havaintoja, näkemyksiä, mielipiteitä sekä saada kuuluviin heidän äänensä (Hirsjärvi ym., 2004; Hirsjärvi & Hurme, 2014). Toisaalta laadullisessa tutkimuksessa myös tutkijan ajatellaan olevan keskeisessä roolissa, minkä takia menetelmää on silloin tällöin pidetty subjektiivisena tapana tuottaa tietoa (Eskola & Suoranta, 2008).

Tässä tutkielmassa pyritään tutkimaan mahdollisimman kokonaisvaltaisesti ketterän ohjelmistokehityksen menestystekijöitä. Tarkoituksena on saada selville yksityiskohtaista tietoa tutkittavasta ilmiöstä, mikä vaatii tarkkaa paneutumista tutkittavien havaintoihin ja näkemyksiin ketterästä ohjelmistokehityksestä omassa elämässään. Näin ollen laadullisen tutkimuksen voitiin olettaa auttavan tutkimusongelman ratkaisussa parhaiten.

Laadullista tutkimusta verrataan usein kvantitatiiviseen eli määrälliseen tutkimukseen (Eskola & Suoranta, 2008) Keskeisimpänä erona laadulliseen tutkimukseen voidaan pitää pyrkimystä tunnistaa syy-seuraussuhteita tutkittavan ilmiön sisällä numeerisen aineiston ja tilastollisen analyysin avulla. Muita keskeisiä piirteitä kvantitatiivisessa tutkimuksessa ovat muun muassa tutkimuksen pohjautuminen aiempaan teoriaan, johtopäätökset aiemmista tutkimuksista ja hypoteesien esittäminen. (Hirsjärvi ym., 2004.) Tämän tutkielman tapauksessa aikaisempaa, empiirisesti testattua teoriaa ketterän ohjelmistokehityksen menestystekijöistä ei ollut, ja kuten luvussa 4 todettiin, ei aikaisempien tutkimuksien tuloksista voida tehdä vahvoja johtopäätöksiä, jolloin myös hypoteesien esittäminen on vaikeaa. Lisäksi kun tarkoitus on saada yksityiskohtaista tietoa todellisesta elämästä, on kvantitatiivista tiedonkeruumenetelmää (esimerkiksi kyselylomaketta) käytettäessä riski, että vastaukset jäävät pinnallisiksi. Näiden syiden takia kvantitatiivisen tutkimuksen ei katsottu soveltuvan tutkielman tutkimusmenetelmäksi.

5.2 Tiedonkeruumenetelmä

Laadullisessa tutkimuksessa aineistoa voidaan hankkia muun muassa kyselyillä, haastatteluilla, havainnoinneilla tai olemassa oleviin dokumentteihin tutustumalla, mutta tyypillistä laadulliselle tutkimukselle on kuitenkin suosia tiedon keruun instrumenttina ihmistä (Hirsjärvi ym., 2004; Eskola & Suoranta, 2008). Tällöin haastattelut ovat hyvä keino kerätä aineistoa. Haastattelut mahdollistavat tutkittavien ja tutkijan välittömän vuorovaikutuksen, mikä avulla voidaan kerätä tutkittavien aitoja kokemuksia ja tulkintoja tutkittavasta aiheesta (Schultze & Avital, 2011). Lisäksi haastattelujen avulla saadaan usein kuvaavia esimerkkejä (Hirsjärvi & Hurme, 2014). Tiedonkeruu haastattelujen avulla on perusteltua erityisesti silloin, kun tiedetään, että tutkimuksen aihe tuottaa hyvin erilaisia vastauksia, jolloin tutkijan on vaikea tietää vastauksien suuntia etukäteen, ja kun halutaan selventää tai syventää vastauksia esimerkiksi pyytämällä perusteluja sekä esittämällä lisäkysymyksiä tutkittaville. (Hirsjärvi ym., 2004.)

Kuten luvussa 4 todettiin, ovat aikaisemmat empiiriset tutkimukset tuottaneet toisistaan eroavia tuloksia ketterän ohjelmistokehityksen menestystekijöiksi, ja vastauksien voitiin olettaa eroavan paljonkin toisistaan. Sen lisäksi, että tämän tutkielman tuloksia peilattiin olemassa olevaan kirjallisuuteen aiheesta, on tutkielmassa myös vahva käytännön näkökulma: mielenkiinnon kohteena ovat erityisesti organisaatioiden päivittäisessä työskentelyssä esiintyvät tekijät, joiden avulla voidaan saavuttaa onnistuneita ketteriä ohjelmistokehitysprojekteja. Tämän näkökulman esiintuomisen uskottiin vaativan tutkittavien aitoja kokemuksia, esimerkkejä ja perusteluja sekä lisäkysymyksien esittämistä aineistoa kerättäessä, jolloin haastattelut todettiin hyväksi tiedonkeruumenetelmäksi.

Hirsjärvi ja Hurme (2014) pitävät erilaisten haastattelunimikkeiden valikoimaa kirjavana ja osin jopa sekavana. Haastattelutyypit erotellaan usein sen

mukaan, kuinka strukturoitu ja muodollinen haastattelutilanne on. Tällöin ääripäitä edustavat strukturoitu haastattelu ja avoin haastattelu. (Hirsjärvi ym., 2004; Eskola & Suoranta, 2008.) Strukturoidussa haastattelussa kysymykset ja niiden muotoilu ovat kaikille haastateltaville samat ja ne esitetään samassa järjestyksessä. Haastattelutilannetta voidaan verrata ennalta määritellyn kyselylomakkeen täyttämiseen, minkä takia tästä haastattelutyypistä käytetään myös nimitystä lomakehaastattelu. (Hirsjärvi ym., 2004.) Toisessa ääripäässä avoin haastattelu on kaikista haastattelun muodoista lähimpänä keskustelua (Eskola & Suoranta, 2008). Tällöin haastattelulla ei ole kiinteää runkoa, vaan haastattelija selvittää haastateltavien ajatuksia sen mukaan, kun ne tulevat esille keskustelun edetessä. Avoimessa haastattelussa myös aihe voi muuttua keskustelun aikana. (Hirsjärvi ym., 2004.)

Näiden ääripäiden välimuodosta ei ole vakiintunutta määritelmää, mutta usein käytetään termejä teemahaastattelu tai puolistrukturoitu haastattelu kuvaamaan haastattelutyyppiä, joka sijoittuu muodollisuudeltaan strukturoidun ja avoimen haastattelun väliin (Hirsjärvi & Hurme, 2014). Yksinkertaisimmillaan tätä haastattelutyyppiä voidaan luonnehtia haastatteluna, jossa on keskeneräinen käsikirjoitus ja tilaa improvisaatiolle (Myers & Newman, 2007). Tässä tutkielmassa käytetään termiä teemahaastattelu kuvaamaan kyseistä haastattelutyyppiä, sillä kuten Hirsjärvi ja Hurme (2014) esittävät, termi kertoo tällöin, mikä haastattelussa on oleellisinta: yksityiskohtaisten kysymysten sijaan, haastattelua viedään eteenpäin tiettyjen keskeisten teemojen varassa. Teemahaastattelu luo käsiteltävien teemojen kautta avointa haastattelua tiukemmat rajat haastattelutilanteelle, mutta antaa haastateltavalle paremmat mahdollisuudet yksilöllisten tulkintojen ja mielipiteiden esittämiseen kuin strukturoitu haastattelu. Kysymysten tarkka muoto ja järjestys voivat vaihdella, eikä kaikkia kysymyksiä välttämättä käydä läpi kaikkien haastateltavien kanssa, mutta aihe ja käsiteltävät teemat säilyvät samoina kaikissa haastattelutilanteissa. (Eskola & Suoranta, 2008; Hirsjärvi & Hurme, 2014.)

Eri haastattelutyypeistä teemahaastattelu sopi parhaiten tutkielman tutkimusongelman ratkaisemiseen. Kun haastatteluja kuljetettiin ennalta määritettyjen teemojen varassa, pitäydyttiin varmemmin tutkimusongelman kannalta olennaisissa aiheissa – esimerkiksi avoimessa haastattelussa olisi saattanut olla hankala säilyttää keskustelun fokus haastateltavan oman organisaation sisällä ja keskustelu olisi voinut ajautua liiaksi menestystekijöihin yleensä. Toisaalta strukturoitu haastattelu olisi tuskin jättänyt tarpeeksi tilaa haastateltavan mielipiteille, perusteluille ja esimerkeille, jolloin tutkimusaiheen kokonaisvaltainen ymmärtäminen ei olisi ollut mahdollista ja tulokset olisivat jääneet pinnallisiksi. Teemahaastattelussa esiin nouseviin menestystekijöihin voitiin keskittyä yksityiskohtaisesti, ilman, että haastattelutilanteessa olisi ollut painetta käydä kaikki kysymykset läpi.

5.3 Haastattelurunko

Tutkielman teemahaastattelun runko (liite 1) muodostuu viidestä osiosta:

- Taustatiedot
- Ketteryys
- Onnistumisen määrittely
- Menestystekijät – haastateltavan esittämät
- Menestystekijät – mielipide kirjallisuudessa esitetyistä

Taustatiedot-osiossa kartoitettiin haastateltavan edustaman organisaation ominaisuuksia muun muassa yrityksen koon, päämarkkinoiden ja kehitettävien ohjelmistojen osalta. Tutkimusongelman kannalta tietojen ajateltiin auttavan ymmärtämään, miten yrityksen ominaisuudet vaikuttavat sille tärkeisiin menestystekijöihin.

Ketteryys-osiossa pyrittiin saamaan tietoa siitä, millä tavalla yritys on ketterä: käyttääkö yritys ketterää menetelmää, ketteriä käytänteitä vai toimiiko se ketterien periaatteiden mukaisesti. Tämän voitiin olettaa vaikuttavan menestystekijöihin esimerkiksi siten, että tiettyä menetelmää käyttävät kokisivat prosessien, työkalujen ja käytänteiden merkityksen menestystekijöinä korkeampana kuin muiden menestystekijöiden.

Onnistumisen määrittely -osiossa kartoitettiin, kuinka yritys määrittelee onnistuneen ohjelmistokehitysprojektin ja mitataanko onnistumista joillakin mittareilla, jolloin haastateltavien vastauksia voitiin verrata aikaisempiin tutkimuksiin.

Menestystekijöistä pyrittiin saamaan tietoa kahden osion avulla. Ensin kartoitettiin haastateltavan omia mielipiteitä, omin sanoin kerrottuna. Kriittisen menestystekijän määritelmä kerrattiin osion alussa, jotta haastateltava keskittyisi vain oman organisaationsa toiminnalle tärkeimpiin menestystekijöihin. Keskustelua pyrittiin viemään myös yksityiskohtiin ja käytännön esimerkkeihin apukysymyksillä mahdollisista rooleista, käytänteistä sekä prosesseista, joiden kautta menestystekijöitä saavutetaan.

Pelkän suoraan kysymisen lisäksi myös Chow'n ja Caon (2008) sekä Misran ym. (2009) aikaisemmin tekemissä tutkimuksissa esitellyt mahdolliset menestystekijät operationalisointiin osaksi haastattelurunkoa (liite 2). Tällä oli kolme päätavoitetta. Ensin haluttiin varmistaa, että haastatteluissa saataisiin kerättyä mahdollisimman kokonaisvaltaista aineistoa – listan avulla myös haastattelutilanteessa mahdollisesti haastateltavalta mainitsematta jääneet menestystekijät saatiin osaksi keskustelua. Toiseksi listalla pyrittiin parantamaan tutkimuksen luotettavuutta: kun esitellyt tekijät olivat aikaisemmista tutkimuksista, voitiin paremmin varmistaa, että haastateltavat ymmärsivät tekijät oikein ja puhuivat samoista menestystekijöistä kuin aikaisemmissakin tutkimuksissa. Kolmanneksi listan avulla voitiin kartoittaa haastateltavan näkemyksiä myös sellaisista tekijöistä, joilla on vähäinen merkitys ohjelmistokehitysprojektin on-

nistumisen kannalta. Näin tuloksia voitiin verrata aikaisemmissa tutkimuksissa esitettyihin tekijöihin, joiden ei todettu olevan menestystekijöitä – tällaisien tuloksien saaminen vain kysymällä olisi todennäköisesti ollut hankalaa. Epäselvyyden ja erilaisten tulkintojen välttämiseksi jokaisesta menestystekijästä kirjoitettiin lyhyt kuvaus, joka pohjautui siihen, miten menestystekijöitä on kuvailtu tai kartoitettu edellä mainituissa tutkimuksissa. Osa kirjallisuudessa raportoiduista menestystekijöistä olivat samankaltaisia tai sisälsivät samoja attribuutteja. Samankaltaisia menestystekijöitä yhdistettiin, jotta vältettäisiin epäselvyyttä menestystekijöiden tulkinnassa.

5.4 Haastateltavien valinta ja haastattelut

Laadulliselle tutkimukselle on ominaista, että kohdejoukko valitaan tarkoituksenmukaisesti (Hirsjärvi ym., 2004). Tämän tutkielman tapauksessa haastateltavat valittiin suomalaisista ketterää ohjelmistokehitystä tekevistä ohjelmistoyrityksistä. Kohdejoukkoon kuuluvia yrityksiä kartoitettiin erilaisista yrityslistauksista kuten IT Expertise Wikistä (www.itewiki.fi) sekä tutkijan tuntemista ohjelmistoyrityksistä. Alustavasti valittuihin yrityksiin tutustuttiin tarkemmin muun muassa yrityksien verkkosivuilla olevan sekä muista julkisista lähteistä löytyvän tiedon avulla, millä pyrittiin varmistamaan, että yritykset edustavat kohdejoukkoa mahdollisimman hyvin.

Haastateltavien roolia tai titteliä ei ollut tarkoituksenmukaista rajata tiukasti, kuten ei myöskään valittavien kohdeyrityksien liikevaihtoluokkaa tai henkilöstömäärää. Haastateltavaksi etsittiin henkilöä, joka on mukana kohdeyrityksen asiakasprojekteissa, mutta jolla olisi myös mahdollisuuksien mukaan hyvä tuntemus kyseisen yrityksen liiketoiminnasta ja toimintamalleista yleisesti. Kohdeyrityksien henkilöstöstä kontaktoitiin oletettavasti sopivia henkilöitä, joille selostettiin tutkielman tausta ja haastattelun tarkoitus. Kymmenestä kontaktista seitsemän lupautui haastatteluun. Yksi haastateltava joutui kuitenkin perumaan osallistumisensa aikataulusyihin vedoten, eikä uutta sopivaa aikaa löydetty, joten yhteensä haastatteluja toteutettiin kuusi. Jokainen haastateltava edusti eri ohjelmistokehitysyritystä.

Kaikki haastattelut toteutettiin Skype for Business -sovelluksen avulla, jolla ne tallennettiin ja tallenteet litteroitiin. Haastattelujen kestot vaihtelivat hieman lyhimmän kestäessä noin 44 minuuttia ja pisimmän noin 55 minuuttia. Keskimäärin haastattelut kestivät noin 48 minuuttia. Haastattelun aluksi kerrattiin tutkielman tavoitteet, haastattelun rakenne ja painotettiin myös, että tulokset raportoitaisiin anonyymisti. Teemahaastattelulle ominaiseen tapaan haastateltaville annettiin mahdollisuus kertoa yksityiskohtia käsiteltävistä teemoista ja heille esitettiin tarvittaessa lisäkysymyksiä, minkä takia haastattelut etenivät erilaisessa tahdissa.

5.5 Aineiston analyysi

Miles, Huberman ja Saldaña (2014) määrittelevät aineiston analyysin kolmena samanaikaisena tehtävänä: aineiston tiivistämisenä, aineiston esittämisenä sekä johtopäätösten tekemisenä. Laadullisen aineiston analyysiä kuvaavassa kirjallisuudessa painotetaankin analyysin iteratiivista luonnetta, jolloin analyysiä tapahtuu jatkuvasti prosessin edetessä aina aineiston keräämisen aloittamisesta johtopäätöksien tekemiseen saakka (Strauss, 1987; Dey, 1989; Miles ym., 2014).

Tämän tutkielman osalta analyysi käynnistyi haastattelujen toteuttamisen ja niiden litteroinnin aikana, jolloin toistuvia teemoja havaittiin haastateltavien välillä. Teemoittelussa analysoitavasta aineistosta tunnistetaan yhteneväisiä teemoja tai piirteitä, jotka esiintyvät useamman haastateltavan vastauksissa (Hirsjärvi & Hurme, 2014). Teemoittelun katsottiin parhaiten auttavan tutkimuskysymykseen vastaamisessa, koska sen myötä voitiin aineistosta tunnistaa merkittävimmät teemat ja verrata niitä aiempaan kirjallisuuteen aiheesta.

Aineiston litteroinnin aikana sekä pian koko aineiston keräämisen jälkeen haastateltavien vastaukset koostettiin kunkin haastattelurungon pääteeman (taustatiedot, ketteryys, onnistumisen määrittely ja menestystekijät) alle. Aineistoa käytiin läpi, esiintyvistä teemoista tehtiin huomioita ja vastauksia värikoodattiin yhtenäisten teemojen osalta. Näin yhtenäiset teemat saatiin näkyväksi aineistosta ja niitä voitiin verrata toisiinsa sekä aiempaan kirjallisuuteen aiheesta.

5.6 Tutkimuksen luotettavuus

Tutkimuksen luotettavuutta arvioidaan usein validiteetin sekä reliabiliteetin näkökulmasta. Validiteetilla määritellään tulosten yleistä luotettavuutta: kuinka luotettavina tuloksia voidaan pitää ja miten tutkijan oman subjektiivisen mielipiteen vaikutusta tutkimustuloksiin pyritty minimoimaan (Runeson & Höst, 2009). Reliabiliteetti taas ottaa kantaa tuloksien toistettavuuteen ja siihen, voidaanko olettaa, että samasta aineistosta saataisiin samanlaiset tulokset, vaikka tutkijaa vaihdettaisiin (Runeson & Höst, 2009; Hirsjärvi & Hurme, 2014). Miles ym. (2014) viittaavat reliabiliteettiin tietynlaisena laadunvalvontana: ovatko tulokset johdonmukaisia, aikaa kestäviä sekä tutkijasta tai metodista riippumattomia.

Tämän tutkimuksen luotettavuutta on pyritty lisäämään läpinäkyvyyden avulla: kaikki tutkimuksen toteutukseen liittyvät valinnat sekä kirjallisuuskatsauksen että empiirisen tutkimuksen osalta on pyritty selostamaan ja perustelemaan huolellisesti, jolloin tutkimus on helposti toistettavissa, minkä voidaan ajatella omalta osaltaan lisäävän tutkimuksen luotettavuutta.

Haastattelujen luotettavuuteen vaikutettiin erityisesti kolmella tavalla. Aikaisemmissa tutkimuksissa esitellyt mahdolliset menestystekijät operationali-

sointiin osaksi haastattelurunkoa huolellisesti, kuten kerrottiin luvussa 5.3, millä pyrittiin varmistamaan, että haastateltavat ymmärsivät tekijät oikein ja puhuivat samoista menestystekijöistä kuin aikaisemmissakin tutkimuksissa. Ennen haastattelujen toteutusta mahdollisesti kontaktoitaviin yrityksiin tutustuttiin huolellisesti, millä pyrittiin varmistamaan, että haastateltavien edustamat yritykset edustaisivat mahdollisimman hyvin tutkimusongelman kannalta oleellista kohdejoukkoa. Haastattelutilanteet ja litterointi toteutettiin myös mahdollisimman vakioidusti: haastattelun tavoitteet ja tärkeimmät määritelmät käytiin läpi kaikkien haastateltavien kanssa huolellisesti, haastattelujen tallennukseen käytettiin tietokoneen lisäksi myös ulkoista tallenninta, jotta vältettiin ongelmat tallennuksessa ja kaikki haastattelut litteroitiin mahdollisimman pian haastattelujen jälkeen.

Myersin ja Newmanin (2007) mukaan laadullisiin haastatteluihin liittyy myös monia ongelmakohtia, jotka voivat osaltaan vaikuttaa tutkimuksen luotettavuuteen. Tämän tutkimuksen osalta mahdollinen ongelmakohta voi liittyä ymmäryksen rakentumiseen haastattelun aikana. Tällöin haastateltavat ovat saattaneet päästä pohtimaan haastattelun aihepiiriä yksityiskohtaisesti ensimmäistä kertaa vasta haastattelun aikana ja täten pyrkiä saamaan vastauksistaan ennemmin loogisia ja johdonmukaisia kuin peilaavan täysin reaalimaailmaa (Myers & Newman, 2007). Vaikka haastattelun aluksi ja sen aikana painotettiin, että haastateltava pyrkisi tunnistamaan ketterän ohjelmistokehityksen menestystekijöitä oman organisaationsa toiminnasta, on edellä kuvattu ilmiö voinut ohjata haastateltavia kertomaan yleisesti olettamistaan ketterän ohjelmistokehityksen menestystekijöistä.

Toisaalta luotettavuutta arvioitassa on huomioitava myös käsiteltävän aiheen liiketoiminnallinen arvo haastateltaville, minkä vuoksi on mahdollista, että haastateltavien vastauksia on saattanut rajoittaa ajatus siitä, että jotkin yrityksen ketterän ohjelmistokehityksen menestystekijät tuovat kilpailuetua muihin ohjelmistoyrityksiin nähden, eikä niitä ole sen takia haluttu jakaa. Voidaan kuitenkin melko todennäköisesti olettaa, että tämän tekijän merkitys tutkielman luotettavuudelle on vähäinen.

Eniten tutkimuksen luotettavuuteen ja erityisesti yleistettävyyteen vaikuttavana tekijänä voidaan nähdä tutkimuksen otanta: kymmenestä kontaktista saatiin lopulta kuusi toteutunutta haastattelua. Kuuden haastattelun pohjalta koottua aineistoa ei voitane pitää aukottomana, joten tutkimuksen ensisijaiseksi arvoksi muodostuu näiden haastateltavien henkilökohtaisten ajatusten ja kokemusten ymmärtäminen ja peilaaminen aikaisempaan tutkimukseen aiheesta. Lisäksi luotettavuuteen voi vaikuttaa se, että pro gradu –tutkielmalle luonteenomaisesti myös tämä tutkimus on toteutettu vain yhden tutkijan toimesta, jolloin oletattavaa on, ettei täydellistä objektiivisuutta ole välttämättä onnistuttu säilyttämään.

6 TULOKSET

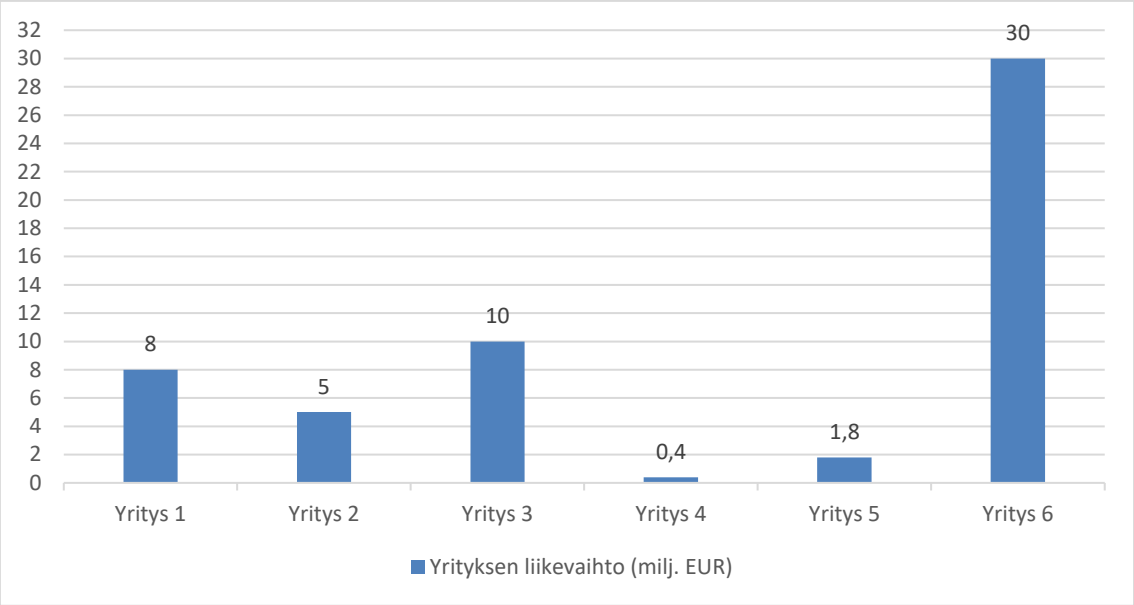
Tässä luvussa käydään läpi empiirisen tutkimuksen tulokset. Ensin esitellään haastateltavien taustatiedot. Tämän jälkeen avataan sitä, millä tavoin haastateltavien edustamat yritykset tekevät ketterää ohjelmistokehitystä ja millaisia hyötyjä he ovat kokeneet saaneensa siitä. Kolmanneksi selvitetään, miten onnistunut ohjelmistokehitysprojekti määritellään haastateltavien edustamissa yrityksissä ja kuinka onnistumista mitataan. Lopuksi raportoidaan haastateltavien esittämät ketterän ohjelmistokehityksen menestystekijät sekä ne tekijät, joilla ei katsota olevan vaikutusta ohjelmistokehitysprojektin onnistumiselle.

6.1 Haastateltavien taustatiedot

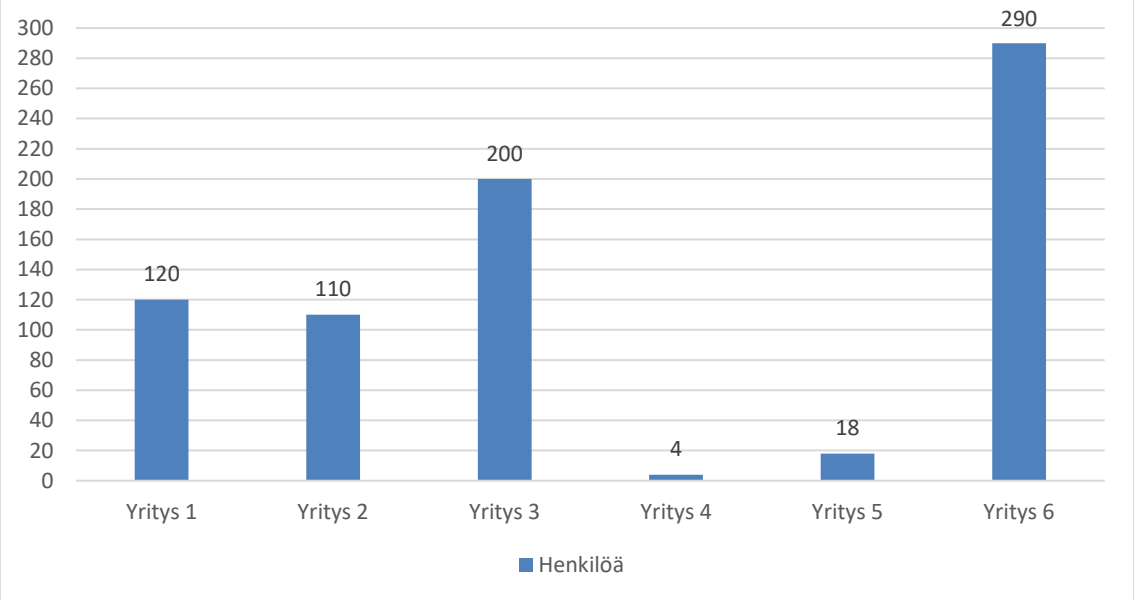
Tutkimusta varten haastateltiin yhteensä kuuden eri ohjelmistokehitysyrityksen edustajaa. Haastateltavilta kartoitettiin pääosin edustamansa yrityksen taustatietoja liikevaihdon, henkilöstömäärän, päämarkkinan, yleisimpien ohjelmistotyyppien sekä yhtäaikaisten ohjelmistokehitysprojektien osalta.

Yritykset edustivat eri liikevaihto- ja henkilöstöluokkia, joita on vertailtu kuvioissa 6 ja 7. Kuvioissa yritykset on numeroitu yhteneväisesti haastateltujen kanssa, jolloin Haastateltavan 1 edustama yritys on kuvioissa Yritys 1, Haastateltavan 2 edustama yritys on Yritys 2 ja niin edelleen. Liikevaihdon suhteen yritykset edustivat paikoin suurtakin hajontaa pienimmän liikevaihdon ollessa 0,4 miljoonaa euroa (Yritys 4) ja suurimman ollessa 30 miljoonaa euroa Yrityksellä 6 (kuvio 6).

Myös henkilöstömäärä erosi haastateltavien edustamien yrityksien välillä paikoin paljonkin. Kaksi yritystä työllisti alle 20 henkilöä (Yritys 4 ja Yritys 5), kolmen henkilöstömäärä oli välillä 100-200 henkilöä (Yritys 1, Yritys 2 ja Yritys 3) ja Yritys 6 työllisti 290 henkilöä (kuvio 7). Viisi yritystä edustaa siis Tilastokeskuksen (2017) määritelmän mukaan Pk-yritystä, yhden yrityksen ollessa Pk-yrityksen määritelmää suurempi henkilöstömäärän suhteen.



KUVIO 6 Yrityksien liikevaihto



KUVIO 7 Yrityksien työllistämien henkilöiden määrä

Yritystä 6 lukuun ottamatta kaikki yritykset mainitsivat päämarkkinakseen yrityksien välisen liiketoiminnan. Yritykset siis toteuttavat ohjelmistoprojekteja, joissa asiakas on toinen yritys, mutta loppukäyttäjä saattaa olla myös kuluttaja.

> Mehän ollaan projektitalo ja tehdään asiakkaille projekteja, että meillä ei oo omia tuotteita ja yleensä suurin osa projekteista on B2B-puolta. Mutta siellä on myös projekteja, jotka suuntaa taas sinne meidän asiakkaan loppuasiakkaille, jotka saattaa myös olla kuluttajia. (Haastateltava 2)

> Pääasiassa B2B, eli yrityspuolella. Mutta ne ite sovellukset sinänsä saattaa olla loppukäyttäjä kuluttajille, mut meidän asiakas on sitten usein yritys, joka tarjoaa sitten niitä palveluita eteenpäin. (Haastateltava 3)

Kysyttäessä yrityksien pääasiallisesti kehittämiä ohjelmistoja, vain yksi haastateltava kertoi yrityksensä keskittyvän web-sovelluksiin. Kaikki muutkin nimesivät web-sovellukset yhtenä ohjelmistotyyppinä, mutta lisäksi useampi haastateltava mainitsi mobiilisovellukset sekä sulautetut järjestelmät ohjelmistotyyppeinä.

6.2 Haastateltavien edustamien yrityksien ketteryys

Seuraavassa haastatteluosiossa kartoitettiin haastateltavien näkemyksiä oman yrityksensä ketteryydestä sekä siitä, millaisia hyötyjä he ovat ketteryydestä mielestään saaneet.

Haastateltavien nimeämät metodit ja tavat tehdä ketterää ohjelmistokehitystä olivat hyvin linjassa aikaisemipien tutkimuksien kanssa käytetyimmistä ketteristä menetelmistä (Rodríguez ym., 2012; VersionOne, 2015). Varsinaisista menetelmistä Scrum esiintyi kaikkien haastateltavien vastauksissa, mutta lähes jokainen haastateltava mainitsi, ettei hyödynnä menetelmää sellaisenaan, vaan on ottanut käyttöön tiettyjä käytänteitä Scrumista tai käyttävänsä myös erilaisia hybridi-menetelmiä. Syitä tähän olivat yrityksen omiin toimintamalleihin liittyvät tekijät sekä erityisesti pyrkimys sopeuttaa ohjelmistokehitys paremmin asiakasyrityksen kulttuuriin ja toimintamalleihin sopivaksi.

> Meillä hieman johtuen liiketoiminnan luonteesta niin käytännöt vaihtelee ehkä hieman projekteittain, mut pääsääntösesti meillä on valtaosassa projekteista Scrum käytössä. (Haastateltava 1)

> Että meillä on ne ketterän kehityksen periaatteet, mutta ei mennä ihan puhdasta Scrumia, vaan ollaan ehkä vähän soviteltu siitä meidän näkönen ja toimiva - meillä toimiva - ratkasumalli tällä hetkellä käyttöön. (Haastateltava 2)

> Sit se, että mitä käytänteitä käytetään, niin se tulee aina sen kontekstin mukaan. Eli sen mukaan, että mitä siellä asiakaspäässä on ihmisiä, ketä siihen tiimiin kuuluu, mikä sitten sen asiakasfirman kulttuuri on. Mut kyllä aika usein tämmöset peruskäytännöt kuten Daily Scrumit, retrospektiivit, sprintin suunnittelut, sprintit. (Haastateltava 4)

> Jos me sanottais, et me tehään tätä Scrumia by the book, niin todennäköisesti hommat ei toimis kummankaan [asiakasyrityksen] kanssa vaan yritetään sovittaa siihen, et minkälaisia rooleja ja minkälaista osaamista asiakkaalta löytyy ja sit sopeutetaan oma toimintamme siihen. (Haastateltava 6)

Suurin osa haastateltavien edustamista yrityksistä on tehnyt ohjelmistokehitystä ketterästi perustamisestaan saakka: viisi haastateltavaa

kertoi, että edustamansa yritys on aina ollut ketterä. Yksi haastateltavista oli työskennellyt edustamassaan yrityksessä vain lyhyen aikaa, eikä osannut arvioida, kuinka kauan ketterää ohjelmistokehitystä oltiin tehty.

Ketterän ohjelmistokehityksen hyötyjä kartoitettaessa tärkeimmäksi teemaksi nousi parempi asiakkaan osallistuminen. Tämä näkyy sekä paremmin toteutuvana kommunikaationa että syvempänä yhteistyönä ohjelmistokehitysprojektissa. Käytännön tasolla kommunikaation puute nähtiin isona ongelmana ohjelmistokehityksessä ja tähän ketteryyden koettiin tuovan apua.

> Jos aattelee omaa kokemusta, ni miks asiat menee päin mäkeä, niin kommunikaation puute on yleensä miks ne alkaa menemään päin mäkeä… Tai hyvässä Scrum-projektissa ni se tapahtuu päivittäin tai vähintäänkin siellä viikkotasolla, riippuen nyt vähän sitten sprintin pituudesta. (Haastateltava 1)

> Kommunikaatio asiakkaan kanssa on paljon välittömämpää ja suorempaa ja päästään yhteistyössä paljon syvemmälle kuin mitä tämmösessä muussa perinteisessä toimintatavassa on millonkaan mahdollista. (Haastateltava 3)

Myös aktiivinen yhteistyö ja kumppanuuden merkitys nousivat esille. Se, ettei ohjelmistokehitysprojekti näyttäydy asiakkaalle projektina, joka ulkoistetaan ja unohdetaan, vaan yhteisenä projektina, johon asiakas aktiivisesti osallistuu, koettiin tärkeäksi. Ketterän ohjelmistokehityksen ajateltiin paremmin mahdollistavan asiakkaan sitoutumisen projektiin sekä yrityksen sitoutumisen asiakkaan liiketoimintaan. Ketteryyden katsottiin myös lisäävän luottamusta osapuolten välillä, kun säännöllisen kommunikaation ja julkaisun avulla yritys voi osoittaa, että se pystyy pitämään sovitusta kiinni.

> Että tavallaan sillä me saadaan jossakin kohti vähän enemmän, jossakin vähän vähemmän, jossakin vähän väkipakolla osallistettua – tai asiakas osallistumaan – siihen työhön, et se ei oo tavallaan heidänkään päässä mikään semmonen ”fire and forget”. Eli tavallaan saadaan asiakas kiinteästi osana sitä ohjelmistokehitysprojektia. Et se kommunikaatiolinkki ei katoa mihinkään. Ja myöskin asiakas ymmärtää oman roolinsa siinä ja sen tärkeyden koko projektin aikana. (Haastateltava 1)

> Että meillä on pidempi sitoutuminen siihen projektiin asiakkaan näkökulmasta kuin vaan se, että me oltais siellä ja tehdään ohjelmistotyöt ja pakataan kamppeet ja lähdetään pois. Ni siinä mielessä tää ketterä kehittäminen toimii paljon paremmin, koska se helpottaa sen kumppanuuden luomista sinne asiakkaan suuntaan. (Haastateltava 2)

> Niin kun säännöllisesti tavataan ja nähdään, että sieltä tulee koko ajan uutta toiminnallisuutta ja pidetään pienissä pätkissä kiinni siitä mitä sovitaan. Pikkuhiljaa syntyy semmonen aika syvä luottamus, jolloin tavallaan ei oo lähtökohtana koko ajan se, et pitää sopimuspykälillä ja tämmösillä yrittää pitää kiinni siitä, että toinen tekee sen mitä on sovittu, vaan voidaan aidosti siihen luottaa, ja sopiminen ja tämmönen muuttuu paljon kevyemmäks. (Haastateltava 6)

Palaute ja mahdollisuus suunnan muuttamiseen palautteen pohjalta nähtiin toisena tärkeänä ketterän ohjelmistokehityksen hyötynä. Ketterään

ohjelmistokehitykseen kuuluva palautesykli – niin asiakkaalta tuleva kuin myös kehitystiimin sisäinen – auttaa löytämään mahdollisia ongelmakohtia, helpottaa riskien hallintaa ja mahdollistaa kehitystiimin tehokkaamman työskentelyn. Ketterälle ohjelmistokehitykselle ominaista on, että toimivaa ohjelmistoa pyritään toimittamaan mahdollisimman varhaisessa vaiheessa käyttäjille, jolloin palautetta voidaan hyödyntää kehitysprojektin seuraavien tehtävien suunnittelussa ja priorisoinnissa. Näin voidaan pyrkiä varmistamaan, että kehitettävä ratkaisu vastaa mahdollisimman hyvin käyttäjien tarpeeseen.

> No ihan suoraan vois sanoo, et läpinäkyvyyden kautta, mitä saadaan niistä projekteista ja projektin etenemisestä, ni saadaan löydettyä niitä ongelmakohtia, mitkä vaatis vielä jatkokehitystä. (Haastateltava 2)

> No jos mä vertaan siihen, et jos me tehtäis jollain perinteisellä tavalla, niin kyllähän siinä nopeemmin saa palautetta siitä, että mikä toimii. Parempi riskien hallinta, että pystytään oikeesti, empiirisesti kattomaan, että toteutuuko jotkut esimerkiks tekniset riskit kokeilemalla niitä asioita ensin, jotka on teknisesti haastavia. Sit on paljon suurempi mahollisuus – tai on ylipäänsä mahollisuus – muuttaa suunnitelmia, jos huomataan jotain yllättävää. (Haastateltava 4)

> Ketteryydessä yritetään mahollisimman pienellä työllä päästä heti käyttäjille viemään se tuote, et päästäis käyttäjien palautteen perusteella kehittämään sitä eteenpäin ja tekemään oikeita asioita sinne. (Haastateltava 6)

Haastateltavien edustamat yritykset toteuttavat ketterää ohjelmistokehitystä hyvin samantapaisesti kuin aiemmissa Rodríguezin ym. (2012) sekä VersionOnen (2015) toteuttamissa tutkimuksissa raportoidut yritykset. Scrum on suosituin varsinainen menetelmä, mutta myös erilaisia hybridimalleja käytetään paljon. Tärkeimmiksi ketterän ohjelmistokehityksen hyödyiksi nimettiin aktiivinen asiakkaan osallistuminen ohjelmistokehitysprojektiin, parempi kommunikaatio kehitystiimin ja asiakkaan välillä sekä välittömämpi palaute ja mahdollisuus suunnan muuttamiseen sen perusteella.

6.3 Onnistuneen ohjelmistokehitysprojektin määrittely ja mittaaminen

Seuraavassa osiossa pyrittiin selvittämään, miten onnistunut ohjelmistokehitysprojekti määritellään haastateltavan edustamassa yrityksessä, millä tavoin onnistumista mitataan ja mitä mittareita yritykset hyödyntävät.

Useimmat haastateltavat kertoivat, että ohjelmistokehitysprojektin onnistuminen määritellään yrityksessä projektikohtaisesti. Yleisimmin haastateltavat luonnehtivat onnistumista liittyvän liiketoiminnallisten tavoitteiden täyttymiseen, asiakastyytyväisyyteen sekä tyytyväisiin loppukäyttäjiin. Liiketoiminnallisten tavoitteiden täyttyminen nähtiin pääsääntöisesti liikevaihdon tavoiteltuna kasvuna – kehitettävän ohjelmiston tuottaessa asiakasyritykselle uutta kassavir-

taa, projekti voidaan määritellä onnistuneeksi. Toisaalta liiketaloudelliset tavoitteet voivat olla myös kulusäästöjä, jolloin niiden toteutuminen tekee ohjelmistokehitysprojektista onnistuneen.

> Eli yks on tietysti se, että onnistutaan tuottamaan ohjelmisto, joka joko tuottaa uutta liiketoimintaa, eli uutta kassavirtaa firmalle, sillä tavalla että se täyttää loppuasiakkaan tarpeita. Tai sitten tuottaa säästöjä firmalle siinä mielessä, että pystytään esimerkiks automatisoimaan jotain ja muut kulut vähenee siitä, että käytetään sitä ohjelmistoa. (Haastateltava 4)

> Esimerkiks jos nyt puhutaan vaikka jostain webbi-kauppa -tyyppisestä projektista, niin sillä luultavasti on jotku liiketoiminnalliset tavoitteet, että kuinka paljon se kasvattaa myyntiä. Ni se määritellään se onnistuminen sen mukaan, että se on onnistunu kasvattaa myyntiä näiden tavoitteiden mukaan tai mielellään jopa enemmän. (Haastateltava 6)

Yksi haastateltavista korosti liiketoiminnallisten tavoitteiden tilannesidonnaisuutta, jolloin yksittäisen ohjelmistokehitysprojektin liiketoiminnallinen onnistuminen voidaan todentaa myös muutoin kuin liikevaihdon kasvuna.

> …joskus onnistuminen voi olla se, et saadaan vaikka rahoitusta seuraavaan vaiheeseen. Määritellään, että tän prototyypin tavoitteena on esitellä meidän ideaa potentiaalisille rahoittajille ja tavoitteena on saada rahoitusta. Tai jossain voi olla tavoitteena saada pilottiasiakkaita esimerkiks. (Haastateltava 5)

Liiketoiminnallisten tavoitteiden rinnalla myös asiakastyytyväisyys koettiin tärkeänä onnistumista määrittelevänä tekijänä. Asiakastyytyväisyyttä kuvattiin sekä asiakasyrityksen tyytyväisyytenä kehitettyyn ohjelmistoon ja kehitysprojektiin sekä loppukäyttäjien tyytyväisyytenä kehitettyyn ohjelmistoon.

> Totta kai meillä on myöskin liiketaloudelliset tavoitteet aina projekteille, et mihin pitäis päästä, mutta tärkeintä on se, että asiakas kokee saavansa sen mitä se tarttee ja on tyytyväinen, ku projekti päättyy. (Haastateltava 1)

> Et eli se onnistunut projekti on semmonen, jonka lopputulos palvelee käyttäjiä mahdollisimman hyvin, käyttäjät tykkää käyttää sitä ja ne käyttäjät, joille se on tehty, niin myöskin tulee käyttämään sitä. (Haastateltava 6)

Eräs haastateltava mainitsi määrittelevänsä projektin onnistuneeksi myös, jos pystytään epäonnistumaan ketterästi. Tällöin ketterästi kehitetyn ohjelmiston avulla voidaan oppia lisää oletetuista liiketoimintamahdollisuuksista ja hyödyntää uutta tietoa ennen lopullista päätöstä ja mahdollisesti välttää suuri taloudellinen riski.

> Eli jos pystytään pienellä budjetilla, tekemällä joku softa, oppimaan, että tää meidän ajatus siitä, että tällä softalla sais uutta liiketoimintaa tai tällä softalla saatettas saada kulusäästöjä, ja pystytään osoittamaan se oletus vääräks... Eli sen sijaan, että siihen

> käytetään 200 000 tai miljoona siihen ohjelmiston tekemiseen, ni pystytään tekemään pieni proto kymppitonnilla ja osoittamaan se, että ei tässä ollukaan sitä liiketoimintamahdollisuutta, ni se on mun mielestä kanssa onnistuminen. (Haastateltava 4)

Toisaalta erään haastateltavan mukaan projektia voidaan pitää onnistuneena myös, jos se tarjoaa mahdollisuuden yrityksen sisäiselle oppimiselle. Tällöin yrityksen omista liiketaloudellisista tavoitteista voidaan tinkiä, jos projekti auttaa oppimaan asiakkaan liiketoiminnasta tai se on keino opettaa kehittäjille uutta.

> Eli sen ohjelmistoprojektin sisällä opitaan enemmän siitä asiakkaan kokonaisliiketoiminnasta ja toimintaympäristöstä, jolloin me voidaan tehdä käytännössä lähempänä tuloksellisesti nollaprojektia - tai jopa vähän miinuksella - jos me taas nähdään, et siinä on kokonaisuutena potentiaalia kasvaa… ja toinen on sitte, että onko siinä vaikka uusia kehittäjiä mukana, jolloin siitä tulee taas kerrointa meille mukaan, että ”Okei, tää voi olla projekti, missä me opetetaan meiän kehittäjillemme jotain uutta teknologiaa tai muuta juttua” (Haastateltava 2)

Useimmissa yrityksissä projektin onnistumisen mittaaminen perustuu vuoropuheluun kehittäjien ja asiakkaiden välillä. Asiakastyytyväisyyden mittaamiseen haastateltavat eivät maininneet tiettyä systemaattista ja strukturoitua tapaa, vaan esimerkiksi Scrumin katsottiin tarjoavan hyvän mahdollisuuden asiakastyytyväisyyden jatkuvaan seurantaan sprintin katselmointien sekä projektin retrospektiivien myötä. Samanlaisia käytänteitä hyödynnettiin myös pidempiaikaisissa kumppanuuksissa esimerkiksi osana erilaisia ohjausryhmäkäytänteitä.

> …tokihan sitten kun projekti päättyy, niin pyritään meidän puolesta järjestämään tämmönen projektin retro, jossa saadaan sit annettuu palautetta puolin ja toisin, ja saadaan feedbackia asiakkaalta. (Haastateltava 1)

> Et aika semmosella mikrotasolla jokaisessa projektissa periaatteessa jokainen Sprint review on semmonen onnistumismittari, myöskin et siinä on tuoteomistaja kattoo, että mitä on saatu viimeisen kahen viikon aikana aikaseks ja peilataan sitä sitten… (Haastateltava 4)

Loppukäyttäjien tyytyväisyyden mittaamiseen haastateltavat hyödyntävät muun muassa käyttäjätutkimuksia sekä -haastatteluja kehitysvaiheen aikana. Pilottivaiheessa saatu palaute nähtiin hyvänä mittarina loppukäyttäjien tyytyväisyyden arvioinnissa ja palautteen pohjalta pystytään tekemään korjaavia toimenpiteitä ennen sovelluksen varsinaista julkaisua. Tutkimuksien ja haastattelujen lisäksi loppukäyttäjien tyytyväisyyttä voidaan pyrkiä mittaamaan myös web-analytiikan avulla: kun kehitettävä ohjelmisto saadaan loppukäyttäjille jo aikaisessa vaiheessa, voidaan käyttötietoja mitata, ja arvioida onnistumista tämän pohjalta.

Yhteenvetona voidaan todeta, että tärkeimmät onnistumista määrittelevät tekijät ovat haastateltavien mielestä liiketoiminnallisten tavoitteiden täyttyminen sekä tyytyväiset asiakkaat ja loppukäyttäjät. Haastateltavien määritelmät

onnistuneesta ohjelmistokehitysprojektista ovat siten lähempänä Misran ym. (2009) määritelmää kuin Chow'n ja Caon (2008), joiden määritelmä onnistuneesta projektista mukailee tunnettua "Iron-Triangle"-määritelmää: laatu, aika, kustannus sekä laajuus (Atkinson, 1999). Pääsääntöisesti ketterän ohjelmistokehitysprojektin katsottiin siis onnistuneen, kun se tuottaa asiakasyritykselle liiketoiminnallista arvoa – joko uutena liikevaihtona tai säästettyinä kuluina. Onnistumisen mittaus haastateltavien edustamissa yrityksissä perustuu vuoropuheluun asiakkaan kanssa, erilaisiin käyttäjätutkimuksiin sekä analytiikkaan. Koska tavoitteet asetetaan projektikohtaisesti, ei yrityksissä ole systemaattista ja toistuvaa tapaa mitata onnistumista, vaan eri tavoitteiden toteutumista mitataan eri tavoin.

6.4 Haastateltavien esittämät ketterän ohjelmistokehityksen menestystekijät

Ketterän ohjelmistokehityksen menestystekijöitä kartoitettiin haastateltavilta sekä avoimilla kysymyksillä että pyytämällä haastateltavia peilaamaan aiemmassa kirjallisuudessa testattuja menestystekijöitä oman yrityksensä toimintaan (Liite 1 ja Liite 2). Haastateltaville kerrattiin kriittisen menestystekijän määritelmä (Bullen & Rockart, 1981) ja painotettiin, että haastateltavat nimeäisivät vain oman yrityksensä toiminnassa tunnistamiaan tekijöitä, jotta tuotoksena ei olisi vain lista asioista, joita on hyvä ottaa huomioon ketterässä ohjelmistokehityksessä.

6.4.1 Asiakasyhteistyö

Kaikki haastateltavat mainitsivat asiakasyhteistyöhön liittyviä tekijöitä ketterän ohjelmistokehityksen kriittisinä menestystekijöinä.

Yhteinen suunta ja ymmärrys tavoitteesta
Tärkeänä nähtiin, että kehitysprojektin alussa voidaan varmistua siitä, että asiakas ja toimittaja ajattelevat projektista samalla tavoin ja pitävät projektin tavoitetta yhteisenä. Näin asiakasyhteistyölle on oikeanlaiset edellytykset ja kehitystyön tärkeimpänä ohjausvoimana ei nähdä esimerkiksi sopimusta, jonka vaatimukset pyritään vain täyttämään.

> …yks kriittisimmistä menestystekijöistä on, et meillä tavalla tai toisella on mahdollisimman yhteneväinen tavoite ja insentiivit ja suunta… Et molemmat vetää sitä laivaa samaan suuntaan. (Haastateltava 5)

Usein yhteiseen suuntaan liittyviä ongelmakohtia pyritään selvittämään hyvin aikaisessa vaiheessa – tarvittaessa haastateltavat ovat valmiita vastaamaan yhteistyöehdotuksiin myös kielteisesti, jos yhteistä suunnasta ei ole takeita.

> …mutta me myös jollakin tavalla myöskin valikoidaan asiakkaita… Et jos me nähdään heti alussa, et tässä ei kumpikaan osapuoli pääse tavallaan voittajana pois ja varsinkaan molemmat, niin ei semmosiin ehon tahoin kannata myöskään ryhtyä. (Haastateltava 1)

> Ni sit sanotaan suoraan, että jos asiakas vaatii jotain sellasta mihin me ei voida sitoutua, et okei selvä homma, että nyt sun pitää mennä jollekin toiselle toimittajalle. Et me ei voida olla tässä mukana, että me nähdään, että tää homma ei voi onnistua. (Haastateltava 5)

Sen lisäksi, että asiakkaan ja toimittajan tulee sitoutua yhteiseen tavoitteeseen, se tulee myös ymmärtää samalla tavalla. Yhteinen ymmärrys kokonaisuudesta on tärkeää, sillä kun päätökset eivät tapahdu keskitetysti, vaan niitä tekevät kaikki tiimin jäsenet ja asiakkaan edustajat, tulee niiden takana olla ymmärrys siitä, mitä kehitettävällä ohjelmistolla tavoitellaan. Tällöin erilliset pienetkin päätökset vievät kokonaisuutta samaan suuntaan. Yhteistä ymmärrystä voidaan lisätä muun muassa mallintamalla ohjelmiston liiketoimintamallia Lean Canvasilla tai Business Model Canvasilla, jolloin malli ei tule annettuna, vaan se luodaan tiimin sisällä, mikä lisää ymmärrystä kokonaisuudesta.

Läpinäkyvyys

Läpinäkyvyys mainittiin keinona tiivistää asiakasyhteistyötä. Tällöin on oleellista, että asiakkaalla on mahdollisuus seurata kehitystyön etenemistä, tuotoksia ja haasteita reaaliaikaisesti, mikä mahdollistaa aktiivisen osallistumisen ja kommunikaation.

> …ja sillon ku se on myös läpinäkyvää asiakkaan suuntaan, että mitä se projektin alkuun määritelty backlog on ja nähdään, että miten se sprinteissä etenee, että kuinka paljon siitä tulee tehdyks, ni sehän antaa jo aika konkreettisesti läpinäkyvyyttä sinne asiakkaan suuntaan… Ni nehän ainakin helpottaa sitä asiakkaan suuntaan viestimistä ja sitä asiakastyytyväisyyden ylläpitämistä. (Haastateltava 2)

> Eli tosiaan se, että sinne sitten avataan näihin kehitysympäristöihin asiakkaille pääsy ja asiakas näkee sen päivittäisen tekemisen ja pystyy antaa palautetta siitä mahdollisimman helposti… (Haastateltava 3)

Kehitysympäristön, dokumentaation, bugi-raporttien ja kehitystyöhön liittyvän viestinnän avaaminen asiakkaalle nimettiin konkreettisina toimina läpinäkyvyyden parantamiseen. Tärkeänä pidettiin myös, että tarvittaessa asiakasta ohjataan seuraamaan kehitystyön etenemistä – koska asiakkaiden arki on kiireistä, ei pelkkä mahdollistaminen aina riitä, vaan ”täytyy myöskin osittain vähän syöttää ja pitää huolta siitä, että asiakas todella sitten seuraa, että mitä on tehty ja miten se toimii (Haastateltava 3)”.

Asiakkaan osallistuminen

Lopulta asiakasyhteistyön onnistuminen on paljolti kiinni asiakkaan mahdollisuuksista ja halusta osallistua kehitystyöhön. Tämä tekijä nähtiin niin tärkeänä, että erityisesti tällöin toimittajayritys voi mahdollisuuksien mukaan

tinkiä omista toimintamalleistaan, jotta asiakkaan osallistumiseen löydetään paras mahdollinen malli.

> …eli siinä pitää aina lähtee siitä asiakkaan tilanteesta ja lähtee sitä rakentaa sitä hommaa. Mä nään, että tää on sellanen, että jos löydetään sellanen yhteinen kompromissi sieltä. Että mihin voidaan molemmat puolin ja toisin sitoutuu, et tää on se toimintamalli ja näin molemmat voi tässä vaik vähän lähentyä ja tää on se, miten yhdessä tehdään asioita, niin se on tosi tärkeetä. (Haastateltava 5)

Toisaalta haastateltavat kertoivat myös hyödyntävänsä tavallisimpia ketterän ohjelmistokehityksen konsepteja – Scrumin tuoteomistaja-rooleja tai XP:n käytännettä läsnä olevasta asiakkaasta – jolla pyritään varmistamaan, että asiakas aidosti osallistuu kehitystyöhön ja on tekemässä ohjelmiston kehittämiseen liittyviä päätöksiä yhdessä kehitystiimin kanssa.

> Ja sitten esimerkiks yks asiakkuus tällä hetkellä… ni he ovat aina alkuviikon täällä paikalla: maanantai, tiistai, keskiviikko. Heillä on kolme tai kaksi työntekijää täällä ja sitä kautta on oikeesti tosi aktiivista se osallistuminen. (Haastateltava 2)

> Tavallaan, että asiakas on osa meidän tiimiä tai että me ollaan osa asiakkaan tiimiä, jolloin siinä tiimissä, sitten kun siellä on se asiakkaan edustaja läsnä, niin on mahdollisuudet tehä ne päätökset. (Haasteltava 6)

Tarvittaessa asiakkaalle voidaan myös ehdottaa ulkopuolisen tuoteomistajan hankkimista, millä voidaan varmistaa, että tuoteomistajalla on aikaa kehitysprojektille.

> …jos se näyttää siltä, että asiakas ei pysty sitä [aikaa] antamaan, ni sillon täytyy käydä asiakkaan kans keskusteluja siitä, että otetaanko sit käyttöön joku ulkopuolinen tuotteenomistaja. Koska tavallaan muuten se projekti ei voi toimia, jos ei oo riittävästi tuotteenomistajalla aikaa siihen. (Haastateltava 3)

6.4.2 Kommunikointi

Toisena kriittisenä menestystekijänä useat haastateltavat mainitsivat kommunikaation. Tämä heijastuu haastateltavien vastauksissa myös ketterän ohjelmistokehityksen tuomista hyödyistä, jossa moni haastateltava nosti esiin juuri parantuneen kommunikaation.

Tehokas kommunikointi asiakkaan ja toimittajan välillä oli useamman haastateltavan mielestä kriittinen menestystekijä ketterässä ohjelmistokehityksessä. Jotta hyötyjä iteratiivisesta ja inkrementaalisesta kehittämisestä voidaan saada, tulee varmistaa, että kommunikaatio on aktiivista. Ohjelmistokehitystä ei voida kovinkaan hyvin suunnata iteraatioiden välillä, jos asiakkaalta saatu palaute jää vähäiseksi. Asiakaskommunikaation toivottiin olevan mahdollisimman välitöntä ja hyvänä keinona tämän edistämiseksi nähtiin melko yksinkertaisesti asiakkaaseen tutustuminen ja virallisen asiakas-toimittaja -suhteen madaltaminen.

> Sitten aika usein just tähän ketterään starttiin liittyen, niin järjestetään projektin alussa jotain tällästa ihan tiimille fasilitoitua tämmöstä tutustumishommaa. (Haastateltava 4)

> …et tutustutaan asiakkaaseen ihmisenä muutenkin, kun vaan työkuvioissa. Se helpottaa aika paljon sitä kommunikointia jatkossa, ku on vähän jutellu muistakin asioista ja ollu semmosessa rennommassa ilmapiirissä. Viettäny aikaa heidän kanssaan. (Haastateltava 6)

Kommunikaatiotavoissa haastateltavien vastaukset jakautuivat hieman. Osan mielestä varsinaista kommunikaatiota voi tapahtua vain kasvotusten, osan mielestä taas digitaaliset välineet soveltuvat viestintävälineiksi yhtä hyvin kuin kasvotusten samassa tilassa tapahtuva viestintä.

> No varsinaisesti mulla on semmonen päähänpinttymä tosta, et kommunikointia mun mielestä varsinaisesti on vaan sillon, ku ollaan fyysisesti läsnä. Että sitten kaikki muu on tiedonsiirtoa tavalla tai toisella… (Haastateltava 6)

Haasteltavien vastauksissa korostui kuitenkin se, että kasvotusten tapahtuvaa viestintää suosittiin parhaana vaihtoehtona kommunikoinnille, mutta tärkeintä oli viestinnän välittömyys. Tämän takia kaikki haastateltavat nimesivät erilaisia pikaviestityökaluja ja monien vastauksissa korostui, että näitä suosittiin perinteisen sähköpostiviestinnän sijaan. Viestintävälineistä Slack, Trello ja Jira mainittiin useimmin. Haasteltavat korostivat myös videoneuvotteluvälineiden merkitystä kommunikoinnissa. Näiden koettiin tuovan kommunikoinnin erittäin lähelle kasvotusten tapahtuvaa kommunikointia, minkä koettiin lisäävän paljon viestinnän välittömyyttä myös sellaisissa tilanteissa, joissa kommunikointi ei tapahdu fyysisesti samassa paikassa.

> …tän [kasvotusten tapahtuvan kommunikoinnin] merkitys on nykypäivänä kuitenkin vähenemässä eli on videoneuvotteluvälineitä, jotka toimii todella hyvin ja pystytään hoitaa näitä asioita käytännössä, vaikka ei fyysisesti ollakaan… Ollaan kyllä kasvotusten, mutta jonkun välineen kautta. (Haastateltava 3)

> …esimerkiks tuo Zoom.us on tosi hyvä työkalu siihen [videoneuvotteluun], ku näkee toisen naaman, niin kommunikaatio on ihan erilaista. Ja sitten toinen on nää pikaviestityökalut… Kommunikaatio on nopeempaa kuin sähköpostia pyöritellessä. (Haastateltava 4)

Eräs haastateltava korosti myös kommunikoinnin tärkeyttä tuotteen kehitysjonoa määriteltäessä ja esimerkiksi seuraavan sprintin suunnittelussa. Nämä ovat kriittisiä hetkiä, jolloin tulee pyrkiä varmistamaan, että kaikki ymmärtävät asiakkaan tarpeen samalla tavalla ja samassa laajuudessa.

> Jos ei siitä [kehitysjonon tehtävästä] keskustella, niin siitä tullee äkkiä, kun joku sanoo, että ”Hetkinen, ku mää oletin, että tämä tarkottaakin tätä.”. Näistä mulla on lukemattomia esimerkkejä sitten reaalielämästä, et oletetaan asioita. Että tämä kuvaakin tämän osan vain, eikä esimerkiks sitä laajempaa kokonaisuutta. (Haastateltava 1)

Tämän ratkaisuksi yrityksessä hyödynnetään ketterää käytännettä, jossa sprinttien välillä tiimi ja tuoteomistaja tarkentavat tuotteen kehitysjonoa. Näin ylätason käyttäjätarinat tai toiminnallisuudet voidaan ottaa tarkempaan käsittelyyn, kysyä tarkentavia kysymyksiä ja varmistua siitä, että kaikki ymmärtävät seuraavaan sprinttiin otettavat tehtävät samalla tavalla.

Samat tekijät korostuivat myös tiimin sisäisessä kommunikaatiossa. Osa haastateltavista ei erotellut vastauksissaan sisäistä kommunikaatiota ja asiakaskommunikaatiota, vaan kertoivat samojen tekijöiden, esimerkiksi kommunikaation välittömyyden, pätevän molemmissa tapauksissa. Eräs haastateltava mainitsi myös, että sisäisessä kommunikaatiossa korostuu psykologinen turvallisuus ja tiimin ymmärrys siitä, että kommunikoinnin tulee olla välitöntä, jotta yhteistyö onnistuu parhaalla tavalla.

> …ni se on just se, että kommunikaatio on silleen tasasta, että kaikki saa puheenvuoron. On sellanen turvallinen ympäristö, että pystytään väittelemään asioista avoimesti ja olemaan eri mieltä ja löytämään silleen paras vaihtoehto ilman, että siitä tulee semmoista egojen välistä konfliktia tai kukaan pahoittaa mielensä tai että mennään henkilökohtaisuuksiin… Koska sit jos on tavallaan semmonen uskomus, että yhteistyötaidot tai kommunikaatio ei oo tärkeetä, niin sitten ei oo halua myöskään kehittää niitä taitoja tai tapoja toimia yhessä ja se voi olla aika hankalakin sitte purkaa tää tilanne, koska ihmisillä uskomukset – tämmöset mindset-tyyppiset uskomukset – on aika syvällä. (Haastateltava 4)

6.4.3 Julkaisustrategia

Ketterä julkaisustrategia ja erityisesti usein tapahtuvien julkaisujen mahdollistama palautesykli loppuasiakkailta on monen haastateltavan mielestä kriittinen menestystekijä ketterässä ohjelmistokehityksessä. Se, että toimivaa ohjelmistoa saadaan mahdollisimman aikaisessa vaiheessa loppuasiakkaiden testattavaksi, nähtiin erittäin tärkeänä. Testauksen pohjalta saatavan tai analytiikan osoittaman palautteen hyödyntäminen ohjelmiston kehittämisen ohjausvoimana on ketterän ohjelmistokehityksen fundamentti, jonka oivaltamista pidettiin oleellisena.

> …eli siis ketterää kehitystähän voi tehdä sillain näennäisen ketterästi, että meillä on joku tämmönen tosi tarkkaan laadittu suunnitelma ja sit sitä lähetään toteuttaa vaikka nyt kahen viikon sprinteissä Scrumilla… Ja sitten ku ollaan toteutettu se suunnitelma, ni julkaistaan sovellus. Toihan on loppupeleissä pelkästään vesiputousmalli, joka on sitten jaksotettu tommoseen kahen viikon tekemiseen. (Haastateltava 6)

Mitä aikaisemmin kehitettävä ohjelmisto saadaan loppuasiakkaan testattavaksi ja kommentoitavaksi, sitä aiemmassa vaiheessa ohjelmistokehitystiimi pystyy arvioimaan esimerkiksi tärkeimpien toiminnallisuuksien suhteen tehtyjä valintoja. Tämän koettiin tuovan kaksi tärkeää hyötyä. Muutokset, joita palautteen pohjalta tehdään ohjelmistoon, ovat edullisempia ja nopeampia toteuttaa, jos niitä päästään tekemään mahdollisimman nopeasti ensimmäisien julkaisujen jälkeen. Tällöin ohjelmistoon ei ole vielä rakennettu yhtä paljon

ominaisuuksia ja riippuvuuksia eri ominaisuuksien välillä ja täten muutoksien tekeminen on helpompaa.

> Et jos saadaan palaute heti, ni on todella yleensä pieni työ muuttaa sitä. Mutta jos se saadaan se palaute niinku perinteisen vesiputousmallin mukaisesti vuoden päästä, kun siihen on rakennettu älyttömästi kaikkee ympärille, ni sit se muutos voi olla todella iso työ. Eli siinä mielessä se toimiva sovellus mahollisimman usein päivittyvänä on yks kriittinen tekijä. (Haastateltava 3)

Toisaalta loppukäyttäjiltä saatava palaute nähtiin kriittisenä myös ohjelmiston onnistumisen näkökulmasta: kun toimivaa ohjelmistoa julkaistaan suoraan loppuasiakkaille, edustaa ohjelmistosta saatava palaute – joko kommenttien, käytettävyystestauksen tai analytiikan avulla hankittuna – paremmin lopullista käyttäjäryhmää, eikä esimerkiksi vain tuoteomistajan mielipidettä. Näin saadaan myös parempi varmuus siitä, että tärkeimmät kehitettävät toiminnallisuudet ovat sellaisia, joita loppuasiakkaat todella tarvitsevat.

> Saatais semmonen sykli rakennettua, että se ei ois vaan yhden henkilön mielipide, että toimiiks tää vai ei. Vaan että ku tiimi esimerkiks rakentaa jonkun osan siitä softasta, ni se pystyttäis kokeilla kentällä. Ja sitten siitä saadaan palautetta, jonka perusteella pystyy oppimaan siitä, et okei mikä toimii, jotta se tiimi sitten pystyy iteroimaan sitä tuotetta kohti sitä onnistumista. (Haastateltava 4)

> …jos sanotaan nyt et tehään vaik 100 tonnin projekti, niin siinä 50 tonnin kohalla pitäis olla jo loppukäyttäjien käsissä se tuote. Ja sit sen loppubudjetti käytetään siinä siihen, että hiotaan sitä sen palautteen perusteella ja jatkokehitetään, jolloin saadaan varmasti tehtyä niitä oikeita toiminnallisuuksia, ku on sitä oikeeta käyttäjäpalautetta siellä jo. (Haastateltava 6)

Ketterää julkaisustrategiaa noudattaakseen useat haastateltavat kertoivat edustamansa yrityksen pyrkivän hyödyntämään jatkuvan integraation sekä testausautomaation käytänteitä mahdollisimman tehokkaasti. Jatkuvan integraation työkalujen ja käytänteiden ansiosta lähdekoodin muutokset voidaan julkaista napin painalluksella. Uuden ohjelmistoversion testit ajetaan automaattisesti, minkä jälkeen ohjelmisto voidaan julkaista suoraan asiakkaalle avoimeen kehitysympäristöön. Näin uutta ohjelmistoversiota julkaistaan jatkuvasti asiakkaalle ja uusista ohjelmistoversioista saadaan palautetta nopeassa syklissä. Tämä antaa mahdollisuuden julkaista nopeasti myös lisäarvoa tuovia toiminnallisuuksia, jotka voivat joissakin tapauksissa kasvattaa myös ohjelmiston rahallista arvoa – esimerkkinä tällaisesta eräs haastateltava käytti kehitettävää verkkokauppasovellusta, johon voidaan ketterän julkaisustrategian ansiosta julkaista nopeasti myös sellaisia ominaisuuksia, jotka lisäävät myyntiä ja siten myös ohjelmiston tuottoa.

6.4.4 Kompetenssi sekä harjoittelu ja oppiminen

Kompetenssi on oletettavasti menestystekijä monissakin yhteyksissä: voidaan yleisesti olettaa, että menestymisen mahdollisuudet ovat korkeammat, jos mukana on substanssiosaamiseltaan korkeatasoisia yksilöitä. Myös haastateltavat korostivat kompetenssin merkitystä, mutta moni ei pitänyt sitä aivan yhtä suoraviivaisena menestystekijänä kuin esimerkiksi Chow ja Cao (2008), joiden tuloksien mukaan ohjelmistokehitystiimin jäsenten tekninen kompetenssi ja asiantuntijuus ovat kriittinen menestystekijä ketterässä ohjelmistokehityksessä. Usean haastateltavan vastauksissa korostui myös tiimin dynamiikan merkitys. Tällöin on toki hyvä, jos tiimiin saadaan mukaan osaavia henkilöitä, mutta tärkeämpää on, että tiimissä on edustettuna oikeanlainen yhdistelmä ihmisiä. Näin kaikkien tiimin jäsenten ei tarvitse olla esimerkiksi teknisesti yhtä osaavia, vaan riittää, että tiimissä on joku, joka tuntee teknologian hyvin.

> Sehän [kompetenssi] on erittäin tärkeä tekijä, mutta lisäks yks tärkeä on myöskin sen tiimin dynamiikka. Että vaikka kaikki olis guru-, stara-koodareita, mutta jos ne ei osaa jutella keskenään, niin ei välttämättä sekään hyvä mixi oo. Että se kokemus ei oo aina kaikki kaikessa. (Haastateltava 1)

> …kompetenssi siltä osin, että kaikkien ei tarvitse olla kokeneita henkilöitä. Eli on paljon hyviä nuoria ihmisiä, jotka on todella motivoituneita oppimaan ja pistää itteänsä likoon ja tekee todella kovasti töitä siinä projektin aikana ja oppii asioita, kunhan siinä projektissa vaan sitten on joku, joka on riittävän kokenu ja osaava, jotta osaa toimia siinä tämmösenä teknisenä leadina tai mentorina… (Haastateltava 3)

Myös kehitystiimin jäsenten harjoittelua ja oppimista korostettiin menestystekijänä. Paikoin on jopa tärkeämpää, että yksilöillä on halua ja kyvykkyyttä opetella uutta kuin että yksilöt ovat kokeneita vaikkapa tietyssä ohjelmointikielessä. Esimerkiksi muutokset teknologioissa, toimintaympäristöissä ja asiakasorganisaatiossa edellyttävät nopeaa oppimiskykyä, minkä takia tämän merkitys nähtiin tärkeänä sekä aloittelevien että pidempään alalla olleiden työntekijöiden osalta.

> Ettei tavallaan ajatella ohjelmistokehitystä semmosena asiana, joka opitaan koulussa ja sit me ollaan ammattilaisia – me tiietään, mitä me tehään. Vaan se, että ajatellaan ohjelmistokehitystä semmosena, että aina kun tehään uutta ohjelmistoa, ni siinä on aika paljon semmosia asioita, joita kukaan ei voi tietää etukäteen, jolloin se aika iso osa siitä prosessista on oppimista. (Haastateltava 4)

Moni haastateltava mainitsi rekrytointien olevan tärkeässä roolissa kompetenssiin ja oppimiskykyyn liittyvissä tekijöissä. Rekrytoinnit ovat luonnollisesti mille tahansa yritykselle tärkeitä, mutta haastateltavien edustamien yrityksien tapauksessa, jossa liiketoiminta on projektimuotoista, eikä yrityksellä ole esimerkiksi omia tuotteita, on yrityksessä työskentelevien ihmisten rooli liiketoiminnan menestymisessä todella suuri. Keinoina rekrytointien parantamiseen mai-

nittiin toki muun muassa teknistä osaamista kartoittavat tehtävät, mutta eräs haastateltava korosti myös keskustelevan ilmapiirin tärkeyttä haastattelussa.

> …silloin ku se haastattelutilanne on enemmän keskusteleva kuin että joudutaan kaivamaan kaverista informaatiota, ni se keskiarvosesti on jo hyvä… Mut harvemmin semmosia, joista joudutaan pumppaamaan tietoa, niin palkataan, koska tämä taas linkittyy siihen aikasempaan ajatukseen siitä, että keskustelemalla asiat yleensä ratkeaa ja selviää ennemmin kuin myöhemmin. Niin jos et sä osaa keskustella, niin sit sä oot myöhemminkin hankaluuksissa. (Haastateltava 1)

Jotta yritykset todella hyötyvät ihmisten oppimiskyvystä ja -halukkuudesta, tulee yrityksien myös mahdollistaa osaamisen kehittäminen työn ohessa. Tähän liittyen haastateltavat mainitsivat erilaisia toimintamalleja. Erään haastateltavan edustamassa yrityksessä on varattu tietty aika sisäiselle tekniselle koulutukselle, jossa ihmiset pääsevät kokeilemaan muun muassa uusia teknologioita. Eräässä yrityksessä taas on käytäntö, jossa työntekijät saavat palkkaa myös vapaa-ajalla tekemästään kompetenssia kehittävistä toimista: tällöin työntekijä voi käyttää maksimissaan 10 tuntia kuukaudessa haluamansa asian opiskeluun, jonka katsoo kehittävän osaamistaan ja siten hyödyntävän myös yritystä. Myös mentorointi-ohjelmat mainittiin keinona kehittää osaamista erityisesti aloittelevien työntekijöiden keskuudessa.

> Meillä on tietyille työntekijöille - varsinkin junioripuolen työntekijöille - määritelty, että ketkä on sopivia mentoreita kenellekin ja sitä kautta vahvistettu tätä samaa [oppimista edistävää] kulttuuria. Sellasia toimenpiteitä, millä halutaan pitää näistä sekä kompetenssista että oppimishalukkuudesta jatkossakin kiinni. (Haastateltava 2)

6.4.5 Yrityskulttuuri ja päätöksenteon nopeus

Yrityskulttuurin merkitys on haastateltavien mukaan suuri ketterien ohjelmistokehitysprojektien onnistumiselle. Haastateltavien vastauksissa korostuivat erityisesti kaksi yrityskulttuuriin liittyvää tekijää: matala hierarkia ja itseohjautuvuus.

Haastateltavat kuvasivat edustamiensa yrityksien kulttuuria muun muassa ”kevyenä”, ”ketteränä”, ”matalana” sekä ”maanläheisenä”. Yhteistä näille eri määritelmille oli perimmäinen ajatus siitä, että ketterää ohjelmistokehitystä tekevässä yrityksessä tulisi vaalia kulttuuria, joka mahdollistaa henkilöstön sekä kehitystiimien autonomisuuden. Avoin ilmapiiri, jossa rohkaistaan keskusteluun sekä ongelmien esiin nostamiseen – myös eri projektien välillä – auttaa luomaa yrityskulttuuria, jossa työntekijöiden on helppo lähestyä toisiaan sekä organisaation johtoa, mikä osaltaan madaltaa hierarkiaa organisaation sisällä.

> …me kannustetaan ihmisiä myöskin ottamaan ja avaamaan suunsa, myöskin keskustelemaan, tuomaan ongelmia esille… Ja se on myöskin ihan tietonen, et ei olla lähetty kirjoittamaan kaikkia asioita auki, vaan pyritään kasvattamaan ja opettamaan ihmisiä siihen, että niitten tekemisellä on väliä. Koska sitä kautta ne oppii sitten seuraa-

> van kerran myöskin, että ”Hei, tässä oli haastava kohta, niin tämähän ratkes kun tehtiin näin”. (Haastateltava 1)

> …vaikka meillä on koko kasvanu, ni me ollaan pyritty pitämään siitä kiinni, että pidetään sellai samanlainen henki, mikä siinä pienemmässä yrityksessä on ollu aiemmin… että se ei lähe menee sellaseks isoks byrokraattiseks möykyks koko yritys, vaan pyritty pitämään suhteellisen välit helppoina ja vois sanoo, että kodikkaana tää paikka… (Haastateltava 2)

Itseohjautuvuus on toinen yrityskulttuuriin liittyvä osatekijä, joka mainittiin. Osa viittasi tähän itsenäisenä tiiminä, osa taas viittasi itseohjautuvuudella yleisemmin koko organisaation henkilöstöön. Oleellista ketterän ohjelmistokehityksen näkökulmasta on kuitenkin se, että ihmisillä on kyky ja halu toimia itseohjautuvasti työtehtävissään. Tällöin yksilöt sekä tiimit saavat vapauden toimia parhaaksi katsomallaan tavalla, ilman ylhäältä päin tulevaa ohjausta, esimerkiksi lähiesimiehen toimesta.

> …et jos se on tosi hierarkinen se organisaatio ja ei oo vapauksia tehä juuri mitään ja kaikki asiat pitää päättää jossain muualla, se on sit sellanen tekijä joka syö menestyksen. (Haastateltava 4)

Matalan hierarkian ja itseohjautuvuuden tuomaksi hyödyksi voidaan ajatella päätöksenteon nopeus, johon kaksi haastateltava viittasi samassa yhteydessä yrityskulttuurin kanssa. Kun organisaatio on hierarkialtaan matala ja ihmiset organisaatiossa toimivat itseohjautuvasti, edistää se päätöksenteon nopeutta. Itseohjautuvuuden ansiosta tiimillä on vapaus tehdä ohjelmistokehitysprojektia koskevia päätöksiä mahdollisimman itsenäisesti, eikä asioita tarvitse hyväksyttää korkeammalla organisaatiossa.

> Sen [päätöksenteon nopeuden] edellytys on se, että siinä tiimissä on ihmisiä, joilla on myös valtaa tehä ne päätökset, ettei jokaista asiaa tarvi kysyä jostain kiireiseltä johtoportaalta. (Haastateltava 6)

6.5 Haastateltavien esittämät tekijät, joilla ei ole vaikutusta ketterän ohjelmistokehitysprojektin menestykseen

Kriittisten menestystekijöiden lisäksi haastateltavilta kartoitettiin myös tekijöitä, joilla on vähäinen merkitys ketterän ohjelmistokehitysprojektin onnistumiselle. Tekijöitä kartoitettiin pyytämällä haastateltavia vertaamaan aikaisemmassa kirjallisuudessa esitettyjä tekijöitä (Liite 2) oman yrityksensä toimintaan. Aineiston analyysissä nousi kaksi teemaa, jotka esiintyivät haastateltavien vastauksissa.

6.5.1 Tiimin maantieteellinen jakauma

Vaikka haastateltavien vastaukset jakautuivat hieman sen suhteen, tulisiko kommunikoinnissa panostaa kasvotusten tapahtuvaan viestintään vai riittääkö erilaisten digitaalisten viestintävälineiden käyttö kommunikoinnin välittömyyden varmistamiseen, ei tuloksien perusteella näyttäsi olevan suurta merkitystä sillä, työskentelevätkö kehitystiimi ja muut sidosryhmät maantieteellisesti toistensa läheisyydessä. Hyvät etätyömahdollisuudet ovat vähentäneet tämän tekijän merkitystä ja toisaalta yleinen työkulttuuri tukee yhä enenevissä määrin etätyön tekemistä, minkä takia etänä tehtävän työn ei juurikaan ajatella hankaloittavan projektin kulkua sisäisesti tai asiakkaiden suuntaan.

Eniten on kyse aiemmin mainitun kommunikaation sujumisen varmistamisesta: kun työkalut ja toimintamallit mahdollistavat välittömän kommunikoinnin kehitystiimin, asiakkaan ja muiden sidosryhmien välillä, ei suuria ongelmia tulisi ilmetä. Eräs haastateltava nosti kuitenkin esiin mahdollisen erikoistapauksen maantieteellisestä jakaumasta, jossa tiimi työskentelee paljolti toisistaan eroavilla aikavyöhykkeillä. Tällaisessa tilanteessa viestinnän toimintamalleihin tulee kiinnittää erityistä huomiota, jotta kommunikaatiosta ei muodostu ongelmaa kehitystyölle.

6.5.2 Projektin luonne

Projektin luonteella on aikaisemmassa kirjallisuudessa viitattu sellaisiin ohjelmistoihin, jotka eivät ole luonteeltaan kriittisiä turvallisuuden näkökulmasta (kuten lentoliikenteessä käytettävät ohjelmistot). Perimmäisenä ajatuksena on, että ketterä ohjelmistokehitys ei soveltuisi tällaisten ohjelmistojen kehittämiseen onnistuneesti.

Haastateltavat eivät allekirjoittaneet tätä väitettä. Myös turvallisuuden näkökulmasta kriittisiä ohjelmistoja voidaan kehittää ketterän ohjelmistokehityksen keinoin, vaikkakin se vaatii todennäköisesti hieman totuttujen menetelmien mukauttamista.

> Mä en nyt niinkun aidosti nää yhtään syytä, miks ydinvoimalasoftaa tai esimerkiks tuossa lentoliikenteessä käytettävää ohjelmistoa ei voisi kehittää ketterästi… mut sillon esimerkis tietyt asiathan sun pitää analysoida paljon tarkemmin kuin ennen kuin sä alat esimerkiks kehittää asioita kuin kevyemmissä projekteissa. Että kaikkia asioita sä et voi taklata lennosta, vaan sulla pittää pystyä ottaan huomioon jo alussa vaikka tämmösiä reaaliaikavaatimuksia tai muita vastaavia tämmösiä, että miten ne vaikuttaa toisiinsa. (Haastateltava 1)

> …mun vahva mielipide on se, että nää [ketterä ohjelmistokehitys ja turvallisuuden kannalta kriittiset järjestelmät] ei oo toisiaan poissulkevia asioita… Jos miettii potilastietojärjestelmää tai lentoliikenteessä olevii ohjelmistoja, niin ei siitä kaikki asiat oo kriittisiä. Siitä on joku tietyt asiat on kriittisii, vaikka se miten niitä tietoja tallennetaan tai miten se dataa käsitellään – missä se data asuu. (Haastateltava 5)

Jos projektissa huomioidaan projektin luonteen tuomat rajoitteet sekä erikoistekijät ja analysoidaan sellaiset projektin vaiheet, jossa tarvitaan erityistä huolellisuutta esimerkiksi määrittelyn suhteen, voidaan ketterää ohjelmistokehitystä haastateltavien mukaan hyödyntää onnistuneesti siinä missä perinteisiäkin ohjelmistokehityksen menetelmiäkin.

7 POHDINTA

Tämän tutkielman tavoitteena oli valottaa ketterän ohjelmistokehityksen menestystekijöitä, eli sellaisia liiketoiminnan avaintekijöitä, joiden tulee sisältyä organisaation toimintaan, jotta ketterä ohjelmistokehitysprojekti olisi onnistunut (Chow & Cao, 2008). Tutkimuskysymys, johon empiirisellä tutkimuksella etsittiin vastausta, oli tässä tutkielmassa:

> Mitkä ovat ketterän ohjelmistokehityksen kriittisiä menestystekijöitä?

Empiirisen tutkimuksen tulosten pohjalta tunnistettiin seitsemän menestystekijää: asiakasyhteistyö, kommunikointi, julkaisustrategia, kompetenssi, harjoittelu ja oppiminen, yrityskulttuuri sekä päätöksenteon nopeus. Aiempien tutkimuksien ollessa tuloksiltaan ristiriidassa, myös tämän tutkimuksen tulokset vahvistavat joitain ja haastavat joitain aikaisempia tuloksia. Seuraavassa on vertailtu tämän tutkimuksen tuloksia aikaisempiin tutkimuksiin ja pohdittu tutkimustuloksien merkitystä teorialle sekä käytännölle.

7.1 Tuloksien vertailu aikaisempiin tutkimustuloksiin

Asiakasyhteistyö koettiin tutkimuksen haastateltavien mielestä tärkeänä ketterän ohjelmistokehityksen menestystekijänä. Tulos vahvistaa Chow'n ja Caon (2008), Misran ym. (2009) sekä Françan ym. (2010) tutkimuksien tuloksia, joissa todettiin sekä asiakkaan osallistumisen että asiakasyhteistyön olevan ketterän ohjelmistokehityksen menestystekijöitä. Tässä tutkielmassa asiakasyhteistyön alle sisällytettiin haastateltavien vastauksissa esiintyneet yhteinen suunta ja ymmärrys tavoitteesta, läpinäkyvyys sekä asiakkaan osallistuminen tuloksien tulkitsemisen selkeyttämiseksi. Asiakasyhteistyön merkitys ketterän ohjelmistokehityksen menestykselle on helppo hyväksyä, sillä asiakkaan merkitys korostuu kaikkialla ketterässä ohjelmistokehityksessä – jo Agile Manifeston (2001) arvoissa arvostetaan ”Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja”.

Kommunikoinnin tärkeys korostui haastateltavien vastauksissa. Tämä on ristiriidassa Misran ym. (2009) tutkimuksen tuloksien kanssa, joiden mukaan kommunikaatio ja neuvottelu eivät olisi menestystekijä ketterässä ohjelmistokehityksessä. Sama pätee myös Chow'n ja Caon (2008) sekä Stankovicin ym. (2013) tutkimustuloksiin verratessa. Vaikka he eivät tutkineet kommunikoinnin merkitystä erillisenä tekijänä, molempien tutkimuksien tuloksissa todettiin organisatorisen ympäristön – jota määriteltiin muun muassa attribuutilla "Keskusteleva kulttuuri, jossa arvostetaan kasvokkain tapahtuvaa kommunikointia" (Chow & Cao, 2008, s. 963) – olevan merkityksetön tekijä ketterien ohjelmistokehitysprojektien onnistumisessa. Sama tulos on todettavissa myös Françan ym. (2010) tutkimuksessa. Tämä on mielenkiintoinen ristiriita. On intuitiivista olettaa, että ohjelmistokehitysprojekteissa, joissa usein toimitaan muuttuvassa liiketoimintaympäristössä, vuorovaikutetaan monien eri sidosryhmien kanssa ja hyödynnetään monimutkaisia teknologioita, olisi onnistumisen näkökulmasta kriittistä, että sisäinen ja ulkoinen kommunikointi on mahdollisimman välitöntä ja tehokasta. Voi olla, että aikaisempien tutkimuksien vastaajien mielestä ainakin ulkoisen kommunikoinnin on katsottu sisältyvän jo asiakasyhteistyöhön ja asiakkaan osallistumiseen, mikä selittäisi sen, miksi aikaisemmat tutkimukset eivät vahvista tämän tutkimuksen havaintoa. Sisäisen kommunikaation taas on voitu katsoa liittyvän tiimityötä määrittäviin menestystekijöihin. Tällaisten epäselvyyksien välttämiseksi kommunikoinnin merkitystä tulisikin tutkia omana tekijänään.

Tämä tutkimus vahvistaa Chow'n ja Caon (2008) sekä Françan ym. (2010) tutkimuksien tuloksen, jonka mukaan julkaisustrategia on ketterän ohjelmistokehityksen kriittinen menestystekijä. Julkaisujen tiheyttä ja toimivan ohjelmistoversion toimittamista mahdollisimman aikaisin loppukäyttäjien testattavaksi sekä ennen kaikkea palautteen hyödyntämistä kehitystyön ohjauksessa voidaan pitää ketterän ohjelmistokehityksen yhtenä tärkeimmistä fundamenteista – organisaatio voi toimia näennäisen ketterästi noudattamalla esimerkiksi Scrumissa määriteltyjä rooleja ja tapahtumia, mutta jollei toimivaa ohjelmistoa kehitetä iteratiivisesti ja tiheiden julkaisujen myötä saatavaa palautetta hyödynnetä aidosti kehitystyön ohjaamiseen, ei yksi oleellisimmista ketteryyttä määrittelevistä tekijöistä – muutokseen vastaaminen – toteudu.

Kompetenssi sekä harjoittelu ja oppiminen korostuivat haastateltavien vastauksissa ketterän ohjelmistokehityksen menestystekijöinä. Samoja havaintoja on tehty myös aiemmissa tutkimuksissa muun muassa tiimin kyvykkyyden, harjoittelun ja oppimisen sekä henkilökohtaisten ominaisuuksien muodossa (Chow & Cao, 2008; Misra ym., 2009; França ym., 2010). Haastateltavien vastauksissa painottui se, että vaikka korkeasta kompetenssista on hyötyä, paikoin suurempi merkitys voi olla tiimin dynamiikalla sekä yksilöiden halulla ja kyvykkyydellä oppia uutta. Vuorovaikutuskykyä arvostetaan tiimityöskentelyssä enemmän kuin guru-statusta ja muutokset teknologioissa, toimintaympäristöissä ja asiakasorganisaatiossa edellyttävät nopeaa oppimiskykyä, minkä vuoksi yksilön halu kehittää omaa substanssiosaamistaan korostuu.

Yrityskulttuurin vaikutus ketterien ohjelmistokehitysprojektien menestymiseen on myös helppo hyväksyä, varsinkin koska kaksi yrityskulttuurin osatekijää korostuivat vastauksissa: matala hierarkia ja itseohjautuvuus. Voidaan olettaa, että yrityskulttuuri luo edellytykset ketterälle ohjelmistokehitykselle: jos yrityksessä vallitsee byrokraattinen johtamisrakenne, jossa yksilöille ei anneta vapautta tehdä päätöksiä, eivät lähtökohdat ole ketterälle ohjelmistokehitykselle otolliset. Kuten Abbas ym. (2008) toteavat ketterän ohjelmistokehityksen määrittelyssään, on yksi ketterän ohjelmistokehityksen tehtävistä tukea kehitystiimiä määrittämään paras tapa tehdä työnsä, mikä vaatii myös tuen yrityksen kulttuurilta. Yrityskulttuurin voidaan näin olettaa vaikuttavan vahvasti myös muihin ketterän ohjelmistokehityksen menestystekijöihin, minkä ovat todenneet myös Misra ym. (2009) omassa tutkimuksessaan. Chow'n ja Caon (2008), Françan ym. (2010) tai Stankovicin ym. (2013) tulokset eivät vahvista samaa näkemystä yrityskulttuurin merkityksestä ketterän ohjelmistokehityksen menestystekijänä. Toisaalta kuten luvussa 4.3 todettiin, on kulttuurin merkityksen arvioiminen hankalaa, koska eri tutkimuksissa samoja attribuutteja on katsottu kuuluvaksi eri tekijöiden alle. Chow'n ja Caon (2008) sekä Françan ym. (2010) tutkimustuloksien mukaan organisatorinen ympäristö, joka mittaa yrityskulttuurin vaikutusta, ei ole ketterän ohjelmistokehityksen menetystekijä, mutta tiimin kyvykkyys ja tiimiympäristö, joiden voidaan katsoa määrittävän itseohjautuvuutta ja johtamisrakennetta, taas ovat.

Samoin kuin Chow'n ja Caon (2008), Françan ym. (2010) ja Stankovicin ym. (2013) tutkimuksien tuloksissa, ei johdon sitoutuminen korostunut ketterän ohjelmistokehityksen menestystekijänä myöskään tämän tutkimuksen tuloksissa. Tämä saattaa johtua siitä, että kaikkien tutkimuksien vastauksissa ketteryyttä vaalivan yrityskulttuurin on katsottu kuvaavan samaa ilmiötä kuin vahva liikkeenjohdon sitoutumisenkin – todennäköisesti organisaatiossa, jossa yrityskulttuuri tukee ketteryyttä, myös liikkeenjohdon tuki ketteryydelle on kiistaton. Toisaalta voidaan ajatella, ettei johdon sitoutumisen määrittely sovellu enää kovinkaan hyvin nykyaikaiseen työhön, jossa itseohjautuvuuden merkitys korostuu yhä enemmän. Tällaisessa ympäristössä, jossa tiimit ja yksilöt voivat tehdä valintoja tavastaan tehdä kehitystyötä, ei tiimillä välttämättä ole tarvetta organisaation sisäiselle tukijalle, joka vastustaisi mahdollista kritiikkiä ja tukisi näin tiimin toimintaa, minkä takia johdon sitoutuminen ei mahdollisesti olisikaan tärkeä ketterän ohjelmistokehityksen menestystekijä.

Tiiviisti yrityskulttuuriin, organisaation itseohjautuvuuteen sekä johtamisrakenteeseen liittyvä menestystekijä on päätöksenteon nopeus. Haastatteluissa tämä tekijä mainittiin yrityskulttuurin yhteydessä ja osittain se voidaan nähdä menestystekijänä, joka edellyttää organisaatiolta matalaa hierarkiaa ja itseohjautuvuutta. Samoin päätöksenteon nopeuteen voidaan katsoa vaikuttavan myös kommunikoinnin sekä asiakasyhteistyön (Misra ym., 2009). Omana tekijänään päätöksenteon nopeutta tulisi tarkastella sen takia, ettei se automaattisesti ole seurausta edellä mainituista tekijöistä: kehitystiimin tulee tietoisesti hyödyntää matalan hierarkian tuomaa vapautta tehdä itsenäisesti kehitystyöhön liittyviä päätöksiä nopeasti, jotta päätöksenteon nopeus voi toteutua – myöskään hyvä

kommunikointi ja asiakasyhteistyö eivät automaattisesti johda nopeaan päätöksentekoon, ellei kehitystiimi pysty kommunikoimaan asiakkailleen kulloisenkin kehitysvaiheen tärkeimmistä aiheista ja hyödyntämään tästä saatua informaatiota osana päätöksentekoa.

Osana tutkimuskysymykseen vastaamista tutkittiin myös tekijöitä, joilla ei ole vaikutusta ketterän ohjelmistokehitysprojektin menestykseen. Kaksi tekijää korostuivat haastateltavien vastauksissa: tiimin maantieteellinen jakauma ja projektin luonne. Tuloksien mukaan, niin kauan kuin kommunikaatio on toteutettavissa toimivien työkalujen ja viestinnän toimintamallien avulla, ei tiimin jäsenten tarvitse tehdä kehitystyötä fyysisesti samassa paikassa, mikä on ristiriidassa Chow'n ja Caon (2008) sekä Françan ym. (2010) tutkimustuloksien kanssa, mutta tukee Misran ym. (2009) löydöksiä. Tulos, jonka mukaan projektin luonteella ei ole merkitystä ohjelmistokehitysprojektin onnistumiselle, on selkeä ristiriita Stankovicin ym. (2013) tutkimustuloksien kanssa, sillä he raportoivat projektin luonteen ainoana ketterän ohjelmistokehityksen kriittisenä menestystekijänä. Toisaalta tämä tutkimus näyttäisi tukevan Chow'n ja Caon (2008) sekä Françan ym. (2010) löydöksiä projektin luonteen vähäisestä merkityksestä. Kun ketterän ohjelmistokehityksen hyödyntäminen erilaisissa liiketoimintaympäristöissä yleistyy, on vaikea ajatella, ettei myös esimerkiksi turvallisuuden näkökulmasta kriittisiä ohjelmistoja voitaisi kehittää ketterästi, olettaen, että projektissa huomioidaan projektin luonteen tuomat rajoitteet sekä erikoistekijät ja analysoidaan ne projektin vaiheet, jossa tarvitaan erityistä huolellisuutta.

7.2 Tuloksien merkitys teorialle sekä käytännölle

Näiden tuloksien valossa vaikuttaa siltä, että tämä tutkimus tuo oman näkemyksensä ketterän ohjelmistokehityksen menestystekijöistä aikaisemmin toteutettujen tutkimuksien rinnalle vahvistaen osaa aikaisemmista havainnoista ja haastaen osan niistä. Tämä on toisaalta odotettavaa, sillä aiemmat tutkimukset ovat esittäneet erilaisia havaintoja ketterän ohjelmistokehityksen menestystekijöistä ja tämän tutkimuksen tutkimusasetelma rakentui Chow'n ja Caon (2008) sekä Misran ym. (2009) tutkimusasetelmien synteesinä. Tuloksia tarkastellessa voidaan myös esittää johtopäätös, että ketterän ohjelmistokehityksen menestystekijöistä puhuttaessa ei voida keskittyä täysin toisistaan erillään vaikuttaviin tekijöihin, vaan menestystekijöitä tulisi tarkastella toisaalta myös osiensa summana, jossa tekijät vaikuttavat toisiinsa ja sitä kautta ketterän ohjelmistokehityksen onnistumiseen. Tällöin esimerkiksi päätöksenteon nopeus ei ole itseisarvo, jos päätökset tehdään nopeasti vääristä syistä. Mutta kun nopea päätöksenteko yhdistetään tehokkaaseen julkaisustrategiaan sekä sen pohjalta saatuun palautteeseen ja tästä luotuun yhteiseen suuntaan asiakasyrityksen kanssa, tehdään päätökset oikeista asioista, mikä tekee myös päätöksenteon nopeudesta ketterän ohjelmistokehityksen menestystekijän.

Ketterää ohjelmistokehitystä tekeville yrityksille tämän tutkimuksen merkitys voidaan tarkastella kahdesta näkökulmasta. Voidaan mitä todennäköi-

simmin olettaa, että digitalisaation tuomat muutokset liiketoimintaympäristöön asettavat uudenlaisia vaatimuksia myös ohjelmistoyrityksille: asiakasyritysten vaatimustason kasvaessa ja kilpailun tiukentuessa tulee yritysten priorisoida, mihin keskittyä toimintansa kehittämisessä. Tutkimuksen tulokset voivat auttaa yrityksiä kehittämään ohjelmistokehityksen ketteryyttä valottamalla sitä, mihin tekijöihin yritysten kannattaisi omassa toiminnassaan paneutua: empiirisen tutkimuksen tulosten pohjalta tunnistetut seitsemän menestystekijää antavat yrityksille mahdollisuuden peilata omaa toimintaansa suhteessa näihin ja kehittää omaa toimintaansa tämän pohjalta.

Tuloksien perusteella yrityksien tulisi pyrkiä varmistamaan asiakasyhteistyön sujuvuus: avainasemassa ovat tällöin yhteinen suunta ja ymmärrys kehitysprojektin tavoitteesta, läpinäkyvyys sekä asiakkaan osallistuminen kehitystyön. Tärkeää on panostaa myös kommunikaation välittömyyteen – joko suosimalla kasvokkain tapahtuvaa viestintää tai erilaisia digitaalisia viestintävälineitä. Hyödyntämällä tehokasta julkaisustrategiaa tulisi toimittaa kehitettävää ohjelmistoa loppuasiakkailleen nopeassa syklissä ja käyttää täten saatavaa palautetta kehitysprojektin suuntaamiseen niin, että tuotettava ohjelmisto täyttää loppuasiakkaiden vaatimukset parhaalla mahdollisella tavalla. Ohjelmistokehitystiimiä rakentaessa yrityksien tulisi huolehtia siitä, että tiimin jäsenillä on tarvittavaa kompetenssia sekä mahdollisuus oppia – joissakin tilanteissa gurustatustakin tärkeämpää on, että tiimin jäsenillä on kykyä ja halua oppia uutta kehitystyön edetessä. Yrityksien tulisi pyrkiä kasvattamaan tai vaalimaan ketterälle ohjelmistokehitykselle suotuisaa yrityskulttuuria, jossa korostuvat matalan hierarkian sekä itseohjautuvuuden merkitys. Samalla tiimejä ja yksilöitä tulisi rohkaista hyödyntämään omaa valtaansa päätöksentekoon ja pyrkiä näin lisäämään päätöksenteon nopeutta kehitystyön aikana. Näiden huomioiden lisäksi luvussa 6 esitetään konkreettisia toimintamalleja, joita ketterää ohjelmistokehitystä jo tekevät tai kohti sitä siirtyvät yritykset voivat adaptoida omaan toimintaansa kehittääkseen ohjelmistokehityksensä ketteryyttä ja saavuttaakseen sitä kautta onnistuneita ohjelmistokehitysprojekteja.

8 YHTEENVETO

Tässä tutkielmassa tarkasteltiin ketterää ohjelmistokehitystä ja ketterän ohjelmistokehityksen menestystekijöitä. Ketterä ohjelmistokehitys on ilmiönä hankalasti määriteltävissä: aiheen tutkimus on epäselvää, suuri osa akateemisista artikkeleista ei perustu teorialle ja täten tutkimustieto ei ole kovinkaan yhtenäistä. Ketterä ohjelmistokehitys yhdistetään kuitenkin yhä suuremmissa määrin onnistuneisiin ohjelmistokehitysprojekteihin, minkä takia oli mielekästä tutkia ketterää ohjelmistokehitystä sekä ketterän ohjelmistokehityksen menestystekijöitä.

Ketterän ohjelmistokehityksen määrittelyn lähtökohtana on usein käytetty Agile Manifeston neljää arvoa. Ne ovat kuitenkin epäselviä ja laveita tieteellisen työn perustaksi. Kun tarkastellaan Agile Manifeston arvoja, ketteriä periaatteita ja tieteellisen kirjallisuuden määritelmiä ketterästä ohjelmistokehityksestä, voidaan todeta, että muutos on olennainen osa ketterää ohjelmistokehitystä. Muutos esiintyy jossakin muodossa kaikissa tutkituissa määrittelyissä. Muita ketterään ohjelmistokehitykseen yhdistettäviä termejä ovat muun muassa keveys, inkrementaalisuus, iteratiivisuus, yksilöt, kanssakäyminen, toimiva ohjelmisto ja asiakasyhteistyö. Tärkeä huomio on myös se, että ketteryys tulisi määritellä ennen kaikkea liiketoiminnallisesta näkökulmasta: ketterä ohjelmistokehitys on pohjimmiltaan keino yrityksille hankkia kilpailuetua, tehdä ohjelmistokehitystä taloudellisemmin ja tarjota asiakkailleen liiketoiminnallista arvoa.

Ketteryyttä saavuttaakseen yritys voi toimia ketterien periaatteiden mukaisesti, hyödyntää toiminnassaan ketteriä menetelmiä tai osaa menetelmien käytänteistä. Tutkielmassa käsiteltiin tarkemmin ketteriä periaatteita sekä esiteltiin suosituimmat ketterät menetelmät (Scrum, Extreme Programming, Lean-ohjelmistokehitys ja Kanban) ja ketterät käytänteet.

Päätutkimuskysymykseen vastattiin empiirisen tutkimuksen avulla. Aineisto kerättiin laadullisilla teemahaastatteluilla, joissa haastateltiin yhteensä kuutta ketterää ohjelmistokehitystä tekevän yrityksen edustajaa. Empiirisen tutkimuksen tulosten pohjalta tunnistettiin seitsemän ketterän ohjelmistokehityksen menestystekijää: asiakasyhteistyö, kommunikointi, julkaisustrategia, kompetenssi, harjoittelu ja oppiminen, yrityskulttuuri sekä päätöksenteon no-

peus. Nämä tulokset vahvistavat osan aikaisemmista löydöksistä ja haastavat osan niistä.

Tutkimukseen sisältyy rajoitteita, joita pohdittiin myös luvussa 5.6, selkeimmän liittyessä tutkimuksen pieneen otokseen. Tämä rajoittaa tuloksien yleistettävyyttä, eikä tuloksista tällaisenaan voida vetää esimerkiksi koko ohjelmistokehityksen toimialaa koskevia johtopäätöksiä. Toisena rajoitteena tuloksien yleistettävyydelle voidaan pitää tutkimuksessa haastateltujen edustamien yrityksien jakaumaa henkilöstömäärän suhteen. Luvussa 6.2 todettiin, että viisi kuudesta yrityksestä edusti Pk-yritystä, yhden ollessa henkilöstömäärän osalta määritelmää suurempi. Otanta ei täten edusta suuryritysten näkökulmaa ketterään ohjelmistokehitykseen juurikaan. Kolmas tutkimuksen rajoite liittyy aiheen tiimoilta aikaisemmin tehtyjen empiiristen tutkimusten niukkuuteen. Vaikka tieteellistä kirjallisuutta ketterästä ohjelmistokehityksestä on kertynyt kohta kahden vuosikymmenen ajan ja parhaista käytänteistä on kirjoitettu paljon, ei kattavaa kvantitatiivisesti tutkittua tietoa ketterän ohjelmistokehityksen menestystekijöistä ole suurta määrää. Toisaalta tilanne tarjoaa lukuisia hedelmällisiä jatkotutkimusaiheita.

Jatkotutkimusaiheina olisi mielenkiintoista tarkastella tiettyä osaa tekijöistä yksityiskohtaisemmin vaikkapa tapaustutkimuksen keinoin: kiinnostavaa olisi selvittää esimerkiksi, miten eri kokoluokan ohjelmistokehitysprojekteissa asiakkaan osallistuminen voidaan varmistaa tai miten organisaatio, jossa työskennellään useamman ohjelmistokehitysprojektin parissa samanaikaisesti, pystyy varmistamaan, että jokaisessa kehitystiimissä on tarvittava substanssiosaaminen sekä mahdollisuus harjoitteluun ja oppimiseen.

Toisaalta mielenkiintoisena tutkimussuuntana voisi olla myös alun perin Chow'n ja Caon (2008) esittelemän tutkimusasetelman mukaisen tutkimuksen toteuttaminen suomalaisten ohjelmistoyritysten joukossa. Luvussa 4.2 esitettiin perusjoukkoon ja otantaan liittyviä ongelmakohtia alkuperäisen tutkimuksen ja myöhemmin samaa tutkimusasetelmaa hyödyntävän Stankovicin ym. (2013) tutkimuksen välillä, jotka ovat osaltaan voineet vaikuttaa siihen, että tutkimustulokset ovat keskenään ristiriidassa. Suomalaiset ohjelmistoyritykset voisivat edustaa paremmin alkuperäisen tutkimuksen kaltaista perusjoukkoa, mikä saattaisi antaa luotettavampia tuloksia. Täten olisi mielenkiintoista nähdä, voitaisiinko tällaisella tutkimuksella vahvistaa Chow'n ja Caon (2008) sekä Françan ym. (2010) tutkimuksien tuloksia vai tuottaisiko tutkimus täysin uusia löydöksiä.

Tässä tutkimuksessa kaikkien haastateltujen henkilöiden edustamat organisaatiot tekevät pääsääntöisesti yritysten välistä liiketoimintaa, mitä voidaan pitää myös rajoitteena tulosten yleistettävyydelle. Niinpä olisikin mielenkiintoista tutkia jatkossa myös täysin kuluttajille suunnattujen ohjelmistojen parissa toimivien yritysten ketterän ohjelmistokehityksen menestystekijöitä. Tällaisia yrityksiä voisivat olla esimerkiksi pelikehitystä tekevät yritykset.

On selvää, että ketterän ohjelmistokehityksen menestystekijät ovat mielenkiintoinen tutkimusaihe myös tulevaisuudessa. Kun edelleen kiihtyvä digitalisaatio asettaa uudenlaisia vaatimuksia ohjelmistoyrityksille, on yhä tärke-

ämpää ymmärtää, mitkä tekijät mahdollistavat onnistuneen ketterän ohjelmistokehityksen.

LÄHTEET

Abbas, N., Gravell, A. & Wills, G. (2008). Historical Roots of Agile Methods: Where did "Agile Thinking" Come from? Teoksessa P. Abrahamsson, R. Baskerville, K.Conboy, B.Fitzgerald & X.Wang (toim.) *Proceedings of the 9th International Conference on Agile Processes in Software Engineering and Extreme Programming* (s. 94-103). Berliini: Springer

Abrahamsson, P., Warsta, J., Siponen, M. & Ronkainen, J. (2003). New Directions on Agile Methods: A Comparative Analysis. Teoksessa A. Jacobs & F. Titsworth (toim.) *Proceedings of the 25th International Conference on Software Engineering,* (s. 244-254). Portland: IEEE Computer Society.

Abrahamsson, P., Conboy, K. & Wang, X. (2009). 'Lots done, more to do': the current state of agile systems development research. *European Journal of Information Systems, 18,* 281–284.

Agile Alliance. (2001). Manifesto for agile software development. Haettu 5.2.2016 osoitteesta http://agilemanifesto.org/

Ahmad, M. O., Markkula, J. & Oivo, M. (2013). Kanban in software development: A systematic literature review. Teoksessa O. Demirors & O. Turetken (toim.), *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference* (s. 9–16). Santander: IEEE Computer Society.

Agarwal, R., & Umphress, D. (2008). Extreme programming for a single person team. Teoksessa *Proceedings of the 46th Annual Southeast Regional Conference* (s. 82–87). Auburn: ACM.

Anderson, D., Concas, G., Lunesu, M. I., & Marchesi, M. (2011). Studying Lean-Kanban Approach Using Software Process Simulation. Teoksessa A. Sillitti, O. Hazzan, E. Bache & X. Albaladejo (toim.), *International Conference on Agile Software Development* (s. 12–26). Madrid: Springer.

Atkinson, R. (1999). Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International Journal of Project Management, 17(6)*, 337–342.

Beck, K. (1999a). *Extreme Programming Explained: Embrace Change.* Boston: Addison-Wesley Professional

Beck, K. (1999b). Embracing Change with Extreme Programming. *Computer, 32(10),* 70-77.

Bullen, C. V. & Rockart, J. F. (1981). *A primer on critical success factors* (Report No. 69, 1220-81.) MIT, Sloan School of Management.

Chow, T. & Cao, D. B. (2008). A survey study of critical success factors in agile software projects. *Journal of Systems and Software, 81(6)*, 961–971.

Chuang, S. W., Luor, T. & Lu, H. P. (2014). Assessment of institutions, scholars, and contributions on agile software development (2001–2012). *Journal of Systems and Software, 93,* 84–101.

Cockburn, A. & Williams, L. (2003). Agile Software development: It's about Feedback and Change. *IEEE Computer, 3(5),* 39-43.

Cohen, D., Lindvall, M. & Costa, P. (2004). An introduction to agile methods. *Advances in Computers, 62,* 1–66.

Conboy K. (2009). Agility from first principles reconstructing the concept of agility in information systems development. *Information Systems Research 20(3)*, 329–354.

Dey, I. (1993). *Qualitative Data Analysis: A User Friendly Guide for Social Scientists.* Lontoo: Routledge.

Dingsøyr, T., Nerur, S., Balijepally, V. & Moe, N. B. (2012). A decade of agile methodologies Towards explaining agile software development. *Journal of Systems and Software, 85(6)*, 1213–1221.

Dybå, T. & Dingsøyr, T. (2008). Empirical studies of agile software development A systematic review. *Information and Software Technology, 50(9)*, 833–859

Erickson, J., Lyytinen, K. & Siau, K. (2005). Agile modeling, agile software development, and extreme programming: the state of research. *Journal of database Management, 16(4)*, 88–100.

Eskola, J. & Suoranta, J. (2008). *Johdatus laadulliseen tutkimukseen*. (8. painos). Tampere: Vastapaino.

França, A. C. C., da Silva, F. Q., & de Sousa Mariz, L. M. (2010). An empirical study on the relationship between the use of agile practices and the success of Scrum projects. Teoksessa B. Rossi (toim.) *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (s. 37–41). Bolzano: ACM.

Hirsjärvi, S. & Hurme, H. (2014). *Tutkimushaastattelu: teemahaastattelun teoria ja käytäntö*. Helsinki: Gaudeamus.

Hirsjärvi, S., Remes, P. & Sajavaara, P. (2004). *Tutki ja kirjoita*. (10. osin uudistettu painos). Helsinki: Kusannusosakeyhtiö Tammi.

Hossain, E., Babar, M. A., & Paik, H. Y. (2009). Using Scrum in Global Software Development: A Systematic Literature Review. Teoksessa P. Kellenberger (toim.) *Fourth IEEE International Conference on Global Software Engineering ICGSE 2009* (s. 175–184). Limerick: IEEE Computer Society

Ikonen, M., Pirinen, E., Fagerholm, F., Kettunen, P. & Abrahamsson, P. (2011). On the impact of Kanban on software project work: An empirical case study investigation. Teoksessa I. Perseil, K. Breitman & R. Sterrit (toim.), *Engineering of Complex Computer Systems (ICECCS), 2011 16th IEEE International Conference* (s. 305–314). Las Vegas: IEEE Computer Society.

Kinnunen, H. (2015). *Ohjelmistokehityksen ketteryys ja sen mittaaminen.* Tietojärjestelmätieteen pro gradu -tutkielma. Jyväskylän yliopisto.

Kniberg, H. & Skarin, M. (2010). *Kanban and Scrum – making the most of both.* InfoQ.

Laanti, M., Similä, J. & Abrahamsson, P. (2013). Definitions of agile software development and agility. Teoksessa F. McCaffery, R. V. O'Connor, R. Messnarz (toim.), *Systems, Software and Services Process Improvement, EuroSPI 2013* (s. 247–258). Berliini: Springer-Verlag

Lee, G., & Xia, W. (2010). Toward agile: an integrated analysis of quantitative and qualitative field data on software development agility. *MIS Quarterly, 34(1)*, 87–114.

Lyytinen, K. & Rose, G. (2006). Information system development agility as organizational learning. *European Journal of Information Systems, 15,* 183-199.

Miles, M. B., Huberman, A. M., & Saldaña, J. (2014). *Qualitative data analysis: a methods sourcebook* (3. painos). Thousand Oaks, Kalifornia: SAGE Publications Inc.

Misra, S. C., Kumar, V. & Kumar, U. (2009). Identifying some important success factors in adopting agile software development practices. *Journal of Systems and Software, 82(11)*, 1869–1890.

Moe, N. B., Dingsøyr, T., & Dybå, T. (2010). A teamwork model for understanding an agile team: A case study of a Scrum project. *Information and Software Technology, 52(5)*, 480–491.

Myers, M. D. & Newman, M. (2007). The qualitative interview in IS research: Examining the craft. *Information and Organization, 17(1)*, 2–26.

Petersen, K., & Wohlin, C. (2011). Measuring the flow in lean software development. *Software: Practice and experience, 41(9)*, 975–996.

Peterson, D. (2015) What is Kanban? Haettu 18.7.2017 osoitteesta http://kanbanblog.com/explained/

Poppendieck, M. & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit.* Boston: Addison-Wesley.

Poppendieck, M. & Cusumano, M. A. (2012). Lean software development: A tutorial. *IEEE Software, 29(5)*, 26–32.

Rockart, J. F. (1979). Chief executives define their own data needs. *Harvard Business Review, 57(2)*, 81–93.

Rodríguez, P., Markkula, J., Oivo, M. & Turula, K. (2012). Survey on Agile and Lean Usage in Finnish Software Industry. Teoksessa *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement* (s. 139–148). Lund: ACM.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering, 14(2)*, 131–164.

Schultze, U. & Avital, M. (2011). Designing interviews to generate rich data for information systems research. *Information and Organization, 21(1)*, 1–16.

Schwaber, K. & Sutherland, J. (2013). Scrum Guide. Haettu 19.2.2016 osoitteesta http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf#zoom=100

Standish Group International. (2013) CHAOS Manifesto 2013. Haettu 3.9.2015 osoitteesta https://larlet.fr/static/david/stream/ChaosManifesto2013.pdf

Stankovic, D., Nikolic, V., Djordjevic, M. & Cao, D. B. (2013). A survey study of critical success factors in agile software projects in former Yugoslavia IT companies. *Journal of Systems and Software, 86(6)*, 1663–1678.

Strauss, A. L. (1987). *Qualitative analysis for social scientists.* Cambridge: Cambridge University Press.

Sutherland, J. (2010). *Scrum handbook.* Sommerville: Scrum Training Institute Press.

Tilastokeskus (2017). Pienet ja keskisuuret yritykset. Haettu 10.9.2017 osoitteesta http://www.stat.fi/meta/kas/pienet_ja_keski.html

van Waardenburg, G. & van Vliet, H. (2013). When agile meets the enterprise. *Information and Software Technology, 55(12)*, 2154–2171.

VersionOne (2015). 9TH ANNUAL State of Agile™ Survey. Haettu 12.3.2016 osoitteesta http://info.versionone.com/state-of-agile-development-survey-ninth.html

Wang, X., Conboy, K., & Cawley, O. (2012). "Leagile" software development: An experience report analysis of the application of lean approaches in agile software development. *Journal of Systems and Software, 85(6)*, 1287–1299.

Yang, C., Liang, P. & Avgeriou, P. (2016). A systematic mapping study on the combination of software architecture and agile development. *Journal of Systems and Software, 111,* 157–184.

LIITE 1 HAASTATTELURUNKO

TAUSTATIEDOT

- Mikä on yrityksen koko? Liikevaihto? Henkilöstömäärä?
- Millä markkinoilla yritys pääasiassa toimii? B2B / B2C?
- Minkälaisia ohjelmistoja yritys pääasiallisesti kehittää?
- Kuinka monta ohjelmistokehitysprojektia yrityksessä on yleensä käynnissä samanaikaisesti?
- Haastateltavan tehtävänimike / rooli yrityksessä?

KETTERYYS

- Millä tavoin yritys on ketterä? Käyttääkö menetelmää tai käytänteitä vai toimiiko ketterien periaatteiden mukaisesti?
- Kuinka kauan ketteriä menetelmiä, käytänteitä tai periaatteita on hyödynnetty?
- Mitä hyötyjä ketteryydestä on saatu? Miten ketteryys on parantanut ohjelmistokehitystä?

ONNISTUMISEN MÄÄRITTELY

- Millä tavoin onnistunut ohjelmistokehitysprojekti määritellään yrityksessä?
- Mitataanko onnistumista jotenkin? Mitä mittareita yrityksellä on käytössä?

MENESTYSTEKIJÄT – HAASTATELTAVAN ESITTÄMÄT

* Käydään läpi kriittisen menestystekijän määritelmä.

- Mitkä tekijät ovat ketterän ohjelmistokehityksen kriittisiä menestystekijöitä yrityksen ohjelmistokehitysprojekteissa? Miksi?
    - Mikä mahdollistaa kyseisen menestystekijän mukaisen toiminnan ohjelmistokehitysprojekteissa?
        - Käytänteet, prosessit, toimintamallit? Ohjelmistot? Roolit?

MENESTYSTEKIJÄT – MIELIPIDE KIRJALLISUUDESSA ESITETYISTÄ

* Esitellään lista kirjallisuudessa esitetyistä menestystekijöistä

- Mitkä esitetyistä tekijöistä ovat kriittisiä menestystekijöitä yrityksen ohjelmistokehitysprojekteissa? Miksi?

- Millä tekijöillä on vähäinen merkitys yrityksen ohjelmistokehitysprojekteissa? Miksi?
- Muut mahdolliset mielipiteet esitetyistä menestystekijöistä tai niiden attribuuteista?

LIITE 2 KIRJALLISUUDESSA ESITETYT MENESTYSTEKIJÄT

Asiakastyytyväisyys
- Projektissa asiakastyytyväisyyden saavuttaminen priorisoidaan erittäin tärkeäksi.

Asiakkaan osallistuminen
- Asiakas tekee läheistä yhteistyötä kehitystiimin kanssa (esim. asiakkaan edustaja työskentelee yhdessä kehitystiimin kanssa, *on-site customer*).
- Asiakas sitoutuu projektiin, on motivoitunut, aktiivinen ja pitää itseään vastuunalaisena projektin osana
- Asiakkaalla on täysi päätösvalta projektissa (esim. vaatimuksien ja muutoksien hyväksyminen tai hylkääminen).

Harjoittelu ja oppiminen
- Kehitystiimin jäsenet ovat yleisesti halukkaita oppimaan toisiltaan sekä opettamaan toisiaan mentoroinnin ja ohjattujen keskustelujen avulla.

Henkilökohtaiset ominaisuudet
- Suurimmalla osalla kehitystiimin jäsenistä on vahvat vuorovaikutus- ja kommunikointitaidot.
- Seuraavat ominaisuudet kuvaavat suurinta osaa kehitystiimin jäsenistä: rehellinen, motivoitunut, yhteistyökykyinen, vastuuntuntoinen ja oppimishaluinen.

Johdon sitoutuminen
- Projektilla on vahva johdon tuki (”johto” voi viitata johtoryhmään, toimitusjohtajaan, talousjohtajaan yms., jolla on vaikutusta päätöksentekoon).
- Projektilla on sitoutunut tukija organisaatiossa, joka esim. vastustaa kritiikkiä ja tukee ketterien menetelmien käyttöä ei-ketterässä ympäristössä.

Julkaisustrategia
- Toimivaa ohjelmistoa toimitetaan säännöllisesti ja lyhyin aikavälein.
- Ohjelmiston tärkeimmät ominaisuudet toimitetaan ensin.

Ketterän ohjelmistokehityksen tekniikat
- Projekti sisältää seuraavat ketterät tekniikat: ennalta määritellyt ohjelmointistandardit, yksinkertainen suunnittelu, tarkka refaktorointi sekä jatkuva yksikkö- ja integraatiotestaus.
- Projektin dokumentointi on ketterää (keskitytään tarpeelliseen informaatioon, mutta ei dokumentoida liian yksityiskohtaisesti).

Kommunikointi ja neuvottelu
- Projekti sisältää prosesseja, jotka mahdollistavat henkilöstön nopean ja tehokkaan kommunikoinnin kehittäjien, tuotantohenkilöstön, tukihenkilöstön, asiakkaiden, johdon ja liiketoimintaedustajien välillä.
- Kommunikointi ja neuvottelu tapahtuvat pääosin kasvotusten, fyysisesti toistensa läheisyydessä ja saman aikavyöhykkeen sisällä olevien ihmisten välillä.

- Suurin osa projektissa työskentelevistä ihmisistä ovat ystävällisiä toisiaan kohtaan, minkä ansiosta he kommunikoivat toistensa kanssa luottamuksellisesti ja hyväntahtoisesti.

Kompetenssi

- Kehitystiimin jäsenet ovat teknisesti osaavia ja kokeneita henkilöitä.
- Kehitystiimin jäsenet ovat kehittäneet vastaavia ohjelmistoja aiemmin ja heillä on käytännön kokemusta vastaavasta teknologiasta.

Organisatorinen ympäristö

- Organisaatiossa on ketterälle toiminnalle sopiva palkitsemisjärjestelmä (esim. palkitsemisessa huomioidaan yksilö- ja tiimitason työpanos).
- Tiimi työskentelee ketteryyttä edistävässä työympäristössä (esim. toimisto, jossa on työasemat pariohjelmoinnille ja yhteinen tila).
- Projektissa hyödynnetään ketterään ohjelmistokehitykseen sopivia alustoja, teknologioita ja työkaluja.

Projektin aikataulu

- Projektilla on dynaaminen aikataulu.

Projektinhallinnan prosessi

- Projektin vaatimusmäärittely on tehty ketterästi (esim. vaatimukset määritellään aluksi hyvin yleisellä tasolla, mikä jättää paljon tilaa tulkinnalle ja mukautumiselle).
- Projektinhallinta on ketterää (esim. suunnitelmia ei dokumentoida yksityiskohtaisesti ja muutokset vaatimuksissa sisällytetään projektisuunnitelmaan).
- Projektissa noudatetaan ketterää konfiguraationhallintaa, ketterää etenemisen seurantaa, täsmällistä kommunikointia ja säännöllistä työaikaa.

Projektin luonne

- Projektissa ei kehitetä ohjelmistoa, joka olisi luonteeltaan kriittinen turvallisuuden näkökulmasta (esim. lentoliikenteessä käytettävät ohjelmistot).

Projektin määrittelyn prosessi

- Projektin laajuus ja tavoitteet on määritelty tarkasti.
- Projektille on tehty kustannus- ja riskianalyysi.

Projektin tyyppi

- Projekti on tyypiltään sellainen, että sen laajuus muuttuu ja uusia vaatimuksia tunnistetaan projektin edetessä.

Päätöksenteon nopeus

- Projektin kannalta tärkeät päätökset pyritään tekemään nopeasti, lyhyessä aikaikkunassa.

Seuranta

- Kehitystiimin edistymistä seurataan laadullisella tarkkailulla, määrällisten suorituskyvyn mittareiden sijaan.

Suunnittelu

- Kehitystiimi luottaa sisäisiin, epävirallisiin ja kirjaamattomiin suunnitelmiin, virallisten ja kirjattujen suunnitelmien sijaan.

Tiimiympäristö

- Kehitystiimi työskentelee itseorganisoituvasti (tiimi luottaa kykyynsä ratkaista ongelmia autonomisesti).

- Projektissa ei työskentele useampaa itsenäistä kehitystiimiä yhtäaikaisesti.

Tiimin koko

- Projektissa työskentelevä kehitystiimi on kooltaan pieni (alle 20–40 henkilöä).

Tiimin kyvykkyys

- Tiimin jäsenet ovat motivoituneita ja sitoutuneita projektin onnistumiseen.
- Projektin johto tuntee ketterät periaatteet sekä käytänteet ja hyödyntää kevyttä /sopeutuvaa johtamisotetta.
- Kehitystiimi saa asianmukaisen teknisen koulutuksen projektin aihepiiriin ja ketteriin prosesseihin.

Tiimin maantieteellinen jakauma

- Kehitystiimin jäsenet ovat maantieteellisesti toistensa läheisyydessä.
- Muut tiimit, organisaation sisäiset tai ulkoiset, joiden kanssa kehitystiimi on vuorovaikutuksessa, ovat maantieteellisesti lähellä kehitystiimiä.

Yhteiskunnallinen kulttuuri

- Suurin osa kehitystiimin jäsenistä edustaa samankaltaista yhteiskunnallista kulttuuria, vaikka he edustaisivatkin eri kansallisuuksia tai olisivat kotoisin eri puolilta maata.
- Yleisesti ihmiset maassamme ovat seurallisia, dynaamisia ja eteenpäin suuntautuvia.

Yrityskulttuuri

- Yrityksessä…
  - ei ole byrokraattista johtamisrakennetta.
  - kannustetaan nopeaan kommunikointiin ja asiakaspalautteeseen.
  - vallitsee ihmisiin luottava ja asiakaskeskeinen kulttuuri.
  - hyväksytään muuttuvat vaatimukset.
- Yrityksen johdon, kehittäjien ja testaajien välillä vallitsee täysi yhteisymmärrys ketterien menetelmien käytöstä.
- Yrityksen johto tukee kehitystiimin jäsenten päätöksiä.