Jaakko Stenudd

Ketterän ohjelmistokehityksen menestystekijät

Tekniikan ja innovaatiojohtamisen yksikkö

Pro gradu -tutkielma

Tietojärjestelmätiede, Kauppatieteiden maisteri

Vaasa 2020

VAASAN YLIOPISTO
Tekniikan ja innovaatiojohtamisen yksikkö
Tekijä: Jaakko Stenudd
Tutkielman nimi: Ketterän ohjelmistokehityksen menestystekijät
Tutkinto: Kauppatieteiden maisteri
Oppiaine: Tietojärjestelmätiede
Työn ohjaaja: Teemu Mäenpää
Valmistumisvuosi: 2020 **Sivumäärä:** 84

TIIVISTELMÄ:
Tässä pro gradu -tutkielmassa tarkasteltiin ketterää ohjelmistokehitystä ja ketterän ohjelmistokehityksen kriittisiä menestystekijöitä. Ketterää ohjelmistokehitystä on tutkittu paljon ja ketteriä menetelmiä käytetään ohjelmistokehityksen lisäksi lukuisilla eri toimialoilla. Ketterän ohjelmistokehityksen lisäksi tutkielmassa keskitytään ketterän ohjelmistokehityksen kriittisiin menestystekijöihin, joilla tarkoitetaan projektitiimin ominaisuuksia, toimintatapoja ja edellytyksiä, jotka antavat parhaan mahdollisuuden onnistumiseen. Toisin sanoen, tutkielma pyrkii kertomaan, mitä vaaditaan ketterässä ohjelmistoprojektissa onnistumiseen.

Ketteriä ohjelmistokehitysmenetelmiä voidaan soveltaa hyvin erilaisin tavoin, mutta tyypillisesti ketterä ohjelmistokehitys tapahtuu sprinteissä, joiden aikana asiakastarpeiden määrittely tarkentuu tiiviin kommunikoinnin ja asiakasyhteistyön myötä. Koska ketteriä menetelmiä käytetään erilaisissa projekteissa ja erilaisin tavoittein, myös kriittisten menestystekijöiden painotukset ja tärkeysjärjestykset ovat projektikohtaisia. Siitä huolimatta huolellisen laadullisen tutkimuksen avulla oli mahdollista selvittää sellaisia menestystekijöitä, jotka ovat onnistumisen kannalta kriittisiä kaikille ketterille ohjelmistokehitysprojekteille. Tutkielman tutkimuskysymyksenä on siis: ”Mitkä ovat ketterän ohjelmistokehitysprojektin onnistumisen kriittiset menestystekijät?” Tutkielman tavoitteena on löytää ketterän ohjelmistokehityksen kriittiset menestystekijät ja luoda käytännön avuksi malli, jonka mukaan ketterän ohjelmistoprojektin edellytykset tulisi suunnitella, jotta projektilla olisi parhaat mahdollisuudet onnistua. Tutkielman tavoite pyrittiin saavuttamaan tarkastelemalla kattavasti ensin aiheeseen liittyviä tutkimuksia, jonka jälkeen haastateltiin alalla työskenteleviä asiantuntijoita teemahaastatteluiden muodossa. Teemahaastatteluiden tuloksia analysoitiin teemoittelun avulla.

Tuloksista löydettiin kuusi kriittistä menestystekijää ketterälle ohjelmistoprojektille, jotka ovat tiimin kyvykkyys, johdon tuki, asiakasyhteistyö, kommunikaatio, julkaisustrategia sekä organisaatiokulttuuri. Tutkielman tavoite saavutettiin löytämällä kriittiset menestystekijät ja luomalla edellä mainittu malli, johon organisaatiot voivat peilata toimintaansa ja tämän pohjalta kehittää omaa ketteryyttä sekä mahdollisuutta saavuttaa onnistuneita ohjelmistoprojekteja. Mallin keskeisimmän sanoman mukaan projektitiimin tulee varmistaa ensi kädessä asiakasyhteistyön sujuvuus varmistamalla toimiva kommunikaatio ja keskittyä vain asiakkaalle eniten arvoa tuottaviin toimiin. Tämä vaatii perusteellista vaatimusten määrittelyä projektin alkuvaiheessa sekä tuoteversioiden toimittamista asiakkaalle lyhyin ja säännöllisin syklein, jotta projekti etenee nopeasti olennaiseen keskittyen ja asiakas voi antaa palautetta kehitystyöstä tai vaatimusten muuttumisista. Projektitiimiltä vaaditaan syvää osaamista, jotta projektitiimillä olisi mahdollisuudet vastata nopeasti vaativiin tehtäviin ja vaatimuksiin sekä laajaa osaamista, jotta vältytään pullonkauloilta. Tärkeää on myös projektitiimin kyky edetä kehitystyössä kokeilemisen ja oppimisen kautta, jotta asiakkaalle eniten arvoa tuottavat asiat löydetään. Projektitiimin sekä koko organisaation tulisi vaalia yrityskulttuuria, joka kannustaa työn tekemiseen kokeilunhalun ja oppimishalukkuuden kautta, itseohjautuvaan työskentelyyn sekä matalahierarkkiseen päätöksentekoon.

AVAINSANAT: ketterä ohjelmistokehitys, kriittiset menestystekijät, Scrum, Lean, XP, Kanban

UNIVERSITY OF VAASA
Tekniikan ja innovaatiojohtamisen yksikkö
Author: Jaakko Stenudd
Thesis topic: Critical success factors of agile software development
Degree: Master of Economics
Subject: Information Systems
Supervisor: Teemu Mäenpää
Graduation year: 2020 **Pages:** 84

ABSTRACT:
This master's thesis researched agile software development and the critical success factors of agile software development. Agile software development has been researched extensively and agile methods are used in many different industries in addition to software development. Together with agile software development, this thesis focuses on the critical success factors of agile software development, which refer to the features, practices, and qualifications of a project team that provide the best chance of success. In other words, this thesis seeks to explain what is required to succeed in an agile software project.

Agile software development methods can be applied in very different ways, but typically agile software development takes place in sprints, during which the definition of customer needs is refined through quality communication and customer collaboration. Because agile methods are used in many different kind projects and for different purposes, the order of importance of critical success factors is also project-specific. Nevertheless, with the help of high-quality and careful qualitative research, it was possible to identify success factors that are critical to success for any agile software development project. The research question of the thesis is: "What are the critical success factors of agile software development project?" The objective of the thesis is to find the critical success factors of agile software development and to create a model for practical help, according to which the features, practices, and qualifications of an agile software project should be designed, that the project has the best chance of success. The objective of this thesis was to achieve by first conducting a comprehensive review of related studies, followed by interviewing experts working in the field in the form of thematic interviews.

The results found six critical success factors for agile software project, which are team capability, management support, customer collaboration, communication, delivery strategy, and organizational culture. The objective of the thesis was achieved by finding the critical success factors and creating the above-mentioned model in which organizations can mirror their activities, and on this basis develop their own agility as well as the possibility to achieve successful software projects. According to the key message of the model, the project team must first and foremost ensure the quality of customer cooperation by ensuring effective communication and focus only on the activities that create the most value for the customer. This requires a thorough definition of requirements at the beginning of the project and the delivery of product versions to the customer in short and regular cycles so that the project progresses quickly with a focus on the essentials and the customer can provide feedback on development work or requirements changes. Deep expertise is required from the project team to enable the project team to respond quickly to demanding tasks and requirements, as well as extensive expertise to avoid bottlenecks. Also important is the project team's ability to move forward in development work through experimentation and learning to find the things that create the most value for the customer. The project team and the entire organization should foster a corporate culture that encourages work through experimentation and a willingness to learn, self-directed work and low-hierarchy decision-making.

KEYWORDS: ketterä ohjelmistokehitys, kriittiset menestystekijät, Scrum, Lean, XP, Kanban

Sisällysluettelo

Kuvio- ja taulukkoluettelo

1 Johdanto

Tässä pro gradu -tutkielmassa tarkastellaan ja tutkitaan ketterän ohjelmistokehityksen määrittelyä, keskeisimpiä ketteriä ohjelmistokehitysmenetelmiä sekä erityisesti ketterien menetelmien onnistuneen hyödyntämisen menestystekijöitä eli onnistuneen hyödyntämisen edellytyksiä. Ketterät menetelmät -käsite on määritelty Agile Manifesto-julistuksessa (2001), jossa korostetaan neljää ketterien ohjelmistokehitysmenetelmien pääperusperiaatetta: panostamista yksilöihin ja kanssakäymiseen, toimivaan ohjelmistoon, asiakasyhteistyöhön ja muutokseen vastaamiseen (Agile Manifesto 2001). Ketteristä menetelmistä ja niiden hyödyntämisestä löytyy paljon kirjallisuutta, mutta esimerkiksi Dingsøyr, Nerur, Balijepally & Moe (2012) korostavat aiheeseen liittyvän kirjallisuuden epäselvyyttä ja kaipaavat yhdenmukaisuutta sen määrittelyyn, mistä syystä aiheen tutkiminen onkin hyvin mielekästä. Ketterien menetelmien onnistuneen hyödyntämisen edellytyksistä eli menestystekijöistä löytyy myös jonkin verran kirjallisuutta, mutta muun muassa Aldahmash, Gravell & Howard (2017) korostavat artikkelissaan ketterien menetelmien hyödyntämisen menestystekijöiden tutkimuskentän tarvetta laajentamiselle ja selventämiselle. Lisäksi myös Misra, Kumar & Kumar (2009) kertovat artikkelissaan ketterien menetelmien laajasta tutkimuksesta koskien erityisesti ketterien menetelmien implementoimista, mutta usein tutkimuksista uupuu ohjeistusta sille, millaisia edellytyksiä ketterien menetelmien onnistunut hyödyntäminen vaatii resursseilta, organisaatiolta ja projektitiimiltä. Tutkielman tutkimuskysymyksenä on siis:

- Mitkä ovat ketterän ohjelmistokehitysprojektin onnistumisen kriittiset menestystekijät?

Kriittisillä menestystekijöillä tarkoitetaan siis niitä onnistumisen edellytyksiä tai ominaisuuksia, jotka projektitiimillä tulisi ainakin olla käytössään, jotta ketterä ohjelmistokehitysprojekti olisi onnistunut. Tutkimusongelman ratkaisemisen alustukseksi tutkimuksessa tarkastellaan aiheeseen liittyvää tasokasta kirjallisuutta kirjallisuuskatsauksen muodossa. Kirjallisuuskatsauksen lisäksi itse tutkimusongelmaan pyritään löytämään vastauksia laadullisella tutkimusmenetelmällä haastattelemalla alalla työskenteleviä

kokeneita asiantuntijoita. Haastattelut toteutetaan teemahaastatteluiden muodossa, jotta vastauksista saataisiin mahdollisimman kattavia ja monisanaisia, ja koska projektin onnistumisen edellytysten tärkeys vaihtelee riippuen toteuttavasta organisaatiosta, projektin tarpeista ja vaatimuksista. Aldahmashin ym. (2017) mukaan ketterien menetelmien soveltaminen jää täysin organisaation vastuulle ja täten myös menestystekijöiden painotukset eriävät usein toisistaan. Tästä syystä laadullinen tutkimusmenetelmä ja erityisesti teemahaastattelut sopivat hyvin tutkimusmenetelmäksi. Aldahmashin ym. (2017) mukaan ketterien periaatteiden mukaan toteutettavien ohjelmistokehitysprojektien yhteisiä menestystekijöitä voidaan ryhmitellä monella eri tapaa, mutta hyväksi tavaksi on osoittautunut jako neljään eri päätekijään: teknisiin-, organisatorisiin-, prosessi-, ja ihmistekijöihin, joiden pohjalta tämän tutkielman haastattelurunko on suunniteltu. Haastattelujen tulosten analysointimenetelmänä käytetään teemoittelua. Tämän jälkeen haastattelujen analysoituja tuloksia ketterien menetelmien onnistuneen hyödyntämisen menestystekijöistä pyritään vertailemaan kirjallisuuskatsauksessa löydettyihin tutkimustuloksiin ja pyritään tekemään johtopäätöksiä, joiden avulla pyritään löytämään uutta tutkimustietoa tiedeyhteisön kannalta sekä luomaan lisäarvoa käytännön kannalta erityisesti kehittämällä ketterän ohjelmistokehityksen malli, johon organisaatiot voivat peilata toimintaansa ja tämän pohjalta kehittää omaa ketteryyttä sekä mahdollisuutta saavuttaa onnistuneita ohjelmistoprojekteja.

Tämä tutkielma on jäsennetty seitsemään eri lukuun. Luvussa 2 keskitytään ketterien periaatteiden määrittelyyn Ketterien periaatteiden julistuksen ja keskeisten kirjallisuuskatsausten pohjalta. Luvussa 3 käsitellään ketterän ohjelmistokehityksen periaatteita aiheeseen liittyvien kirjallisuuskatsausten pohjalta sekä esitellään neljä keskeistä ketterää ohjelmistokehitysmenetelmää: Scrum, Lean-ohjelmistokehitys, Extreme Programming(XP) ja Kanban. Luvussa 4 tarkastellaan, esitellään ja vertaillaan aikaisempia tutkimuksia liittyen ketterän ohjelmistokehityksen menestystekijöihin. Luvussa 5 esitellään tutkielmassa käytettävät tutkimusmenetelmät, perustellaan niiden käyttöä sekä esitellään tutkimusmenetelmänä käytettävän teemahaastattelun haastattelurunko. Tämän lisäksi luvussa 5 perustellaan myös haastateltavien valintaa, esitellään ja perustellaan teemahaastattelulla kerättävän aineiston analyysimenetelmää sekä perustellaan tutkimuksen

luotettavuutta. Luvussa 6 esitellään haastateltavat asiantuntijat sekä esitellään teemahaastatteluilla kerätyn aineiston teemoittelemalla analysoidut tulokset ja löydökset. Löydösten pohjalta luvussa 6 esitellään erityisesti käytännön avuksi ketterän ohjelmistokehityksen malli, johon ketterää ohjelmistokehitystä implementoivat organisaatiot voivat peilata omaa toimintaansa kehittyäkseen. Luvussa 7 havaintoja ja tuloksia verrataan aikaisempiin tutkimustuloksiin ja kirjallisuuskatsauksessa löydettyihin väitteisiin, ja tämän avulla arvioidaan tutkimuksen onnistumista. Luvussa 7 arvioidaan muun muassa kirjallisuuskatsaukseen vertaamisen kautta tutkimuksen merkitystä ja arvoa sekä teorian että käytännön kannalta.

2 Ketterän ohjelmistokehityksen määrittely

Ketterä ohjelmistokehitys katsotaan ensimmäisenä määritellyksi Agile Manifesto julistuksessa vuonna 2001, jonka jälkeen ketteryyttä ja ketterää ohjelmistokehitystä on määritelty monin eri tavoin ja tärkeysjärjestyksin lukuisissa eri tutkimuksissa. Ketterän ohjelmistokehityksen määrittely aiheeseen liittyvässä kirjallisuudessa ei siis ole täysin yksiselitteistä eikä myöskään yhdenmukaista. (Dingsøyr ym. 2012) Seuraavissa alaluvuissa tarkastellaan tarkemmin ketterän ohjelmistokehityksen käsitteen syntyä ja sen määrittelyä.

2.1 Ketterän ohjelmistokehityksen julistus

Yhdysvalloissa vuonna 2001 Agile Allianceksi itseään kutsuva ryhmä ohjelmistokehityksen asiantuntijoita julkaisi Agile Manifesto-julistuksen, jossa esitetään periaatteet ja suuntaviivat ketterälle ohjelmistokehitykselle. Agile Manifesto on myöhemmin hyvin vakiintunut termi aiheeseen liittyvässä kirjallisuudessa. Suomeksi Agile Manifestoa kutsutaan Ketterän ohjelmistokehityksen julistukseksi ja se on käännetty myös lukuisille muille eri kielille nettisivuillaan. (Agile Manifesto 2001)

Julistuksessa ketterän ohjelmistokehityksen periaatteet on tiivistetty neljään perusperiaatteeseen tai arvoon: ”Kokemuksen perusteella arvostamme: 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 Manifesto 2001). Julistuksessa mainitaan kuitenkin myös: ”Jälkimmäisilläkin asioilla on arvoa, mutta arvostamme ensiksi mainittuja enemmän” (Agile Manifesto 2001). Ketterän ohjelmistokehityksen julistuksen neljä perusperiaatetta on laajennettu 12 periaatteeseen, jotka on tarkemmin eritelty taulukossa alla (ks. taulukko 1).

Taulukko 1. Ketterän ohjelmistokehityksen arvot ja periaatteet (mukaillen Kinnunen 2015: 18).

Agile Manifeston perus-periaatteet	**Agile Manifeston 12 laajennettua periaatetta**
Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja	Liiketoiminnan edustajien ja ohjelmistokehittäjien tulee työskennellä yhdessä päivittäin
	Projektit rakennetaan motivoituneiden yksilöiden ympärille, joille tarjotaan puitteet ja tuki, jotta he saavat työn tehtyä
	Kasvokkain keskustelu on tehokkain tapa jakaa tietoa kehitystiimille ja sen sisällä
	Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat syntyvät itseohjautuvissa tiimeissä
	Tiimi arvioi säännöllisesti, kuinka parantaa tehokkuuttaan ja mukauttaa toimintansa sen mukaisesti
	Ketterät menetelmät kannustavat kestävään toimintatapaan ja saavutettu työtahti pitäisi pystyä ylläpitämään
Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota	Edistymisen ensisijainen mittari on toimiva ohjelmisto
	Oleellista on yksinkertaisuus eli tekemättä jätetyn työn maksimointi
	Teknisen laadun ja ohjelmiston hyvän rakenteen jatkuva huomioiminen edesauttavat ketteryyttä
Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja	Asiakkaan tyydyttäminen toimittamalla tämän tarpeet täyttäviä versioita ohjelmistosta aikaisessa vaiheessa ja säännöllisesti
	Versioita toimivasta ohjelmistosta toimitetaan säännöllisesti, parin viikon tai kuukauden välein
Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa	Muuttuvien vaatimuksien vastaanottaminen kehityksen myöhäisessä vaiheessa, ketterät menetelmät hyödyntävät muutosta asiakkaan kilpailukyvyn edistämiseksi

Kuten aiemmin jo mainittiin, Ketterän ohjelmistokehityksen julistus kertoi siis vain suuntaviivoja ketterän ohjelmistokehityksen tutkimukselle. Myös Abbas, Gravell & Wills (2008) kertovat artikkelissaan, että ketterän ohjelmistokehityksen mukaisia työskentelytapoja on käytetty jo vuonna 1957, mutta mainitsevat myös, että ketterästä ohjelmistokehityksestä puhuttiin ilmiönä vasta Ketterässä ohjelmistokehitysjulistuksessa. Kuitenkin Conboy (2009) mainitsee artikkelissaan yllä mainitun Ketterän ohjelmistokehityksen julistuksen olevan pikemminkin ketterän ohjelmistokehityksen mainostamista, eikä sen laajaa akateemista tutkimista ja pohtimista. Tämä mielessä pitäen on relevanttia tarkastella myös muita ketterään ohjelmistokehitykseen liittyviä määrittelyitä aiheeseen liittyvässä kirjallisuudessa, joihin keskitymme tarkemmin seuraavassa alaluvussa.

2.2 Ketterän ohjelmistokehityksen muita määrittelyjä

Ketterästä ohjelmistokehityksestä löytyi paljon tutkimuksia tätä tutkielmaa tehdessä ja määrittelyistä löytyi myös paljon eroja. Dingsøyrin ym. (2012) tutkimuksen mukaan osa ketterän ohjelmistokehityksen tutkimuksista ovat hyvin epäselviä ja ne kaipaavat paljon yhdenmukaisuutta ketterän ohjelmistokehityksen käsitteen määrittelyyn. Ketterä ohjelmistokehitys ei olekaan helppo määritellä eikä se ole käsitteenä kovin selvärajainen (Kinnunen 2015: 10–11).

Sana ketteryys rinnastetaan Suomen kielitoimiston sanakirjan (2018) mukaan seuraaviin adjektiiveihin: keveä, joustava, notkea, sukkela ja vikkelä. Erickson, Lyytinen ja Siau (2005) kuvailevatkin tutkimuksessaan ketterän ohjelmistokehityksen periaatteita paljolti samoin adjektiivein: ketteryys, taipuisuus, nopeus, eloisuus ja nopea reagointi. Käsitettä on siis edellä mainitun mukaisesti määritelty monin eri tavoin, mutta esimerkiksi Dingsøyrin ym. (2012) mukaan erityisesti Conboy (2009) määritteli kirjallisuuskatsauksessaan ketterän ohjelmistokehityksen käsitteen osuvimmin ja kattavimmin:

> Ohjelmistokehityksen metodin valmius tuottaa muutosta ennakoivasti, reagoivasti tai luonnostaan vastaanottaa ja hyväksyä muutos ja samalla oppia muutoksesta, lisätä asiakkaan kokemaa arvoa(taloudellisuus, laadullisuus ja

> yksinkertaisuus) hyödyntäen kehitysprojektin osia ja sen suhteita ympäristöön (Conboy 2009: 340).

Conboyn (2009: 331) tutkimuksessa ketteryys kiteytyy kahteen perusajatukseen: joustavuus (engl. flexibility) ja turhan poistaminen (engl. leanness). Monissa tutkimuksissa korostuu Conboyn (2009) tutkimuksen lopputuloksena kerrotun ketterän ohjelmistokehityksen määrittelyn tapaan muutokseen ja ympäristöön reagoimisen nopeus, kuten esimerkiksi Cohenin, Lindvallin & Costan (2004) kirjallisuuskatsauksessa.

Lukuisat tutkimukset, kuten Ketterän ohjelmistokehityksen julistus, Ericksonin ym. (2005), Conboyn (2009) sekä Dingsøyrin ym. (2012) kirjallisuuskatsaukset korostavat erityisesti muutoksen merkitystä ketterässä ohjelmistokehityksessä. Poth, Sasabe, Mas & Mesquida (2018) korostavat liiketoimintaympäristöjen ja sitä mukaa tilaajan tarpeiden ja vaatimusten muuttumista hurjaa vauhtia. Ketteryys nähdään toisin sanoen kykynä tasapainotella tarpeiden ja vaatimusten muuttuessa joustavuuden ja vakauden välillä toimimalla nopeasti, mukautuvasti ja itseohjautuvasti. Edellä mainitut ketterän ohjelmistokehityksen julistus ja kirjallisuuskatsaukset eivät korosta ketterässä ohjelmistokehityksessä ainoastaan muutokseen vastaamisen tärkeyttä vaan myös muutoksen luomista, siitä oppimista ja oppimisen kautta toiminnan parantamista. Myös Cockburn ja Williams (2003: 40) toteavat hyvin muutoksiin reagoimisen edellytyksien tärkeyden artikkelissaan; ohjelmistokehitysprojekti on epälineaarinen prosessi, eikä prosessin voida aina olettaa päätyvän samaan lopputulokseen, koska toimintaympäristössä tapahtuu niin paljon muutoksia. Kyseisessä artikkelissa kerrotaan myös liiketoiminnallista näkökulmaa korostaen, että ohjelmistokehitysprojektin toimintaympäristössä tapahtuvat muutokset tulisi nähdä pikemminkin uusina liiketoimintamahdollisuuksina eikä uhkina projektin etenemiselle. Tällaisia projektinaikaisia muutoksia voivat olla esimerkiksi edellä mainittu vaatimusten muuttuminen tai projektin työntekijöiden vaihtuminen (Cockburn ja Williams 2003: 40).

Suuressa roolissa sekä Ketterän ohjelmistokehityksen julistuksessa, Ericksonin ym. (2005), Conboyn (2009) ja Dingsøyrin ym. (2012) kirjallisuuskatsauksissa ovat myös itse tuotteen toimitusnopeus, asiakaslähtöisyys, läheinen ja suora kommunikaatio sekä myös ohjelmistokehitysprojektin toistava ja lisäävä luonne (engl. iterative and incremental

development) ketterää ohjelmistokehitystä määritellessä. Toistavalla ja lisäävällä luonteella tarkoitetaan toimintatapaa, jossa ohjelmaa kehitetään ja toimitetaan asiakkaalle pienin parannuksin, ja tilaajalta saadaan palautetta alituisesti projektin edetessä (Greer & Hamon 2011: Larman & Basili 2003).

Ericksonin ym. (2005), Conboyn (2009) ja Dingsøyrin ym. (2012) kirjallisuuskatsauksissa ketteryyttä rinnastetaan myös raskauden ja kankeuden vastakohdaksi, jonka esimerkkinä muun muassa Erickson ym. (2005) mainitsevat perinteiset ohjelmistokehitysmenetelmät, kuten esimerkiksi vesiputousmalli tai v-malli.

2.3 Ketterän ohjelmistokehityksen tiivistetty kuvaus

Ketterää ohjelmistokehitystä on siis määritelty monella eri tapaa lukuisissa eri tutkimuksissa. Yhdistäviä tekijöitä on kuitenkin havaittavissa runsaasti. On kuitenkin otettava huomioon, että ketteryyden määrittely vaihtelee paljon riippuen projektista ja sen luonteesta, kuten esimerkiksi siitä, mitä vaatimuksia ja tavoitteita projektilla on, kuinka iso projektitiimi on, millainen on sitä toteuttava organisaatio ja millaisia menetelmiä projektissa käytetään. (Dingsøyr ym. 2012; Greer & Hamon 2011; Kinnunen 2015) Yhteenvetona voidaan kuitenkin todeta, että ketteryys ohjelmistokehityksessä on kykyä luoda asiakkaalle arvoa vastaamalla asiakkaan muuttuviin tarpeisiin ja toimintaympäristön alati muuttuviin vaatimuksiin dynaamisesti, joustavasti, itseohjautuvasti ja asiakaslähtöisesti palvelemalla asiakasta ja toimittamalla tuoteversioita iteratiivisesti ja inkrementaalisesti, eli toistavan ja lisäävän luonteen mukaisesti (Conboy 2009: Larman & Basili 2003).

Tässä tutkielmassa pyritäänkin selvittämään, millaiset ovat ketterien menetelmien onnistuneen hyödyntämisen menestystekijät, eli optimaaliset edellytykset ketteryyden soveltamiseen ohjelmistokehitysprojektissa.

3 Ketterän ohjelmistokehityksen yhteiset periaatteet ja menetelmät

Tässä luvussa ketterän ohjelmistokehityksen keskeisimpiä periaatteita tarkastellaan tarkemmin, joita käsiteltiin jo alustavasti johdannossa ja tutkielman toisessa luvussa. Ketterän ohjelmistokehityksen yhteisten periaatteiden lisäksi tarkastelemme esimerkkinä myös neljää yleistä ketterää ohjelmistokehitysmenetelmää: Scrum, Lean-ohjelmistokehitys, Extreme Programming(XP) ja Kanban. On huomionarvoista mainita, että ketterän ohjelmistokehityksen periaatteet ovat kaikille menetelmille yhteisiä, kun taas ketterät ohjelmistokehitysmenetelmät ovat menetelmiä tai viitekehyksiä, joiden avulla ketterää projektinhallintaa voi harjoittaa.

3.1 Ketterän ohjelmistokehityksen yhteiset periaatteet

Johdannossa ja toisessa pääluvussa kävimme läpi Ketterän ohjelmistokehityksen julistuksessa mainitut neljä ketterän ohjelmistokehityksen perusperiaatetta. Ketterän ohjelmistokehityksen julistuksessa (2001) kyseiset neljä perusperiaatetta on laajennettu 12 periaatteeseen, jotka on tarkemmin eritelty toisen pääluvun taulukossa (ks. taulukko 1). Jokaisen näiden 12 periaatteen yksityiskohtainen läpikäynti ei kuitenkaan ole perusteltua, koska esimerkiksi Conboyn (2009) mukaan kyseinen Ketterän ohjelmistokehityksen julistus ei ole aiheen laajaa ja puolueetonta pohtimista, vaan pikemminkin ketterän ohjelmistokehityksen mainostamista. Ketterän ohjelmistokehityksen julistus on kuitenkin arvostettu julkaisu ja ensimmäinen julkaisu, jossa käsiteltiin ketterän ohjelmistokehityksen termiä. Ketterää ohjelmistokehitystä tutkivasta kirjallisuudesta voidaan kuitenkin havaita lukuisia yhteneväisyyksiä. Kinnunen (2015: 14) toteutti kattavan tutkimuksen ketterään ohjelmistokehitykseen liittyvästä kirjallisuudesta, jonka mukaan eniten mainittuina ketteryyden määrittelyinä hän löysi seuraavat periaatteet; joustavuus ja turhan poisto, muutoksiin reagoiminen, nopeus, asiakasyhteistyö, kasvokkain tapahtuva ja jatkuva kommunikaatio sekä kehitysprojektin toistava ja lisäävä luonne. Myös Cohenin ym. (2004), Cockburnin & Williamsin (2003) sekä Conboyn (2009) määritelmät ketterästä

ohjelmistokehityksestä perustuvat juuri näihin periaatteisin, joita tarkastelemme tarkemmin seuraavaksi.

Kuten edellä jo mainittiin, Dingsøyrin ym. (2012: 1) artikkelin mukaan ketterän ohjelmistokehityksen perustarkoituksena on luoda asiakkaalle arvoa toimittamalla toimiva ohjelmisto mahdollisimman nopeasti ja asiakaslähtöisesti. Conboyn (2009: 331) mukaan joustavuudella (engl. flexibility) ja turhan poistolla (engl. leanness) keskitytään nopeaan toimittamiseen, liiketoimintaympäristön muutoksiin vastaamiseen ja keskittymään vain asiakkaalle arvoa tuottaviin toimiin. Nopeasti toimittamisella pyritään vastaamaan toimintaympäristön ja vaatimusten alati muuttumiseen, koska teknologia kehittyy ja tarpeet muuttuvat hurjaa vauhtia. Asiakaslähtöisellä lähestymistavalla ja nopealla toimittamisella varmistetaan, että asiakkaalle toimitetaan vain aidosti lisäarvoa tuovaa palvelua, koska asiakkaalta saadaan välitöntä palautetta. (Cao, Mohan, Xu & Ramesh 2009: Conboy, 2009)

Muutoksiin reagoiminen, niihin vastaaminen ja muutoksen luominen nähdään laajalti aihetta tutkivassa kirjallisuudessa ketterän ohjelmistokehityksen kulmakiveksi. Sekä Ericksonin ym. (2005), että Conboyn (2009) mukaan ketterän ohjelmistokehityksen tarkoituksena on toimia dynaamisesti ja joustavasti, eli pyrkiä nopeasti reagoimaan toimintaympäristössä tapahtuviin muutoksiin ja vaatimuksiin, ja osata muuttaa toimintatapoja ja tavoitteita vaatimusten mukaan. Myös Cockburn ja Williams (2003: 40) korostavat artikkelissaan paljon muutoksiin reagoimista tärkeänä ketterän ohjelmistokehityksen periaatteena, ja artikkelin mukaan ketterän ohjelmistoprojektin tulisikin olla epälineaarinen ja empiirinen prosessi. Empiirisellä prosessilla ketterässä ohjelmistokehitysprojektissa tarkoitetaan käytännössä ”tarkasta ja sopeuta” -syklejä sekä lyhyitä ja useasti pidettäviä palautekeskusteluja sekä projektitiimin kesken, että asiakkaan kanssa. Näiden toimien ansiosta projektitiimin on helpompi ja nopeampi reagoida ja käsitellä toimintaympäristössä tapahtuvia muutoksia. (Cockburn ja Williams 2003: 40) Projektitiimin nopea ja reaktiivinen päätöksenteko vaatii projektitiimiltä itseohjautuvuutta, oikeanlaista organisaatiokulttuuria ja valtuuksia, joihin palaamme tarkemmin viidennessä pääluvussa. Mukautuvuus on siis ketterän lähestymistavan avainominaisuus, toisin kuin ennustettavuus,

joka taas mielletään perinteisen ohjelmistokehityksen avainominaisuudeksi (Poth ym. 2018).

Ohjelmistokehitysprojektin toistava (engl. iterative) ja lisäävä (engl. incremental) luonne yhdistettynä läheiseen ja suoraan kommunikointiin asiakkaan kanssa pienentää väärinymmärryksen ja turhan työn tekemisen riskiä. Kun kehitysprojektin tuotetta parannetaan toistavasti ja lisäävästi, projektitiimi keskittyy eniten asiakkaalle arvoa tuottaviin asioihin ja ominaisuuksiin. Toistavan ja lisäävän luonteen ansiosta projekti etenee tahdikkaasti ja asiakkaalta saadaan koko projektin ajan palautetta ominaisuuksista, ja projektin tuotosta voidaan kehittää asiakkaan haluamaan suuntaan. Ohjelmistokehitysprojektin toistava ja lisäävä luonne on näin ollen hyvin oleellinen osa liiketoimintaympäristön muutoksiin vastaamista ja niihin reagoimista. (Conboy, 2009; Greer & Hamon, 2011)

Dingsøyr (2012), Poth ym. (2018) sekä Greer & Hamon (2011) korostavat tutkimuksissaan erityisesti suoran kommunikaation, nopean päätöksentekokyvyn ja läheisen asiakasyhteistyön merkitystä muutoksiin vastaamisessa ja niihin reagoimisessa. Itseohjautuva tiimi, matalahierarkkinen organisaatiokulttuuri sekä kasvokkain tapahtuva läheinen ja suora kommunikaatio ovat avainasemassa, kun projektin vaatimuksissa tapahtuu muutoksia ja projektitiimin täytyy kyetä tekemään päätöksiä nopeasti ja dynaamisesti. Asiakasyhteistyön merkitystä korostetaan myös paljon, koska asiakkaan osallistuessa kehitysprojektiin projektitiimi saa asiakkaalta koko kehitysprojektin ajan välitöntä palautetta ja informaatiota asiakkaan tarpeista. Näin projektissa voidaan keskittyä aidosti asiakasarvon luontiin. Lisäksi muutokset asiakkaan tarpeissa ja vaatimuksissa sekä asiakkaan omassa liiketoimintaympäristössä kantautuvat välittömästi projektitiimin korviin. Läheinen ja suora kommunikaatio on avainasemassa luonnollisesti myös asiakasyhteistyössä. (Cobb, 2011; Dingsøyr, 2012; Poth ym. 2018)

3.2 Ketterän ohjelmistokehityksen menetelmiä

Ketterän ohjelmistokehityksen julistukseen, muihin edellä mainittuihin ketterän ohjelmistokehityksen määrittelyihin sekä seuraavaksi mainittaviin neljään ketterän ohjelmistokehityksen menetelmään liittyy paljon yhteneväisiä periaatteita ja tapoja, joiden avulla

ohjelmistokehityksestä pitäisi tulla ketterää. Aiheeseen liittyvän kirjallisuuden mukaan suosituimmat ketterän ohjelmistokehitysmenetelmät ovat Scrum, Lean-ohjelmistokehitys, Extreme Programming ja Kanban, joita tarkastellaan seuraavissa luvuissa. Näiden menetelmien yksityiskohtainen vertailu ei kuitenkaan tämän tutkimuksen puitteissa ole perusteltua, koska kokemukset niiden soveltamisesta ovat usein hyvin subjektiivisia.

Ketterät ohjelmistokehitysmenetelmät, niiden periaatteet ja käytänteet tarjoavat siis ohjeita ja käytänteitä saavuttamaan ketteryyttä, mutta itse menetelmien käyttöönotto ja soveltaminen jäävät yrityksen vastuulle. On myös huomionarvoista, että todellisuudessa totuus ei menetelmien suhteen ole niin yksiselitteinen; yritysten ohjelmistokehityksessä sovelletaan hyvin usein eri mallien käytänteiden yhdistelmiä ja mukana voi olla käytänteitä myös perinteisistä ohjelmistokehitysmenetelmistä. Kinnusen (2015) tutkimuksen mukaan suurin osa ketterien menetelmien käytänteiden sovelluksista ohjelmistokehityksessä ovat XP:hen tai Scrumin käytäntöihin rinnastettavia. Seuraavaksi tarkastellaan neljää yllämainittua ohjelmistokehitysmenetelmää, jotka ovat Scrum, Lean-ohjelmistokehitys, Extreme Programming (XP) ja Kanban.

3.2.1 Scrum

Scrum on ketteristä ohjelmistokehitysmenetelmistä käytetyin. Scrum on ohjelmistokehitysmenetelmänä niin suosittu, että se rinnastetaan joissain kirjallisuuksissa itse ketteryyteen (Poppendieck ja Cusumano 2012: 30). Vaikka monissa yhteyksissä virheellisesti luullaan, Scrum ei ole ainoastaan ohjelmistokehitysmenetelmä, vaan Scrumin kehittäjien Jeff Sutherlandin ja Ken Schwaberin Scrum-käsikirjan (2017: 20) mukaan se tulisi nähdä enemminkin viitekehyksenä, ja sitä sovelletaan laajasti eri toimialoilla kehittämään esimerkiksi ohjelmistoja, sulautettuja järjestelmiä, itseohjautuvia ajoneuvoja, kouluja tai markkinointia. Scrum-viitekehykseen luetaan Scrum-tiimit rooleineen, tapahtumineen, tuotoksineen ja sääntöineen. Jokaisella näillä Scrumin elementillä on omat tarkoituksensa, ja ne ovat Scrumin oleellisia osia. Kehittäjät määrittelevät Scrumin luonteen kevyeksi, yksinkertaiseksi ymmärtää, mutta suhteellisen vaikeaksi toteuttaa lukuisten sääntöjensä ja elementtiensä takia. (Schwaber ja Sutherland 2017: 4)

Scrumin keskeisimpinä periaatteina on työskennellä ketterän ohjelmistokehityksen periaatteiden tavoin läpinäkyvästi, luovasti, joustavasti ja tuotteen kehittämisen tulisi edetä toistavasti ja lisäävästi, ja keskeisenä Scrumin menetelmänä ovatkin pyrähdykset (engl. sprint), joista lisää tämän otsikon alakappaleessa. Myös itseohjautuvuus ja matalahierarkkisuus ovat suuressa roolissa Scrumia määritellessä, koska Scrum-tiimin tulee kyetä tekemään päätöksiä dynaamisesti, joten kaiken tarvittavan osaamisen ja päätöksentekokyvyn tulisikin löytyä Scrum-tiimin sisältä. (Kniberg & Skarin 2010; Schwaber ja Sutherland 2017: 4) Seuraavissa alakappaleissa käsittelemme kolmea tärkeää Scrumin viitekehyksen elementtiä; rooleja, tapahtumia ja tuotoksia.

3.2.1.1 Scrumin roolit

Scrum-ohjelmistokehitysmenetelmän toteuttamisen säännöt ja käytännöt ovat suhteellisen tarkasti määriteltyjä, joten myös roolit ovat tarkasti määriteltyjä. Scrum-viitekehyksen mukaan toteutetussa projektissa toimii yksi tai useampi Scrum-tiimi projektin koon mukaan, ja jokaiseen Scrum-tiimiin kuuluu tuoteomistaja, Scrum-master sekä kehitystiimi. (Schwaber ja Sutherland 2017: 6-9; Sutherland 2010: 14)

Tuoteomistaja vastaa nimensä mukaisesti tuotteen ja kehitystiimin työn arvon maksimoimisesta sekä määrittelee tuotteen vaatimukset ja suunnittelee suoritettavat tehtävät yhdessä koko Scrum-tiimin kanssa. Tuoteomistaja on siis vain yksi henkilö, jonka tulee tuntea tuotteen liiketoiminta sekä seurata projektin edistymistä taloudellisin tunnusluvuin, kuten esimerkiksi projektiin sijoitetun pääoman tuottoprosentin avulla. Näin ollen tuoteomistajan täytyy kyetä seuraamaan ja osin hallitsemaan projektin etenemistä sekä projektinjohdollisesta, että liiketoiminnallisesta näkökulmasta (Hart 2011: 2). Tuoteomistajan rooli on hieman valvojamainen, ja tuoteomistajan tulee seuraa projektin etenemistä tarkasti ja varmistaa kullakin hetkellä, että Scrum-tiimi toteuttaa jokaisessa pyrähdyksessä toimia, jotka ovat keskeisimpiä kullakin hetkellä. (Schwaber ja Sutherland 2017: 7; Sutherland 2010: 14) Tutkimuksessaan Moe, Dingsøyr ja Dybå (2010) korostavat kuitenkin, että Scrum-tiimin ideana on olla matalahierarkkinen ja päätökset tehdään

operatiivisella tasolla yhdessä koko Scrum-tiimin kanssa, eikä esimerkiksi yksin tuoteomistajan toimesta.

Scrum-masterin tehtävänä on vastata Scrumin edistämisestä, tukemisesta ja pitämällä huolta, että kaikki Scrum-tiimin jäsenet ymmärtävät Scrumin teorian, käytännöt ja säännöt (Schwaber ja Sutherland 2017: 8-9). Sutherland (2010: 16-17) kuvailee Scrum-masteria palvelevana johtajana, joka työskentelee tiiviisti sekä tuoteomistajan, kehitystiimin, että koko organisaation kanssa. Scrum-master varmistaa, että koko Scrum-tiimi ymmärtää projektin tavoitteet, priorisoinnin kussakin pyrähdyksessä ja keskeisimmät ohjelmiston toiminnallisuudet sekä ehdottaa ja neuvoo tekniikoita kehitysjonon hallitsemiseen. Scrum-master siis mahdollistaa, kehittää ja valmentaa koko Scrum-tiimiä toimimaan itseohjautuvasti, mahdollisimman tuottoisasti ja saavuttamaan keskeisimmät tavoitteet sekä koko projektin, että yksittäisen pyrähdyksen osalta. Lisäksi Scrum-master myös suunnittelee Scrumin toteutusta ja käyttöä lisäksi myös organisaation sisällä, johon kuuluu myös tiivis yhteistyö muiden Scrum-mastereiden kanssa. (Schwaber ja Sutherland 2017: 8-9; Sutherland 2010: 16-17) Huomionarvoista Scrum-masterin tehtäviin liittyen on muistaa, että Scrum-viitekehystä voidaan soveltaa monella eri tavalla, ja tavat vaihtelevatkin paljon toimintaympäristöstä, resursseista ja tuotteesta riippuen (Sutherland 2010: 16-17). Hartin (2011: 2) mukaan Scrum-masterin ja tuoteomistajan tulisi täydentää toistensa työpanoksia.

Kehitystiimin jäsenten tulisi olla ammattimaisia ja kyetä työskentelemään itseohjautuvasti ja joustavasti. Tiimin sisällä ei ole erilaisia alitiimejä, vaan koko kehitysjonon vastuu painottuu koko kehitystiimille. Kehitystiimin kaikkien jäsenten on siis tarkoitus kyetä suorittamaan koko kehitysjonon kaikkia tehtäviä, kuten esimerkiksi koodausta tai testausta. Tällä ehkäistään muun muassa pullonkaulojen syntymistä. Sutherland (2010: 15) kuitenkin kertoo, että kehitystiimin jäsenillä voi kuitenkin olla erilaisia painotuksia esimerkiksi integraatioon tai testaukseen riippuen osaamisesta. Kehitystiimin jäsenten määrä on optimaalinen pienimmillään, jotta sen keskeisin periaate, ketteryys, tulee parhaiten esiin. Huomionarvoista on merkittävän työn aikaan saaminen, joka ei onnistu liian pienellä kehitystiimillä. Pienempi kuin kolmen henkilön kehitystiimi vähentää vuorovaikutusta eivätkä tuottavuushyödyt ole tarpeeksi isot, mutta toisaalta taas suurempi kuin yhdeksän

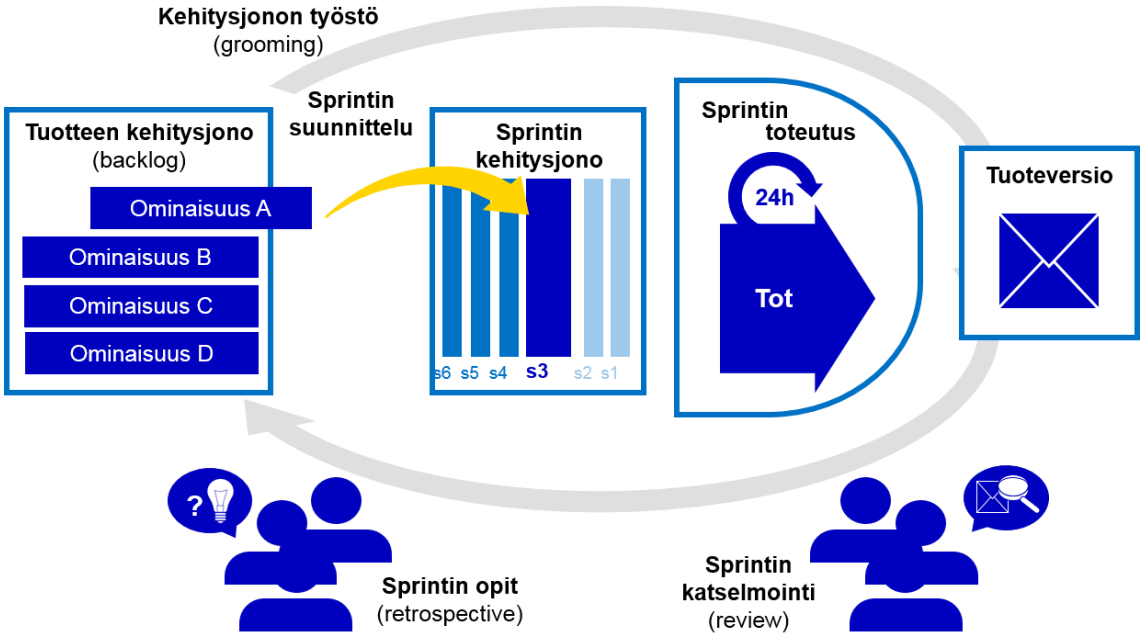
henkilön tiimi vaatii liikaa koordinointia. Huomionarvoista on myös se, että Scrum-master ja tuoteomistaja voivat kuulua kehitystiimiin, mikäli he osallistuvat tuotteen kehitysjonoon. (Schwaber ja Sutherland 2017: 15-16)

3.2.1.2 Scrumin tapahtumat

Scrumin tapahtumat ovat tarkasti ennalta määritettyjä luomaan säännöllisyyttä projektin etenemiselle ja minimoimaan muiden kuin Scrum-palavereiden tarve, jotta keskittyminen olennaiseen säilyy. Sutherland (2010: 6) ohjeistaa käsikirjassaan kaikkien kyseisten tapahtumien olevan välttämättömiä ja tapahtumille on asetettu maksimipituudet, kuinka paljon aikaa niihin on kannattavaa käyttää. Scrumin tapahtumien ytimenä on pyrähdys, joka kiteyttää Scrumin toistavan ja lisäävän luonteen, toisin sanoen jokaisen pyrähdyksen tuloksena tulisi siis olla potentiaalisesti julkaisukelpoinen tuoteversio. Pyrähdyksien pituudet säilyvät samana koko Scrumin ajan ja yhden pyrähdyksen pituudeksi on määritelty enintään kuukausi. (Schwaber ja Sutherland 2017: 9-14; Sutherland 2010) Huomionarvoista on mainita, että pyrähdys on myös mahdollista keskeyttää tuoteomistajan toimesta, mikäli pyrähdyksen tavoite muuttuu tarpeettomaksi. Keskeyttäminen kuluttaa resursseja ja nähdään harmillisena kokemuksena Scrum-tiimille ja vaatii tiimin uudelleenjärjestäytymisen uutta pyrähdystä varten, joiden takia pyrähdysten keskeyttämiset ovat hyvin harvinaisia. (Schwaber ja Sutherland 2017: 10)

Kukin pyrähdys aloitetaan koko Scrum-tiimin kesken suunnittelupalaverilla (engl. sprint planning meeting), jossa suunnitellaan pyrähdyksen aikana tehtävä kehitysjono. Pyrähdyksen aikana pidetään myös päivittäin enintään 15 minuutin mittaisia seisten toteutettavia päivittäispalavereita (engl. daily scrum), joissa kukin Scrum-tiimin jäsen kertoo edellisen päivittäispalaverin jälkeen tuotetut työpanokset ja luodaan suunnitelma seuraavalle 24 tunnille ja optimoidaan työpäivän arvo. Tuotteen kehitysjonon työstössä (engl. product backlog grooming) suunnitellaan yhdessä pyrähdyksen kehitysjonon yksityiskohdista ja tuotteen toiminnallisuuksista kyseisen pyrähdyksen aikana. Kehitysjonon työstön jälkeen on aika itse pyrähdyksen kehitystyölle, jossa kullakin Scrum-tiimin jäsenellä on siis selkeä toteutettavien tehtävien tehtävälista, jota ei enää kehitystyön aikana

voida muuttaa. Pyrähdyksen lopussa pidetään pyrähdyksen katselmointi (engl. sprint review), jossa tarkastellaan Pyrähdyksen aikaansaannos koko Scrum-tiimin ja Scrumin sidosryhmien voimin. Pyrähdyksen lopussa pidetään myös retrospektiivi, jossa Scrum-tiimi tarkastelee yhdessä työskentelyään ja jossa suunnitellaan yhdessä parannuksia työskentelyn tuottavuuden maksimoimiseksi. Kaikilla edellä mainituilla tapahtumilla on aikarajansa luonnollisesti suhteutettuna koko pyrähdyksen pituuteen, ja erityisesti palaverit pyritään pitämään mahdollisimman lyhyinä ja tuottavina. (Kniberg & Skarin 2010: 13-18; Schwaber ja Sutherland 2017; Sutherland 2010) Alla olevassa kuviossa on kuvattu pyrähdyksen tapahtumat tiivistettynä (ks. kuvio 1).



Kuvio 1. Scrumin roolit, tapahtumat ja tuotokset (mukaillen Sutherland 2010: 11).

3.2.1.3 Scrumin tuotokset

Scrum perustuu ennalta mainitun mukaisesti läpinäkyvyyteen, joustavuuteen ja ketteryyteen (Schwaber ja Sutherland 2017). Scrumin tuotokset on suunniteltu erityisesti läpinäkyvyyden maksimoimiseksi, jotta kaikilla projektiin liittyvillä on yhdenmukainen käsitys tuotteesta ja julkaisukelpoisesta inkrementistä eli tuoteversiosta kunkin

pyrähdyksen jälkeen. Scrumin neljä tuotosta ovat tuotteen kehitysjono, pyrähdyksen tehtävälista, edistymiskäyrä ja julkaisukelpoinen tuoteversio, joita tarkastelemme tarkemmin seuraavaksi.

Tuotteen kehitysjono on Sutherlandin (2010: 18-20) Scrum-käsikirjan sanoin järjestetty lista kaikesta, mitä tuotteessa tiedetään tarvittavan sekä ainoa lähde tuotteeseen toteutettaville vaatimuksille ja muutoksille. Kuten aiemmin mainittiin, tuoteomistajan rooliin kuuluu vastata tuotteen kehitysjonosta sekä sen sisällön, saatavuuden että järjestämisen suhteen. Tuotteen kehitysjonoa kutsutaan alati eläväksi dokumentiksi, koska ketterän ohjelmistokehitysperiaatteiden mukaisesti muutoksiin tulee varautua ja tämän takia mukaisesti kehitysjonoa muutetaan tarvittaessa. Tuotteen kehitysjonon elinkaari on yhtä pitkä kuin tuotteen elinkaari, joten kehitysjono ei siis ole vain tuotteen luomiseen tarkoitettu tehtävälista, vaan kehitysjonossa on mukana myös ylläpidollisia toimia ja parannuksia. (Schwaber ja Sutherland 2017: 15; Sutherland 2010: 18-20)

Pyrähdyksen kehitysjono on tehtävälista pyrähdyksen suunnittelupalaverissa valituista pyrähdyksessä toteutettavista kehitysjonon kohdista. Pyrähdyksen tehtävälista tekee Sutherlandin (2010: 20-24) mukaan näkyväksi kaiken sen työn, joka suunnittelupalaverissa sovitun tavoitteen saavuttamiseksi tarvitaan. Pyrähdyksen kehitysjono sisältää tyypillisesti vähintään yhden hyvin tärkeän parannuksen tuoteversioon, joka on tunnistettu edellisen pyrähdyksen retrospektiivissä. Pyrähdyksen kehitysjono on hyvin yksityiskohtainen tehtävälista, jotta työn edistyminen voidaan havaita päivittäispalavereissa ja pyrähdyksen kehitysjonoa on myös mahdollista muuttaa kehitystiimin toimesta sen mukaan, mikä on parasta pyrähdyksen tavoitteen saavuttamiseksi. (Schwaber ja Sutherland 2017: 16; Sutherland 2010: 20-24)

Pyrähdyksen edistymistä seurataan tai kuvataan edistymiskäyrällä, jossa aika on suhteutettuna pyrähdyksen kehitysjonon tehtävien toteutumiseen. Edistymiskäyrältä voidaan selkeästi nähdä, kuinka suuri on työmäärä suhteessa pyrähdyksessä jäljellä olevaan aikaan. (Schwaber ja Sutherland 2017: 16)

Tuoteversio voidaan nähdä konkreettisimpana Scrumin tuotoksena. Tuoteversio on siis kaikkien tuotteen kehitysjonon kohtien summa, jotka ovat valmistuneet pyrähdyksen ja

sitä aiempien pyrähdyksien aikana. Kunkin pyrähdyksen lopussa siinä valmistuneen tuoteversion tulee täyttää valmiin kokonaisuuden määritelmä ja olla käyttökelpoisessa kunnossa. (Schwaber ja Sutherland 2017: 17)

3.2.2 Lean-ohjelmistokehitys

Lean-johtamistapa on ketterä johtamistapa, jonka ytimessä on turhan työn minimoiminen ja olennaiseen keskittyminen (Poppendieck ja Cusumano 2012: 1; Wang, Conboy & Cawley 2012). Lean-käsite on johdettu englannin kielen sanasta turhan poisto ja Lean-ajattelun mukaisesti on Conboyn (2009: 334) mukaan valmistettu esimerkiksi Toyota-merkkisiä autoja jo 1950-luvulla ja lentokoneita toisen maailmansodan aikana. Lean-ajattelulla on pitkä historia monilla eri toimialoilla, kuten esimerkiksi tuotekehityksessä, valmistuksessa, terveydenhuollossa ja rakentamisessa (Poppendieck ja Poppendieck 2003: 21). Lean-ajattelu kertoo periaatteita, joita yritysten tulisi soveltaa toimintaympäristön asettamien vaatimusten ja projektin tavoitteiden puitteissa keskittyäkseen projektissa olennaiseen ja minimoidakseen turhaa työtä (Petersen & Wohlin 2001: 8). Keskeisimmät periaatteet on jaettu seitsemään periaatteeseen: eliminoi hukka, vahvista jatkuvaa oppimista, kehitä sisäistä laatua, toimita nopeasti, sitouta kaikki, panosta jatkuvaan parantamiseen ja optimoi kokonaisuus, joihin keskitymme tarkemmin seuraavissa luvuissa ohjelmistokehityksen kontekstissa. (Poppendieck ja Poppendieck 2003: 25; Poppendieck ja Cusumano 2012: 3; Wang ym. 2012: 6)

Eliminoi hukka -käsitteellä (engl. eliminate waste) tarkoitetaan kaiken sellaisen työn välttämistä, mikä ei luo suoraan arvoa asiakkaalle tai lisää tietoisuutta siitä, kuinka nopeuttaa omaa työskentelyä. Ohjelmistokehityksen kontekstissa merkittävimpiä arvoa luomattomia toimia ovat esimerkiksi tarpeettomat toiminnallisuudet ohjelmistossa ja huonosti tai puoliksi tuotettu työpanos. (Poppendieck ja Cusumano 2012: 3)

Panostaminen jatkuvaan parantamiseen (engl. keep ketting better) tarkoittaa työpanoksen jatkuvaa parantamista kaikilla projektin osa-alueilla. Toisin sanoen työntekijöiden pitäisi tehdä entistä parempia päätöksiä ja ratkaista ohjelmistokehityksessä kohdattavat ongelmat entistä paremmin. (Poppendieck ja Cusumano 2012: 3)

Sisäisen laadun kehittämisellä (engl. build quality in) tarkoitetaan jokaisen ohjelmiston moduulin laatuun panostamisella. Ohjelmiston tulisi säilyttää erinomaisen käytettävyytensä pitkänkin ajan kuluttua, mahdollistaa ylläpidon ja laajentamisen sekä ohjelmiston arkkitehtuurin tulisi olla yhdenmukainen. (Poppendieck ja Cusumano 2012: 3)

Jatkuva oppiminen (engl. learn constantly) tarkoittaa ohjelmistokehitysprojektin aikana tapahtuvaa oppimista. Työntekijöiden tulisi kartoittaa alituisesti eri vaihtoehtojen hyviä ja huonoja puolia kunnes oppia on kertynyt mahdollisimman paljon, ja päättää sitten vaihtoehtojen välillä mahdollisimman myöhään. Oppimista tulisi tapahtua myös tuotteen ominaisuuksien suhteen; ohjelmiston kehittäminen voidaan aloittaa vähimmäisvaatimuksista ja parantaa ja laajentaa sitten ominaisuuksia erityisesti asiakaspalautteiden perusteella. (Poppendieck ja Cusumano 2012: 4)

Nopeasti toimittaminen (engl. deliver fast) tarkoittaa Poppendieckin ja Poppendieckin (2003: 70) mukaan Lean-ajattelun mukaisesti tuotoksen toimittamista mahdollisimman nopeasti sekä myös useasti, jopa viikoittain tai päivittäin projektin aikana. Myös Poppendieck ja Cusumano (2012: 4) korostavat Lean-ohjelmistokehityksen oppien mukaan tuotetun ohjelmistoprojektin määrittelemistä pikemminkin jatkuvaksi tuotoksien virraksi, eikä nimenomaan perinteisenä projektina, jossa tuote on käyttökelpoinen vasta projektin päätyttyä.

Kaikkien sitouttamisella (engl. engage everyone) tarkoitetaan kaikkien projektiin osallistuvien sitouttamista. Ohjelmistokehitysprojektin toimiminen parhaalla mahdollisella tavalla vaatii vain IT-osaston sitoutumisen sijasta sitoutunutta yrityskulttuuria koko organisaation tasolla, koska Lean-ajattelun mukaisesti tuotoksia tulisi julkaista jatkuvana virtana. Lean-ohjelmistoprojekti tulisi siis nähdä ikään kuin koko organisaation liiketoiminnan pienoisversiona. Lean-ohjelmistoprojektiin tulisi sitouttaa niin ikään asiakkaat, suunnittelijat, ohjelmistokehittäjät, testaajat, myyjät ja jopa talousosaston työntekijät. (Poppendieck ja Cusumano 2012: 4)

Optimoi kokonaisuus -käsitteen (engl. optimize the whole) ytimessä on asiakaslähtöisyys, koska kehitettävä ohjelmisto tulisi olla osa suurempaa kokonaisuutta, koska asiakkaan tilaaman ohjelmiston arvo liittyy yleensä laajempaan liiketoimintakokonaisuuteen.

Esimerkiksi toimivan verkkokaupan ohjelmistoa kehittäessä pitäisi ottaa huomioon myös asiakaspalautteiden vastaanottamisen helppous. Ohjelmiston kehittämisessä pitäisi siis keskittyä isompaan kokonaisuuteen, kuin vain esimerkiksi ohjelmistokoodiin tai johonkin yhteen ainoaan toiminnallisuuteen yrityksen nettisivuilla. (Poppendieck ja Cusumano 2012: 3)

3.2.3 Extreme Programming

Extreme Programming-ohjelmistokehitysmenetelmän eli XP:n teki tunnetuksi Kent Beck vuonna 1999 teoksellaan Extreme Programming Explained: Embrace Change. Extreme Programming on ensimmäisiä ketterän ohjelmistokehityksen menetelmiä, joka julkaistiin 1990-luvun lopulla ja sitä pidetään eräänlaisena ketterän ohjelmistokehityksen alkupisteenä. (Abrahamsson ym. 2002; Beck 1999a; Beck 2001) Extreme Programming (lyhennettynä XP) on myös monien muiden ketterien ohjelmistokehitysmenetelmien tavoin toistava ja lisäävä, ja sen keskeiset periaatteet on jaettu neljään osaan: viestintä, yksinkertaisuus, palaute ja rohkeus, ja sen toiminnot on jaettu myös neljään osaan: ohjelmointi, testaus, kuuntelu ja virheiden korjaus. Kyseiset periaatteet ja toimet johtavat yhteensä 12 eri toimintatapaan, jotka onnistuneessa XP:llä toteutetussa projektissa tulisi toteuttaa. (Cao ym. 2009; Cohen ym. 2004: 13) Kyseisiin 12 toimintatapaan paneudutaan tarkemmin alla.

Suunnittelupeli (engl. the planning game) on jokaisen iteraation eli syklin alussa pidettävä palaveri, johon osallistuvat asiakkaat, johtajat ja kehittäjät priorisoimaan vaatimuksia syklin tuotosta varten. Tuotoksen vaatimuksia kutsutaan käyttäjätarinoiksi ja ne kirjoitetaan ylös tarinakortteihin. Suunnittelupalaveri nähdään siis eräänlaisena pelinä, jossa on tarkoituksena ideoida ja kirjoittaa ylös syklin tärkeimmät vaatimukset mahdollisimman asiakaslähtöisesti ajateltuna. (Cohen ym. 2004: 13)

Pienet julkaisut -käsite (engl. small releases) viittaa Extreme Programmingin toistavaan ja lisäävään luonteeseen, ja tuotoksia julkaistaan tiheään tahtiin jopa muutaman päivän tai viikon välein (Cohen ym. 2004: 13).

Metaforalla eli vertauskuvalla (engl. metaphor) pyritään ohjelmistoa suunnitellessa ja vaatimuksia kartoittaessa suunnittelemaan ohjelmisto ja sen käytettävyys vertauskuvien avulla (Cohen ym. 2004: 13). Metaforien käyttö ohjelmiston vaatimuksia kuvailtaessa helpottaa huomattavasti esimerkiksi suurten kokonaisuuksien eri toiminnallisuuksien hahmottamista (Cohen ym. 2004: 39).

Yksinkertainen suunnittelu (engl. simple design) tarkoittaa sitä, että kehittäjien tulisi pitää suunnittelu mahdollisimman yksinkertaisena, jotta toteuttaminen olisi helpompaa (Cohen ym. 2004: 13).

Testausvetoisuudella (engl. tests) Extreme Programming -menetelmässä pyritään kiinnittämään mahdollisimman paljon huomiota testaukseen ja sitä kautta ohjelmiston käytettävyyteen. Kehittäjät siis kirjoittavat testauskoodit ennen varsinaista ohjelmistokoodia sekä myös asiakkaat testaavat ohjelmiston toimintaa jokaisen kehityssyklin jälkeen. (Cohen ym. 2004: 13)

Lähdekoodin selkeyttäminen (engl. Refactoring) on prosessi, jossa ohjelmiston lähdekoodia muutetaan selkeämmäksi, mutta kuitenkin niin, että koodin toiminnallisuus ei muutu. Kyseisellä prosessilla ei siis pyritä esimerkiksi korjaamaan virheitä tai lisäämään ominaisuuksia, vaan sillä pyritään parantamaan esimerkiksi koodin luettavuutta (Cohen ym. 2004: 13).

Pariohjelmointi (engl. pair programming) tarkoittaa nimensä mukaisesti ohjelmoimista pareittain, jolloin mahdolliset virheet ja puutteet huomataan helpommin (Cohen ym. 2004: 13).

Jatkuvalla integraatiolla (engl. continuous integration) tarkoitetaan uuden ohjelmistokoodin integroimista ohjelmistoon niin usein kuin mahdollista. Jatkuvan integraatio -periaatteen mukaisesti ohjelmiston tulisi läpäistä toiminnalliset testit uuden koodin lisäämisen jälkeen tai uusi koodi on hylättävä. (Cohen ym. 2004: 13)

Yhteinen omistajuus (engl. collective ownership) tarkoittaa, että kaikki kehittäjät omistavat koodin. Toisin sanoen jokaisella kehittäjällä on vapaus muuttaa ohjelmistokoodia koska vain, jos he näkevät sen välttämättömäksi ohjelmiston toiminnallisuuden kannalta. (Cohen ym. 2004: 13)

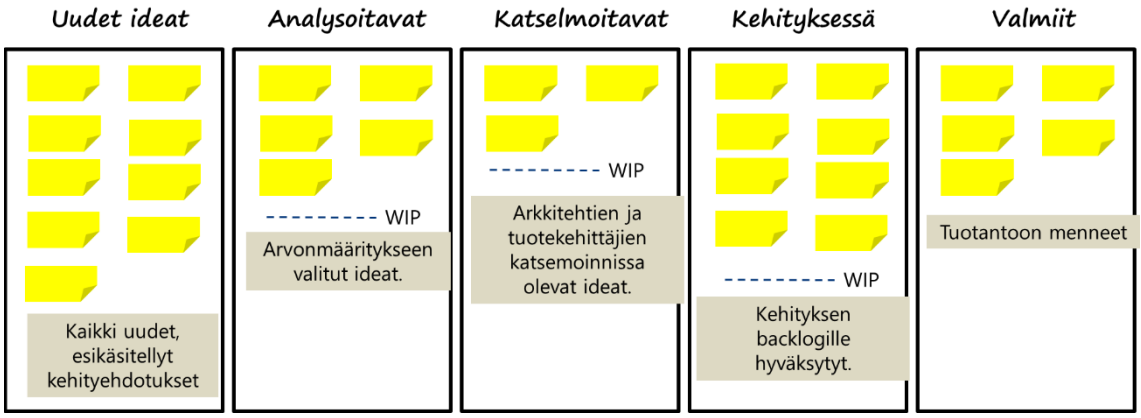
Asiakkaan läsnäolo (engl. on-site customer) ohjelmistokehitysprojektissa mahdollistaa välittömän asiakaspalautteen saamisen ja asiakastyytyväisyyden varmistamisen (Cohen ym. 2004: 14).

40-tuntisella työviikolla (engl. 40-hour weeks) varmistetaan vaatimusten valitseminen iteroinnin suunnitteluvaiheessa jokaista iterointia varten niin, että kehittäjien ei tarvitsisi tehdä ylitöitä ohjelmistoprojektissa. Tällöin esimerkiksi projektin aikataulun suunnittelu on myös helpompaa ja työntekijöiden jaksaminen parempaa säännöllisen työskentely-rytmin ansiosta. (Cohen ym. 2004: 14)

Jaetulla työtilalla (engl. open workspace) mahdollistetaan kehittäjien parempi kommunikointi ohjelmistokehitysprojektissa. Käytännössä jaettu työtila tarkoittaa työskentelyä yhteisessä työtilassa (Cohen ym. 2004: 14).

3.2.4 Kanban

Kanban on alun perin tuotantoteollisuudessa hyväksikäytetty toimintatapa, jossa projektin arvoketju kuvataan taululle, jossa yksittäiset tehtävät liikkuvat järjestyksessä kehityksen vaiheesta toiseen. Kanban-ajattelun mukaisesti tuotteen arvoketju nähdään putkena ja se perustuu ajatukseen, jossa vältetään pullonkauloja, turhaa työtä ja pyritään pitämään työn virta mahdollisimman tasaisena. Taululle kuvaamisen tarkoituksena on siis pitää kehitysprojektiin osallistuvat tietoisina eri projektin vaiheissa, mikä työ on kullakin hetkellä välttämätöntä, ja mikä työ synnyttäisi pullonkauloja (ks. kuvio 2). (Anderson 2010: 4; Poppendieck ja Cusumano 2012; Peterson 2015)



Kuvio 2. Esimerkki Kanban-taulusta (mukaillen Peterson 2015).

Esimerkki ohjelmistokehitysprojektin arvoketjussa olevasta pullonkaulasta olisi esimerkiksi tilanne, jossa ohjelmistokehittäjät kehittävät kymmenen toiminnallisuutta viikossa, mutta testaajat kykenevät testaamaan vain viisi toiminnallisuutta viikossa. Kyseisessä tilanteessa koko prosessin tuotos olisi siis vain viisi testattua toiminnallisuutta. (Peterson 2005)

Kanban on suhteellisen helposti laajennettavissa suurempaankin kokonaisuuteen, esimerkiksi ohjelmistoprojektin arvoketjun kuvaamiseen voidaan yhdistää myös markkinoinnillisia tai ohjelmiston ylläpidollisia toimia (Poppendieck ja Cusumano 2012). Kanbanin mainitaan monien tutkimusten ja artikkeleiden mukaan olevan myös hyvin yhteensopiva työkalu käytettäväksi myös muiden ketterien ohjelmistokehitysmenetelmien kanssa, esimerkiksi Lean-johtamistapa sopii Kanbanin kanssa hyvin käytettäväksi yhdessä (Poppendieck ja Cusumano 2012; Kinnunen 2015: 16-17).

4 Aiemmat tutkimukset ketterän ohjelmistokehityksen menestystekijöistä

Tämän tutkielman tutkimuskysymykseen vastaamisen alustukseksi on välttämätöntä tarkastella aiheeseen liittyviä tutkimuksia. Seuraavaksi esitellään aiheeseen liittyviä tutkimusasetteluja sekä vertaillaan tutkimuksien tuloksia toisiinsa. Aiheen kriittisen tarkastelun kannalta on perusteltua tarkastella ensin projektin menestystekijöiden tutkimusta, ja sen jälkeen ketterän ohjelmistokehityksen kriittisten menestystekijöiden tutkimusta.

4.1 Aikaisemmat tutkimukset projektien kriittisistä menestystekijöistä

Projektin onnistumisen kannalta kriittisiä menestystekijöitä on Dvirin, Lipovetskyn, Shenharin & Tishlerin (1998) tutkimuksen mukaan tutkittu kyseisen tutkimuksen kirjoitushetkellä jo yli kaksi vuosikymmentä. Stankovicin, Nikolicin, Djordjevicin & Caon (2013) mukaan Bullenin & Rockartin (1981) tutkimus on ensimmäinen tutkimus kriittisistä menestystekijöistä, joten aihetta on tutkittu jo suhteellisen kauan. Dvirin ym. (1998) sekä esimerkiksi Shenharin, Dvirin, Levyn & Maltzin (2001) mukaan on tärkeää huomata, että projektin kriittiset menestystekijät eivät ole täysin universaaleja kaikille projekteille, vaan vaihtelevat riippuen projektin muuttuvista tekijöistä, kuten esimerkiksi tavoitteista ja vaatimuksista, liiketoimintaympäristöstä ja projektin resursseista. Projektien menestystä on mitattu monella eri tapaa, mutta esimerkiksi Dvirin ym. (1998) sekä Radujkovićin & Skejavican (2017) mukaan projektin menestystä perinteisesti mitattu kolmen muuttujan avulla, jotka ovat aika, kustannukset ja laatu. Kuitenkin esimerkiksi Dvirin, Lipovetskyn, Shenharin & Tishlerin (2003) mukaan projektien menestykselle paras mittari on näiden kolmen perinteisen mittarin lisäksi asiakastyytyväisyys.

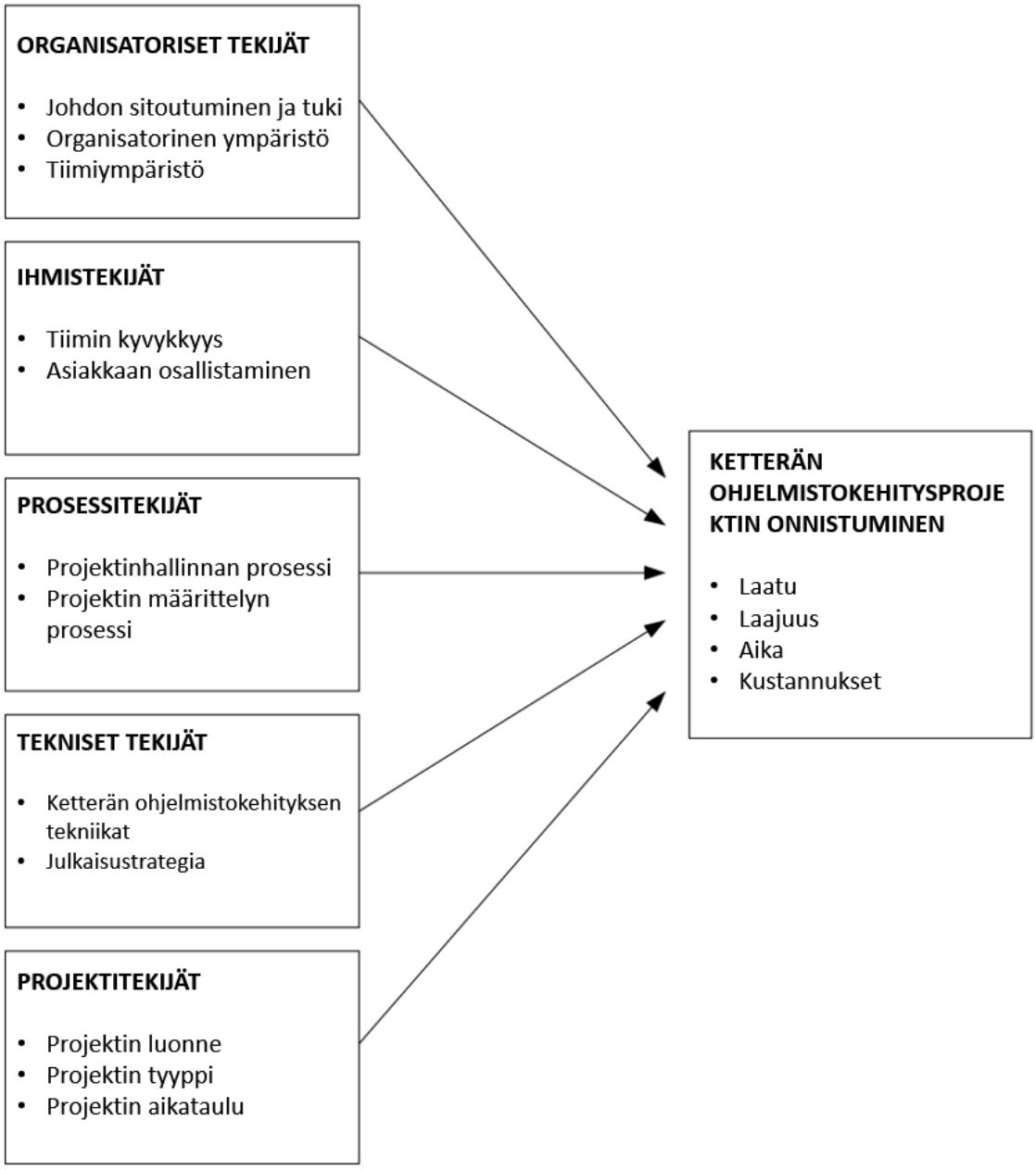
Tietojärjestelmäprojektien kriittisiä menestystekijöitä tutkiessa tutkimuskenttä pienenee huomattavasti. Esimerkiksi Dong, Chuah & Zhai (2004) ovat tutkineet kriittisiä menestystekijöitä määrällisesti 247 tietojärjestelmäprojektissa, ja tutkimuksen mukaan kommunikaatio, vaatimusmäärittely, johdon tuki, projektijohdon kyvykkyys, asiakasyhteistyö, sidosryhmäsuhteiden hallinta ja projektin kontrollointi ovat kriittisiä

menestystekijöitä. Myös hieman tuoreemmissa tutkimuskentän tutkimuksissa esimerkiksi Guntur, Purwandari, Raharjo, Solichah & Kumaralalita (2018) sekä Sanchez, Terlizzi & de Moraes (2017) toteavat samojen menestystekijöiden kriittisyyden tietojärjestelmäprojekteille.

Yhteenvetona voidaan todeta yleisesti projekteille yhteisinä kriittisinä menestystekijöinä olevan huolellinen projektin sopimuksen ja vaatimusten määrittely, asiakaslähtöisyys, projektinhallinta (kuten esimerkiksi aikatauluttaminen, tavoitteiden asettaminen, budjetointi ja työnjako), ja organisaation johdon tuki ja käytännöt projektia kohtaan. (Dvir ym. 1998: 932; Shenhar ym. 2001; Rolstadås, Tommelein, Schiefloe & Ballard 2014)

4.2 Aikaisempien ketterän ohjelmistokehityksen kriittisten menestystekijöiden tutkimusasettelujen esittely

Ketterän ohjelmistokehityksen kriittisiä menestystekijöitä tarkastellessa tutkimuskenttä pienenee entisestäänkin. Chow & Cao (2008) mainitsevat tutkimuksessaan, että ketterän ohjelmistokehityksen kriittisistä menestystekijöistä ei ole oikeastaan yhtään virallista tutkimusta. Stankovic ym. (2013) mainitsevatkin Chowin & Caon (2008) tutkimuksen olevan aihepiirin ensimmäinen virallinen tutkimus. Chow & Cao (2008) suorittivat tutkimuksensa käyttäen määrällisiä menetelmiä, ja päätyivät tutkimuksessaan tutkimusasetteluun, jossa ketterän ohjelmistokehityksen kriittisiä menestystekijöitä tarkastellaan viiden eri kategorian mukaisesti, jotka ovat: organisatoriset tekijät, ihmistekijät, prosessitekijät, tekniset tekijät ja projektitekijät. Kyseiset viisi tekijää on laajennettu yhteensä 12 menestystekijään. Tutkimuksessa tutkittavien projektien menestystä mitattiin neljän eri attribuutin mukaisesti: projektin laatu, laajuus, aika ja kustannukset. Chowin & Caon (2008) tutkimusasettelu on kuvattu alla olevassa kuviossa (ks. kuvio 3).

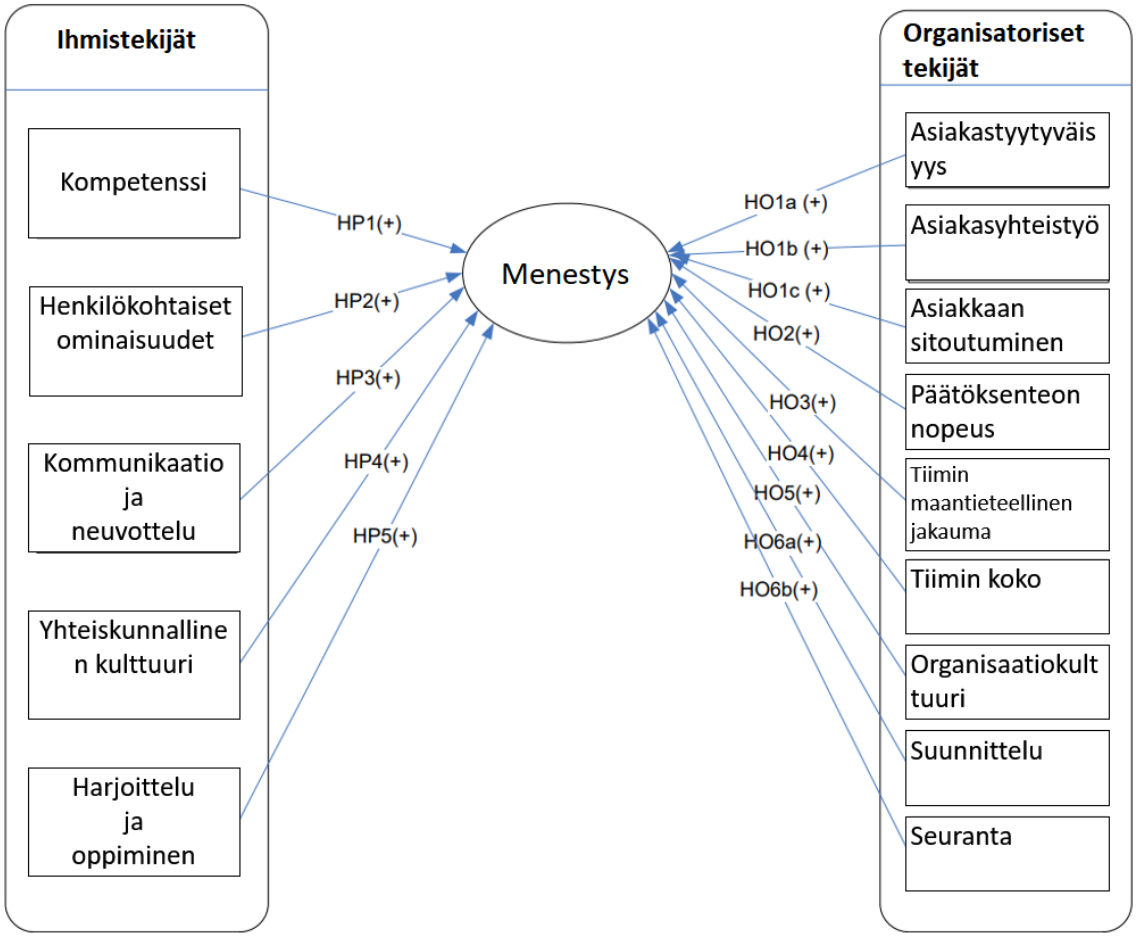


Kuvio 3. Chow'n ja Caon tutkimuksen tutkimusasetelma (Chow & Cao, 2008: 964)

Aldahmash ym. (2017) suorittivat systemaattisen kirjallisuuskatsauksen ketterän ohjelmistokehityksen kriittisten menestystekijöiden arvostetuista julkaisuista lähtöisin olevista empiirisistä tutkimuksista viimeisen kymmenen vuoden ajalta tutkimushetkestä laskettuna. Aldahmash ym. (2017) huomioivat sellaiset kriittiset menestystekijät kirjallisuuskatsauksensa löydöksissä, jotka oli havaittu empiirisen tutkimuksen avulla vähintään kahdessa eri empiirisessä tutkimuksessa. Aldahmash ym. (2017) määrittelevät ennen

tulosten kertomista kriittisten menestystekijöiden olevan sellaisia tekijöitä, joilla on valtava vaikutus projektityön onnistumiselle. Aldahmashin ym (2017) pääasiallisena tavoitteena oli selkeyttää alan tutkimusta, löytää yhteneväisyyksiä ketterän ohjelmistokehityksen kriittisiä menestystekijöitä koskevista empiirisistä tutkimuksista sekä luoda kriittisten menestystekijöiden viitekehys niiden perusteella, jota tarkastellaan seuraavassa alaluvussa.

Misran ym. (2009) määrällisin menetelmin suorittamassa tutkimuksessa ketterän ohjelmistokehityksen kriittisistä menestystekijöistä projektin menestys määritellään lyhentyneenä aikana, pienempinä kustannuksina ja parantuneena laatuna. Menestyksen mittareina tutkimuksessa käytettiin nopeutettua julkaisuaikataulua, kasvatettua sijoitetun pääoman tuottoprosenttia (ROI), kasvanutta kykyä täyttää nykyiset asiakasvaatimukset ja vastata muuttuviin vaatimuksiin sekä projektin yleisten liiketoimintaprosessien parantumisena. Tutkimuksen tutkimusasettelussa ketterän ohjelmistokehityksen kriittiset menestystekijät on jaettu 14 eri hypoteesiin, jotka on jaettu kahteen eri pääkategoriaan: organisatorisiin tekijöihin ja ihmistekijöihin. Tutkimusasetelmaa ja hypoteeseja on selvennetty tarkemmin kuviossa 4 (ks. kuvio 4).



Kuvio 4. Misran ym. hypoteettiset menestystekijät (Misra ym. 2009: 1873)

Misran ym. (2009), Aldahmashin ym. (2017) sekä Chow & Caon (2008) tutkimukset ketterän ohjelmistokehityksen menestystekijöistä nähdään monien lähteiden mukaan kyseisen aihepiirin uraauurtavimpina tutkimuksina. Myös França, da Silva & de Souza Mariz (2010) ovat tutkineet aihepiiriä määrällisin keinoin ja suurilta osin samoin tutkimusasetteluin, kuin Chow & Cao (2008), mutta ovat määritelleet projektin onnistumisen hieman eri tavoin. Stankovicin ym. (2013) laajasti toteutettu kyselytutkimus entisen Jugoslavian alueen IT-yrityksistä tutki ketterän ohjelmistokehityksen kriittisiä menestystekijöitä täsmälleen samojen tutkimusasettelujen mukaisesti kuin Chow & Cao (2008). Näiden lisäksi myös Tam, da Costa Moura, Oliveira & Varajão (2020) tutkivat ketterän ohjelmistokehityksen kriittisiä menestystekijöitä. Erona muihin tässä mainittujen tutkimusten tutkimusasetteluihin oli se, että Tam ym. (2020) tutkivat ketterän ohjelmistoprojektin

menestymiseen vaikuttavista menestystekijöistä ainoastaan ihmistekijät -kategoriaa. Kyseinen tutkimus toteutettiin haastattelemalla 216 ketterien ohjelmistokehitysmenetelmien asiantuntijoita ja mittareina käytettiin projektin kustannusta, aikaa ja asiakastyytyväisyyttä.

4.3 Aikaisempien ketterän ohjelmistokehityksen kriittisten menestystekijöiden tutkimusten tulokset

Chowin & Caon (2008) tekemän määrällisen tutkimuksen tutkimusasettelun hypoteesin mukaan kriittisiä menestystekijöitä olisi ollut 12, mutta tulosten mukaan näistä vain kuusi menestystekijää olivat kriittisiä projektin onnistumiselle: julkaisustrategia, ketterät ohjelmistokehitystekniikat, tiimin kyvykkyys, projektijohdon prosessi, tiimin ympäristö ja asiakasyhteistyö. Chowin & Caon (2008) tutkimuksen mukaan näistä tutkimuksen mukaan kuudesta kriittisestä menestystekijästä julkaisustrategialla on suurin merkitys ketterän ohjelmistoprojektin onnistumiselle. Seuraavaksi tärkeimpinä menestystekijöinä he nostavat esille erityisesti ketterän ohjelmistokehityksen tekniikoiden hallitsemisen ja tiimin kyvykkyyden. Kuten edellisessä alaluvussa mainittiin, Chowin & Caon (2008) tutkimusasettelun menestystekijät jaettiin viiteen pääkategoriaan: ihmistekijöihin, organisatorisiin tekijöihin, teknisiin tekijöihin, prosessitekijöihin ja projektitekijöihin, joista tulosten mukaan tekniset tekijät osoittautuivat kaikista kriittisimmiksi. Teknisten tekijöiden jälkeen ihmistekijät-kategoria osoittautui seuraavaksi kriittisimmäksi näistä viidestä kriittisten menestystekijöiden pääkategoriasta. Näistä viidestä kriittisten menestystekijöiden pääkategoriasta ainoastaan yhdellä, projekti -kategorian menestystekijöillä, ei ollut Chowin & Caon (2008) tutkimuksessa vaikutusta projektin menestymiselle. Projekti-kategorian hypoteettisia menestystekijöitä ovat kyseisessä tutkimuksessa esimerkiksi dynaaminen aikataulu, helposti hallittava projektitiimi tai projektitiimin riippumattomuus projektin sidosryhmien tarpeista. (Chow & Cao, 2008)

Stankovicin ym. (2013) tutkimus on eräänlainen laajennus Chow & Caon (2008) tutkimukselle, ja siinä käytettiin täsmälleen samanlaista tutkimusasettelua ja menetelmiä. Kuten aiemmin tämän tutkielman johdannossa mainittiin, ketterän ohjelmistokehityksen

tutkimuskenttä on hyvin epäselvä ja tulokset eroavat toisistaan jonkin verran, joten myös Stankovicin ym. (2013) tutkimus ja Chow & Caon (2008) tutkimuksen tulokset eroavat paljon toisistaan. Stankovicin ym. (2013) tutkimuksen mukaan kriittisinä menestystekijöinä voidaan pitää tiimin ympäristöä, tiimin kyvykkyyttä, asiakkaan osallistamista, ketterän ohjelmistokehityksen tekniikoita, julkaisustrategiaa, projektijohdon prosessia sekä projektin tyyppiä ja aikatauluttamista. (Stankovic ym. 2013)

Aldahmashin ym. (2017) tutkimuksessa kriittisten menestystekijöiden kriteerien mukaan menestystekijä tuli olla löydetty vähintään kahdessa kahdeksasta tutkimukseen mukaan otetussa empiirisessä tutkimuksessa. Tutkimuksessa löydettiin lopulta yhteensä kahdeksan kriittistä menestystekijää, jotka olivat järjestyksessä kriittisimmät ensimmäisenä mainittuna: tiimin kyvykkyys ja oppiminen, organisaatiokulttuuri, ketterän ohjelmistokehityksen tekniikat, asiakasyhteistyö, projektinjohdon prosessi, julkaisustrategia, kommunikaatio, ja ylimmän johdon tuki. Aldahmashin ym. (2017) tutkimuksen tutkimusasettelussa kriittiset menestystekijät on jaettu neljään kategoriaan: organisatorisiin tekijöihin, teknisiin tekijöihin, prosessitekijöihin ja ihmistekijöihin. Tutkimuksen tulosten mukaan näistä neljästä kategoriasta ihmistekijät ja organisatoriset tekijät ovat kriittisimmät ketterän ohjelmistokehitysprojektin onnistumiselle. (Aldahmash ym. 2017)

Misran ym. (2009) data-analyysin tavoin toteutetussa tutkimuksessa kriittiset menestystekijät jaettiin 14 hypoteesiin tutkimuksen tutkimusasettelussa, jotka on luokiteltu kahteen pääkategoriaan: ihmistekijöihin ja organisatorisiin tekijöihin. Misra ym. (2009) käsittelivät dataa monilla eri menetelmillä, kuten esimerkiksi korrelaatioanalyysillä ja regressioanalyysillä. Yhdeksän menestystekijää 14 tekijästä todettiin lopulta kriittisiksi projektien menestymiselle. Nämä yhdeksän menestystekijää olivat asiakastyytyväisyys, asiakasyhteistyö, asiakassitoutuminen, päätöksentekoaika, yrityskulttuuri, projektinhallinta, henkilökohtaiset kyvyt, yhteiskunnallinen kulttuuri sekä tiimin koulutus ja oppiminen. Tiimin jakautumisella, ryhmän koolla, suunnittelulla, teknisellä pätevyydellä, viestinnällä tai neuvottelulla ei analyysissä löydetty olevan suurta vaikutusta ketterän ohjelmistokehitysprojektin menestymiseen. (Misra ym. 2008)

Françan ym. (2010) määrällisin menetelmin toteuttama tutkimus suoritettiin siis hyvin samalla tavalla kuin Chowin & Caon (2008) tutkimus. Ketterän

ohjelmistokehitysprojektin menestymiseen vaikuttavia kriittisiä menestystekijöitä ovat Françan ym. (2010) mukaan säännöllinen julkaisustrategia, ohjelmiston tärkeimpien osien julkaisu ensin, integraatioon panostaminen, tiimin kyvykkyys ja asiantuntijuus, ketterien ohjelmistokehitystekniikoiden hallitseminen, johdonmukainen ja selkeä tiimityö sekä hyvä asiakasyhteistyö.

Tam ym. (2020) toteuttivat tutkimuksensa haastattelemalla 216 ketterän ohjelmistokehityksen parissa työskentelevää asiantuntijaa. Ketterän ohjelmistokehitysprojektin kriittisten menestystekijöiden ihmistekijöistä kyseisen tutkimuksen mukaan olivat ainoastaan tiimin kyvykkyys ja asiakasyhteistyö sekä asiakkaan osallistuminen.

4.4 Aikaisempien tutkimustulosten vertailu

Ketterän ohjelmistokehityksen kriittisten menestystekijöiden tutkimukseen pätee samat huomiot kuin koko ketterän ohjelmistokehityksen tutkimukseen, sen löydökset ja tulokset eivät ole yhteneväisiä. Myös kriittisten menestystekijöiden tutkimuksessa on ketterän ohjelmistokehityksen tutkimuksen tavoin tärkeä ottaa huomioon, että menestystekijät vaihtelevat suuresti riippuen projektista, sen vaatimuksista ja liiketoimintaympäristöstä. Tulevaisuuden tutkimuksen ja käytännön hyötyjen kannalta aikaisempien tutkimusten tarkastelu ja uuden tutkimustiedon löytäminen on kuitenkin hyvin antoisaa. Alla olevassa taulukossa (ks. taulukko 2) on esitelty tässä luvussa mainittujen arvostettujen tutkimusten löydökset kriittisistä menestystekijöistä. Projektin menestymiseen vaikuttavat tekijät on siis jaettu kolmeen eri luokkaan: kriittisiin menestystekijöihin, tärkeisiin tekijöihin sekä tekijöihin, joilla ei paljoa vaikutusta projektin menestykseen. Kaikissa näistä viidestä vertaillusta tutkimuksesta ei luokiteltu tekijöitä yksiselitteisesti näihin kolmeen luokkaan, mutta löydöksistä käy kuitenkin hyvin ilmi menestystekijöiden tärkeydet.

Tämän lisäksi vertailussa haasteita tuottaa myös menestystekijöiden erilainen määrittely tutkimuksissa. Esimerkiksi Aldahmash ym. (2017) toivat tutkimuksensa tuloksissa esille ainoastaan kriittiset menestystekijät. Misran ym. (2009) tutkimuksessa ei käsitellä kaikkia samoja tekijöitä, kuin muissa tutkimuksissa. Esimerkiksi Julkaisustrategia menestystekijänä ei kyseisellä nimellään ole mukana Misran ym. (2009) tutkimuksessa, mutta se

on mukana näissä muissa neljässä tutkimuksessa. Vertailuun ei otettu mukaan Tamin ym. (2020) tutkimusta, koska kyseisessä tutkimuksessa tutkittiin kriittisistä menestystekijöistä ainoastaan ihmistekijät -kategoriaa. On kuitenkin mainittava, että kyseisen tutkimuksen löydökset vahvistavat ainakin osaltaan Aldahmashin ym. (2017), Françan ym. (2010), Misran ym. (2009) sekä Chowin & Caon (2008) tutkimusten tuloksia, koska myös näissä tiimin kyvykkyys ja asiakasyhteistyö ovat kriittisiä tai tärkeitä menestystekijöitä.

Näiden viiden tutkimuksen tuloksista on vaikea tehdä selkeää johtopäätöstä, mitkä menestystekijät ovat kriittisimpiä ketterien ohjelmistokehitysprojektien kannalta. Eniten mainittuina kriittisinä tai tärkeinä tekijöinä nousivat kuitenkin esiin projektinhallinnan prosessi, tiimin kyvykkyys sekä asiakasyhteistyö, jotka nähtiin Stankovicin ym. (2013) tutkimuksesta huolimatta kaikissa tutkimuksissa tärkeänä tai kriittisenä tekijänä ketterän ohjelmistoprojektin menestymisen kannalta.

Taulukko 2. Aikaisemmissa tutkimuksissa kerrottujen menestystekijöiden vertailu

	Kriittiset menestystekijät	**Tärkeät tekijät**	**Tekijät, joilla ei paljoa vaikutusta menestykseen**
Chow & Cao, 2008	• Julkaisustrategia • Ketterän ohjelmistokehityksen tekniikat • Tiimin kyvykkyys	• Asiakasyhteistyö • Projektinhallinnan prosessi • Tiimin ympäristö	• Johdon sitoutuminen • Organisaatioympäristö • Projektin määrittelyn prosessi • Projektin luonne • Projektin tyyppi • Projektin aikataulu
Stankovic ym. 2013	• Projektin luonne	• Projektin aikataulu • Projektinhallinnan prosessi • Projektin määrittelyn prosessi	• Asiakasyhteistyö • Johdon sitoutuminen • Julkaisustrategia • Ketterän ohjelmistokehityksen tekniikat • Organisatorinen ympäristö • Projektin tyyppi • Tiimiympäristö • Tiimin kyvykkyys
Misra ym. 2009	• Asiakasyhteistyö • Asiakkaan sitoutuminen • Päätöksenteon nopeus ja dynaamisuus • Organisaatiokulttuuri • Yhteiskunnallinen kulttuuri • Projektin seuranta • Harjoittelu ja oppiminen	• Asiakastyytyväisyys • Henkilökohtaiset ominaisuudet ja kyvyt	• Tiimin maantieteellinen sijainti • Tiimin koko • Projektin suunnittelu • Tiimin kompetenssi • Kommunikointi ja neuvottelu
França ym. 2010	• Julkaisustrategia • Projektin hallinnan prosessi	• Asiakasyhteistyö • Ketterän ohjelmistokehityksen tekniikat • Tiimin kyvykkyys • Tiimiympäristö	• Johdon sitoutuminen • Organisaatioympäristö • Projektin määrittelyn prosessi • Projektin luonne • Projektin tyyppi • Projektin aikataulu
Aldahmash ym. 2017	• Tiimin kyvykkyys ja oppiminen • Organisaatiokulttuuri • Ketterän ohjelmistokehityksen tekniikat • Asiakasyhteistyö • Projektinhallinnan prosessi • Julkaisustrategia • Kommunikaatio • Ylimmän johdon tuki		

Kuten myös ketterän ohjelmistokehityksen tutkimukseen, myös kriittisten menestystekijöiden tutkimusten tuloksista löytyy monia ristiriitoja. Kun tarkastellaan organisaatiokulttuurin ja yhteiskunnallisen kulttuurin merkitystä ketterän ohjelmistokehitysprojektin

menestykselle, Misra ym. (2009) sekä Aldahmash ym. (2017) korostavat niiden merkitystä projektin menestymiselle, mutta muissa kolmessa tutkimuksessa niiden merkitystä ei tuoda esille kriittisinä, eikä tärkeinä tekijöinä projektin onnistumiselle. Ainoastaan Chow & Cao (2008) sekä Aldahmash ym. (2017) pitävät tiimin kyvykkyyttä ja kompetenssia kriittisenä menestystekijänä projektin onnistumiselle. Toisaalta França ym. (2010) ja Misra ym. (2009) pitävät tiimin kyvykkyyttä ja kompetenssia tärkeänä tekijänä projektin onnistumiselle, kun taas Stankovicin ym. (2013) mukaan tiimin kyvykkyydellä ja kompetenssilla ei nähdä paljoa vaikutusta projektin edistymiselle. Oletettavasti kuitenkin substanssiosaamista tarvitaan, koska esimerkiksi ohjelmointi tai asiakkaan tarpeiden määrittäminen voidaan nähdä toimena, jossa tarvitaan erityisasiantuntemusta.

Myöskään tiimiympäristön arvostaminen menestystekijänä projektin onnistumiselle ei ole näiden tutkimuksen tuloksissa yhteneväistä. Chow & Cao (2008) ja França ym. (2010) pitävät tiimiympäristöä tärkeänä tekijänä, kun taas Stankovicin ym. (2017) mielestä tiimin ympäristöllä ei ole paljoa vaikutusta menestykseen.

Edellä mainituista ristiriidoista huolimatta tutkimuksista löytyy kuitenkin monia yhteneväisyyksiä, joita pidetään joko kriittisinä menestystekijöinä tai tärkeinä tekijöinä projektin onnistumiselle. Näitä tekijöitä ovat asiakasyhteistyön tai asiakkaan osallistumisen merkitys, ketterän ohjelmistokehityksen mukainen julkaisustrategia sekä osaavat tiimin jäsenet.

Vaikka kirjallisuuskatsauksen tuloksien voidaan sanoa olevan kattavat, on kuitenkin tarpeen selvittää haastattelujen avulla ketterän ohjelmiston kriittisiä menestystekijöitä, erityisesti jotta tutkielma toisi yhteneväisyyttä aiheen tutkimuskenttään. Tämän lisäksi haastattelujen tarpeellisuutta perustelee se, että tässä tutkielmassa esiteltävä ketterän ohjelmistokehityksen malli on kehitelty silmällä pitäen erityisesti suomalaisen ohjelmistokehityksen ja ohjelmistoliiketoiminnan kehittymistä, jolloin suomalaisten asiantuntijoiden haastattelun voidaan katsoa olevan tarpeellista. Myös tutkielman tekijän työura keskittyy todennäköisesti suomalaiseen ohjelmistoliiketoimintaan, jolloin henkilökohtainen hyöty korostuu käsitellessä suomalaisten haastattelijoiden kertomia ketterän ohjelmistokehityksen kriittisiä menestystekijöitä.

5 Empiirisen tutkimuksen menetelmä

Tässä luvussa tarkastellaan tutkielman tutkimusmenetelmää ja perustellaan sen valintaa. Tämän jälkeen käydään läpi tiedonkeruumenetelmää sekä perustellaan myös tiedonkeruumenetelmän valintaa. Tiedonkeruumenetelmäksi valittiin teemahaastattelut, joten tässä luvussa tarkastellaan myös haastateltavien valintaprosessia, haastateltavien taustoja, haastattelurunkoa, haastatteluaineiston analyysimenetelmää sekä arvioidaan haastattelujen sekä koko tutkimuksen luotettavuutta.

5.1 Laadullinen tutkimus

Laadullista tutkimusta on määritelty Hirsjärven, Remeksen & Sajavaaran (2009) mukaan menetelmäsuuntaukseksi, joka pyrkii ymmärtämään tutkittavaa ilmiötä syvällisesti ja kokonaisvaltaisesti. Tilastollisen yleistettävyyden tai ennalta-asetettujen väittämien todentamisen sijasta laadullisen tutkimuksen tavoitteena on kuvata, ymmärtää, tulkita ja paljastaa tosiasioita tutkittavasta aihepiiristä (Hirsjärvi ym. 2009; Eskola & Suoranta, 2008) Laadullinen tutkimusmenetelmä sopii myös tämän tutkimuksen tutkimusmenetelmäksi, koska kerätyn aineiston pitäisi olla mahdollisimman kokonaisvaltaista ja syvällistä, pääosin siitä syystä, että ketterän ohjelmistokehityksen menestystekijöissä on niin paljon liikkuvia tekijöitä, kuten esimerkiksi liiketoimintaympäristö, projektin vaatimukset ja tavoitteet. Johtuen liikkuvien tekijöiden runsaasta määrästä, yleistettävyys eri tapausten välillä on hieman epäluotettavaa ja vaikeaa. (Hirsjärvi ym. 2009; Pyörälä 2002) Tämän tutkielman luvussa 4 tarkasteltiin ketterän ohjelmistokehityksen kriittisten menestystekijöiden tutkimuksia, jotka oli toteutettu käyttäen pääasiassa kvantitatiivisia eli määrällisiä tutkimusmenetelmiä. Kuten muun muassa luvun 4 yhteenvedossa todettiin, ketterän ohjelmistokehityksen kriittisten menestystekijöiden tutkimuksista ei löydetty korkeaa tilastollista yleistettävyyttä lähinnä johtuen tutkittavan aiheen monimutkaisuudesta. Tästä syystä laadullisten tutkimusmenetelmien käyttö on perusteltua tässä tutkielmassa.

5.2 Tiedonkeruumenetelmä

Tämän tutkielman tiedonkeruumenetelmäksi valittiin haastattelut, koska tavoitteena oli kerätä ja analysoida nimenomaan käytännön tietoa ja kokemuksia. Parhaiten soveltuvana haastattelumenetelmänä päädyttiin valitsemaan teemahaastattelut. Haastattelumenetelmäksi valittiin teemahaastattelut, koska tällöin aiheen tutkiminen on mahdollisimman kokonaisvaltaista ja syvällistä, mutta silti pidetään huoli, että haastattelu pysyy tietyissä teemoissa haastattelun ajan, eikä haastattelun aihe karkaa epäolennaisiin aiheisiin. (Hirsjärvi & Hurme, 2014)

Haastattelun toteuttaminen avoimena haastatteluna olisi saattanut johtaa aiheen eksymiseen muihin aiheisiin haastattelun aikana, koska ketterä ohjelmistokehitys ja ketterän ohjelmistokehityksen kriittiset menestystekijät on aihealueena vaikeasti rajattavissa. Avoimessa haastattelussa aihe olisi saattanut karata helposti muihin asioihin, kuten läheisesti ketterään ohjelmistokehitykseen liittyviin aiheisiin, kuten esimerkiksi henkilöstöjohtamiseen, organisaatiorakenteisiin, ohjelmointiin tai projektijohtamiseen.

Avoimen haastattelun vastakohtana pidetään strukturoitua haastattelua, joka taas olisi saattanut olla liian rajattu, eikä välttämättä olisi jättänyt tilaa haastateltavan improvisaatiolle, esimerkeille ja perusteluille (Eskola & Suoranta, 2008; Hirsjärvi & Hurme, 2014). Haastattelumenetelmän valinnassa pyrittiin siis kiinnittämään huomiota siihen, että aiheesta saadaan mahdollisimman syvällistä, hyödyllistä ja käytännönläheistä tietoa, eli niin sanottua hiljaista tietoa organisaatioiden sisältä, mutta kuitenkin niin, että haastattelun puheenaihe pysyy tietyissä teemoissa.

5.3 Haastattelurunko

Teemahaastattelujen runko muodostuu neljästä pääosiosta:

- Haastateltavan taustatiedot
- Ketteryys ja ketteryyden määritelmä haastateltavan organisaatiossa
- Haastateltavan kertomat menestystekijät ilman alan tutkimusten johdattelua

- Haastateltavan kertomat menestystekijät Aldahmashin ym. (2017) viitekehyksen mukaan

Taustatiedot -osiossa kartoitettiin haastateltavan työtehtävää, hänen edustamaa organisaatiota, organisaation toimialaa ja markkinoita, haastateltavan työkokemusta sekä hänen henkilökohtaista osaamistaan ketteristä menetelmistä. Taustatietojen kartoittamisella pyrittiin luomaan ymmärrystä, miten edellä mainitut tekijät vaikuttaisivat haastateltavan edustaman organisaation ketterien ohjelmistokehitysprojektien menestystekijöihin.

Haastateltavan organisaation ketteryyden selvittäminen on olennaista, jotta tiedetään miten ketteriä periaatteita, ketteriä käytänteitä tai ketteriä menetelmiä sovelletaan organisaatiossa. Organisaation ketteryyden selvittämisellä pyritään siis toisin sanoen ymmärtämään, mitkä menestystekijät ovat tärkeimpiä organisaatiossa ja organisaation projekteissa.

Itse menestystekijöistä pyrittiin saamaan tietoa aluksi niin, että haastateltavalle annettiin suhteellisen vapaat kädet kertoa, mitkä menestystekijät hänen organisaatiolleen ovat tärkeimpiä projektien onnistumisessa. Tällä pyrittiin saamaan esille nimenomaan haastateltavan organisaatiolle tärkeimpiä tekijöitä onnistua ketterissä ohjelmistoprojekteissa johdattelematta haastateltavaa tarkemmin tutkimuksissa löydettyihin menestystekijöihin. Tällä tavalla pyrittiin löytämään mahdollisimman syvällistä ja käytännönläheistä tietoa eli niin sanottua hiljaista tietoa.

Tämän jälkeen siirryttiin tarkemmin tarkastelemaan tutkimuksissa löydettyjä ketterän ohjelmistokehityksen kriittisiä menestystekijöitä Aldahmashin ym. (2017) luoman kriittisten menestystekijöiden viitekehyksen mukaisesti. Viitekehyksessä kriittiset menestystekijät on siis jaettu neljään eri osaan: ihmistekijöihin, teknisiin tekijöihin, projektijohdon prosessi -tekijöihin sekä organisatorisiin tekijöihin. Haastatteluun sopivia viitekehyksiä olisi ollut tarjolla useampiakin, mutta Aldahmashin ym. (2017) viitekehys tuntui sopivimmalta haastattelun runkoa ajatellen.

5.4 Haastateltavien valinta ja haastattelut

Hirsjärvi ym. (2004) painottavat tutkimuksessaan, että laadulliselle tutkimukselle on ominaista ja tärkeää valita tutkimuksen haastateltavat huolella. Tämän tutkielman haastateltavien valinnassa pyrittiin kiinnittämään huomiota haastateltavien työkokemukseen ketterien ohjelmistokehitysmenetelmien parissa, tietämykseen organisaation liiketoiminnasta ja läheiseen yhteyteen ketterään projektityöskentelyyn ja asiakasyhteistyöhön. Haastattelijoita etsittiin ja valikoitiin LinkedIn -yhteisöpalvelun kautta, jossa keskustellaan pääasiassa työelämään, talouteen, liiketoimintaan ja teknologiaan liittyvistä aiheista. Tästä syystä LinkedInin käyttäjätkin ovat yleensä kyseisten toimialojen työntekijöitä. LinkedIniä käytetään hyvin paljon myös työnhaussa ja rekrytoinneissa, ja osin tästä syystä palvelun käyttäjien profiilit ovat hyvin kattavia ja toimivat myös ansioluetteloina. Ansioluettelomaisten profiilien takia haastattelua varten valikoitujen asiantuntijoiden etsiminen ja löytäminen oli hyvin helppoa. Otin yhteyttä lopulta yhteensä 24 kokeneeseen asiantuntijaan, jotka työskentelivät pääasiassa it-alalla projektin johtotehtävissä tai keskijohdossa. Näistä 24 asiantuntijasta haastattelin lopulta kahdeksaa asiantuntijaa. Kaikki haastattelut toteutettiin etäyhteyksien avulla, pääosin Skypen, Microsoft Teamsin ja Zoomin kautta. Kyseiset sovellukset ovat videokonferenssialustoja. Haastattelujen järjestäminen etäyhteyksien avulla oli paras valinta osin Covid-19-viruksen takia säädettyjen rajoitusten takia, ja osin etäyhteyksien käytön helppouden takia. Kyseiset sovellukset ovat siis videokonferenssialustoja. Haastattelujen keskimääräinen kesto oli noin 40 minuuttia, lyhimmän kestäessä vajaa puoli tuntia ja pisimmän kestäessä noin tunnin. Teemahaastatteluille ominaiseen tapaan haastateltaville annettiin aikaa kertoa kokemuksista ja yksityiskohdista rauhassa, josta johtuen haastattelujen kestot vaihtelivat hieman. Haastatteluiden alussa kerrattiin haastattelun rakenne ja muoto sekä painotettiin haastattelijoille, että haastateltavat sekä heidän edustamansa organisaatiot pysyvät täysin anonyymeinä, joka toimi osittain myös keinona saada haastateltavat suostuteltua haastatteluun ja saada ehkä helpommin kerättyä arvokasta haastattelutietoa.

5.5 Aineiston analyysi

Aineiston analyysimenetelmäksi valikoitui teemoittelu. Teemoittelu on Tuomen & Sarajärven (2018) mukaan teemoittelussa on kyse laadullisen aineiston pilkkomisesta ja ryhmittelystä erilaisten aihepiirien tai teemojen mukaan, jonka ansiosta on mahdollista vertailla tiettyjen teemojen esiintymistä aineistossa. Teemoittelun voidaan siis nähdä mahdollistavan kriittisten menestystekijöiden löytämisen ja niiden tärkeyden arvioimisen ketterien ohjelmistokehitysprojektien onnistumiselle. Teemoittelun valitsemista aineiston analyysimenetelmäksi puolsi myös Saaranen-Kauppisen & Puusniekan (2006) mukaan myös se, että teemoittelu on hyvin luonteva analyysimenetelmä teemahaastattelulla kerätyn aineiston analysoimiseen, koska tällöin analysointiprosessi voi edetä mahdollisimman suoraviivaisesti. Aineiston analyysiä helpotti huomattavasti se, että haastattelijat antoivat luvan haastattelujen nauhoittamiseen. Nauhoituksen ansiosta haastatteluihin voitiin palata analysointivaiheessa. Nauhoituksen lisäksi haastattelut litteroitiin ja useissa haastatteluissa ilmenneiden yhtenevät menestystekijät värikoodattiin litteroidessa analysoimisen ja tulosten kirjallisuuteen vertaamisen helpottamiseksi, joten aineiston analyysi aloitettiin jo haastattelujen aikana. Itse analysoimisessa hyödynnettiin paljon vinkkejä Milesin, Hubermanin ja Saldañan (2014) tutkimuksesta, jonka mukaan aineiston analyysi on määritelty kolmena samanaikaisena tehtävänä: aineiston tiivistämisenä, aineiston esittämisenä sekä johtopäätösten tekemisenä.

5.6 Tutkimuksen luotettavuus

Myersin & Newmanin (2007) mukaan laadullisiin haastatteluihin liittyy monia ongelmakohtia, jotka on syytä ottaa huomioon tutkimuksen luotettavuutta ajateltaessa. Runesonin & Höstin (2009) tutkimuksen mukaan tutkimusten luotettavuutta arvioidaan usein validiteetin ja reliabiliteetin näkökulmasta. Validiteetilla tarkoitetaan tulosten yleistä luotettavuutta, eli kuinka luotettavina tuloksia voidaan yleisesti pitää. Reliabiliteetti taas ottaa kantaa tulosten toistettavuuteen, eli saataisiinko samasta aineistosta samat tulokset eri analysointikerroilla.

Tämän tutkimuksen luotettavuuteen vaikuttavia ongelmakohtia ja haasteita voidaan ajatella olevan ainakin seuraavat kolme asiaa. Ensinnäkin haastattelussa haastateltavilla saattaa olla paine vastata kysymyksiin, vaikka heillä ei olisikaan empiirisiä kokemuksia asiaan liittyen. Haastattelun alussa haastateltaville kuitenkin painotettiin, että haastattelussa halutaan kuulla nimenomaan reaalimaailman tapahtumista, mutta tästä huolimatta on hyvin mahdollista, että erityisesti haastateltavan empiirisen tiedot ollessa rajallinen haastateltava saattaa kertoa menestystekijöistä lukemiensa teorioiden tai luulojen pohjalta, eikä omien kokemusten pohjalta.

Toiseksi joitain asioita ei haluta kertoa, koska ne saattavat olla liiketoiminnallisesti herkkää tietoa. Tätä ongelmakohtaa pyrittiin ehkäisemään painottamalla haastateltaville, että heidän nimensä ja edustamansa organisaatiot pysyvät täysin anonyymeinä.

Suurimpana tutkielman luotettavuuteen vaikuttavana tekijää voidaan kuitenkin ajatella olevan tutkimuksen otanta: 23 haastateltavasta asiantuntijasta, joihin otettiin yhteyttä, vain 8 suostui haastatteluun. On kuitenkin syytä painottaa tässä kohtaa sitä, että haastatteluihin haluttiin valikoida vain sellaisia henkilöitä, joilla on tarpeeksi asiantuntemusta aiheesta ja vastaukset olisivat mahdollisimman laadukkaita.

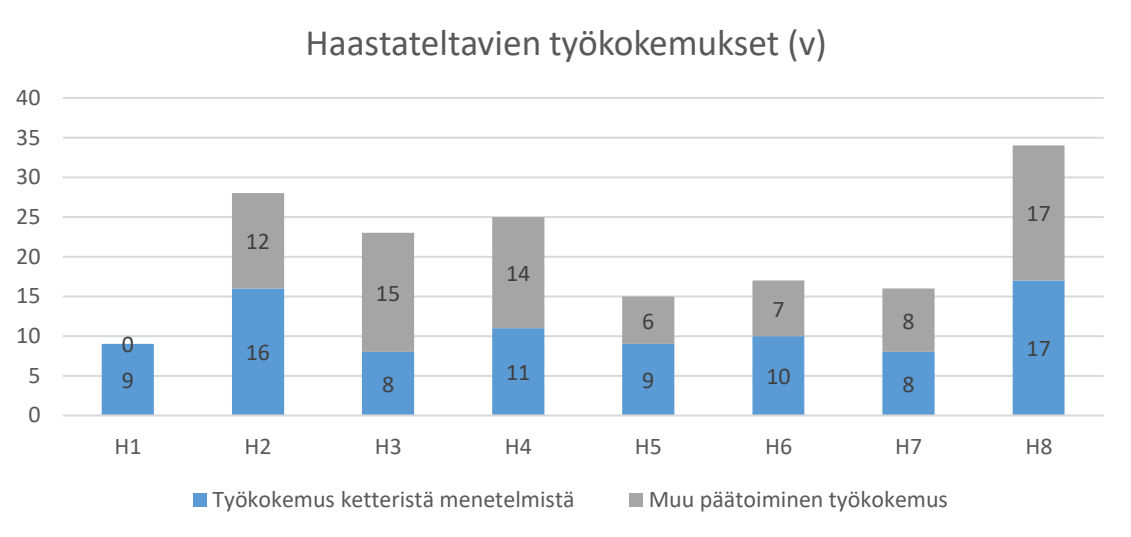
6 Tulokset

Tässä luvussa käsitellään tutkielman empiirisen osion tulokset. Ennen itse tuloksiin syventymistä on tarpeellista käsitellä haastateltavien roolia, työkokemusta, tietämystä ketteristä menetelmistä sekä heidän edustamiaan yrityksiä.

6.1 Haastateltavien taustatiedot

Kuten aiemmin tässä tutkielmassa mainittiin, haastateltavien valinnassa pyrittiin kiinnittämään huomiota haastateltavien työkokemukseen ketterien ohjelmistokehitysmenetelmien parissa, tietämykseen organisaation liiketoiminnasta ja läheiseen yhteyteen ketterään projektityöskentelyyn ja asiakasyhteistyöhön.

Haastateltavien asiantuntijoiden työkokemuksissa oli aika paljon vaihtelua, mutta kaikilla haastateltavilla oli kuitenkin vähintään 8 vuoden kokemus ketterien menetelmien parissa työskentelystä. Haastateltavien keskimääräinen työkokemus ketterien menetelmien parissa työskentelystä oli 11 vuotta, joka huomioon ottaen haastatteluun saatiin valikoitua suhteellisen kokeneita asiantuntijoita. Lyhin päätoiminen työkokemus oli 9 vuotta ja haastateltavien keskimääräinen työkokemus 21 vuotta. Haastateltavien työkokemukset on eritelty tarkemmin kaaviossa alla. (ks. kuvio 5).



Kuvio 5. Haastateltavien työkokemukset

Haastateltavien roolit vaihtelivat hieman riippuen osaksi siitä, millä titteleillä he itseään kutsuvat LinkedInissä. Haastateltavista neljä asiantuntijaa toimivat projektinjohdon tehtävissä, kuten projektipäällikkönä tai Scrum Masterina. Haastateltavista kaksi asiantuntijaa toimivat keskijohdossa ja loput kaksi asiantuntijaa it-konsultteina ja tuoteomistajina.

6.2 Haastateltavien edustamien yritysten ketteryys

Tässä luvussa tarkastellaan haastateltavien käsitystä ketteryydestä, heidän edustamiensa yritysten ketteryyttä sekä sitä, millaisia hyötyjä yritykset ovat ketterien periaatteiden mukaisesta työskentelystä saaneet.

Haastateltavien käsitykset ketteryydestä ja ketterästä ohjelmistokehityksestä olivat hyvin linjassa sen kanssa, miten esimerkiksi Conboy (2009) määrittelee ketterän ohjelmistokehityksen. Jokaisessa yrityksessä haastateltavien kertomusten mukaisesti ketteryyttä arvostettiin sekä sen hyödyntämiseen ja kehittämiseen panostettiin. Varsinaisista ketteristä ohjelmistokehitysmenetelmistä erityisesti Scrum esiintyi kaikkien haastateltavien vastauksissa, mutta jokainen haastateltava mainitsi, että ketteriä menetelmiä harvoin käytetään sellaisenaan, vaan ketteriä menetelmiä käytetään kullakin hetkellä parhaiten tarpeisiin, tavoitteisiin ja kontekstiin sopivin tavoin. Syitä tähän olivat yrityksen omiin toimintatapoihin liittyvät vaatimukset sekä erityisesti asiakkaan ja liiketoimintaympäristön vaatimukset.

> Kyllä me ehdottomasti valitaan sen mukaan ne käytettävät menetelmät, mitä meiltä vaaditaan. Jos me tehtäisiin täysin oppikirjan mukaan, niin jotenkin se olis tosi kankeata. Asiakaslähtöisyys on tässä se ykkösjuttu, eli kommunikoidaan koko ajan asiakkaan kanssa, että onko tämä hyvä ja onhan tämä se mitä haluatte. Eli iteroiden ja inkrementaalisesti edetään. Samalla sitten kuullaan asiakkaalta uusia ideoita, ja monesti pyritään ehdottelemaankin niitä mahdollisimman aktiivisesti. (Haastateltava 5)

Tämän lisäksi muutama haastateltava kertoi aidoksi ketteryydeksi sen, että yrityksellä on kyky valita kuhunkin projektiin vaatimusten mukaiset työkalut, eikä työskentele ennalta määritettyjen menetelmien mukaisesti.

> Scrumin pohjalta me yleensä lähdetään rakentamaan meidän projektia, mutta ei mennä täysin sen mukaisesti. Tärkeintä on keskittyä siihen, mistä asiakas saa arvoa, eikä siihen, mitä oppikirjan mukaan meidän pitäisi tehdä. Jos mentäisiin pelkästään oppikirjan eikä asiakkaan tarpeiden mukaan, se olisi kaukana ketteryydestä. (Haastateltava 2)

Haastateltavat määrittelivät ketteryyttä monin eri tavoin, kuten ketteryyttä määritellään monin eri tavoin myös alaa tutkivassa kirjallisuudessa. Kuten alan kirjallisuudessakin, esiin tärkeinä tekijöinä haastatteluissa ketterää ohjelmistokehitystä määritellessä nousivat erityisesti asiakaslähtöisyys, matalahierarkkinen ja itseohjautuva päätöksenteko, kommunikaatioon panostaminen, usein tapahtuva julkaisustrategia sekä oppimisen ja kokeilemisen kulttuuri.

> Jos tiivistää pitää muutamaan pointtiin, niin kyllä se on se, että mennään täysin asiakkaan specsien mukaisesti. Ja tärkeätä on se, että niitä specsejä tarkennetaan koko ajan. Koko ajan siis parannetaan olemassa olevaa. Meidän julkaisustrategia on siis mahdollisimman tiivis ja koko ajan kysytään palautetta asiakkaalta. Tää tietysti vaatii asiakkaan sekä tietysti meidän puolelta hyvää kommunikaatiota ja nopeata itseohjautuvaa päätöksentekoa. (Haastateltava 2)

> No tärkein asia on keskittyä siihen, että tuotetaan asiakkaalle mahdollisimman paljon arvoa, eli keskitytään olennaiseen. Vaatii tietysti suoraa ja avointa keskusteluyhteyttä asiakkaan kanssa, että voidaan heti sanoa jos mennään väärään suuntaan, tarpeet on muuttunu tai jotain pitäisi muuttaa ohjelmistossa. Parashan se on jos asiakas on mukana koodaajien kanssa, mutta aina se ei onnistu. (Haastateltava 7)

> Asiakkaalla on tietysti aina kiire saada ohjelma käyttöön, eli meidän pitää toimia nopeasti ja koodata vaan tärkeimmät asiat ensin niin tiedetään, että onhan oikea suunta. Tää tietysti vaatii hyvää specsausta, ei vaan alussa vaan koko projektin ajan. On tärkeä aina kysyä asiakkaalta, että onhan tämä se oikea suunta. Tietysti meidän pitää jotenkin tuntea asiakkaan liiketoiminta niin ei näytetä ihan pelleiltä. (Haastateltava 1)

6.3 Haastateltavien esittämät ketterän ohjelmistokehityksen menestystekijät

Haastattelut haastateltavien asiantuntijoiden kanssa kriittisistä menestystekijöistä etenivät hyvin ja kriittiset menestystekijät oli käsitteenä tuttu suurimmalle osalle haastateltavista, mutta osalle kriittiset menestystekijät selitettiin seuraavasti: ”edellytykset, jotka projektilla tulisi vähintään olla, jotta projekti voisi edetä onnistuneesti ketterien periaatteiden mukaisesti”. Haastatteluissa nousivat esille erityisesti seuraavat kuusi menestystekijää, joita käsittelemme tarkemmin seuraavissa alaluvuissa.

6.3.1 Tiimin kyvykkyys

Haastatelluista asiantuntijoista jokainen nosti esille ihmistekijät ja tiimin yhteensopivuuden yhtenä tärkeimpänä kriittisenä menestystekijänä ketterän ohjelmistoprojektin onnistumiselle. Esimerkiksi Cockburnin (2001) sekä McLeodin & MacDonellin (2011) mukaan ihmistekijöistä tärkein tekijä on työntekijöiden kompetenssiosaamisen taso, mutta heidän mukaansa on myös tärkeä muistaa, että hyvin yhdessä toimiva tiimi on paljon tehokkaampi kuin ryhmä taitavia yksilöitä. Myös haastateltavien mukaan projektitiimin jäsenten kompetenssiosaamisen tulisi olla sillä tasolla, että esimerkiksi perustasoinen ohjelmointityö onnistuisi rutiininomaisesti. Haastateltavien mukaan tiimitasolla kompetenssiosaamisen tulisi lisäksi olla laaja-alaista. Tiimin jokaisen jäsenen tulisi kyetä esimerkiksi ohjelmoimaan ja testaamaan ainakin osaa kehitettävästä ohjelmistosta sekä ymmärtämään päällisin puolin asiakkaan liiketoimintaa ja sitä kautta asiakkaan tarpeita. Laaja-alaisen osaamisen avulla ehkäistään pullonkauloja projektin aikana. Pullonkaulojen ehkäisy korostuu siis erityisesti ketterissä ohjelmistoprojekteissa, koska projektin tarpeiden muuttuessa projektitiimin rakennetta ei keretä muuttamaan niin, että saatavilla olisi esimerkiksi huipputason ohjelmointitaitoa ja huipputason ohjelmistotestaustaitoa. Laaja-alaisen osaamisen lisäksi tiimistä pitäisi kuitenkin löytyä riittävää syväosaamista, jotta asiakkaan tarpeet pystytään saavuttamaan ja projektitiimin sisällä on monenlaisia eri näkemyksiä.

Haastateltavien mukaan kompetenssiosaamisen lisäksi tiimin jäseniltä vaaditaan tietynlaista persoonallisuutta ja asennetta, jotka mahdollistavat nopeatempoisen työskentelyn,

suoran ja avoimen kommunikaation ja jotka auttavat kestämään nopeita muutoksia ja sopeutumaan muuttuviin tilanteisiin. Moni haastateltava kertoi työntekijöiden hyvinä ominaisuuksina käsitteet itsejohtaminen, adaptiivisuus ja resilienssi.

> Niin muuttuvissa tilanteissa kun Scrum-projekteissa joutuu työskentelemään, niin pitää pystyä sietämään tietynlaista kaaosta ja oppia elämään sen kanssa ja jopa luomaan sellaista. Pitää tietyllä tavalla osata ja uskaltaa olla skeptinen ja haastaa kaikkia ratkaisuja, jotta löydetään paras mahdollinen ratkaisu. (Haastateltava 1)

> Kyllähän itsensä johtaminen on yksi avainsana tässä. Jos tiimille kerran annetaan valtuudet itseohjautuvasti suunnitella projektin toteutus itse ja tehdä ratkaisuja end-to-end, niin kyllä se vaatii tunnollista ja vastuuntuntoista työotetta kaikilta tiimin jäsenitä. (Haastateltava 6)

Monet haastateltavat nostivat esille myös henkilökemiat tiimihengen taustalla. Nopeatempoista ja nopeasti muuttuvaa tiimityötä tehdessä tiimin tiimihengen tulee olla erittäin hyvä. Monien haastateltavien mukaan tiimihengen tulee olla hyvin läheistä sekä kommunikaation suoraa ja avointa. Hyvän tiimihengen luomisessa projektin johdon lisäksi myös jokaisella tiimin jäsenellä on suuri vastuu. Hyvään tiimihenkeen vaikuttaa erityisesti henkilökemiat, yhteiset tavoitteet sekä avoin ja suora kommunikaatiokulttuuri.

> Itse on sillee ollu aika hyvässä asemassa, että on saanu aina työskennellä hyvissä tiimeissä sillä tavalla, että kaikkien mielipiteitä on kunnioitettu ja sillä tavalla on sitten noussut paljon hyviä ideoita esille. Tärkeää on sellanen meininki, että ei tuomita toisten mielipiteitä, joka johtaa muun muassa siihen, että jokaisella on rohkeutta kertoa ideoita. Tottakai henkilökemiat tuo oman mausteensa tähän. Kyllä se nyrkkisääntö on, että tiimi on aina tehokkaampi kuin kasa yksilöitä. Hyvä kommunikaatio on tän kaiken edellytys. (Haastateltava 3)

6.3.2 Johdon tuki

Kaikki haastateltavat nostivat esille myös johdon tuen. Heidän mukaansa ketteryydestä puhutaan yleisesti paljon ja kehotetaan itseohjautuvuuteen, mutta monesti johto sortuu liikaa pienimpienkin asioiden johtamiseen. Pienten asioiden johtaminen syö haastateltavien mukaan luottamusta ja kangistaa projektitiimin päätöksentekokykyä huomattavasti.

Boehm & Turner (2005) korostavat tutkimuksessaan, että ketterän ohjelmistoprojektin onnistuminen vaatii johdolta valtavasti luottamusta ja vastuun siirtämistä projektitiimille, mutta heidän mukaansa johdon aito tuki projektitiimiä kohtaan on erittäin tärkeä tekijä ketterän ohjelmistoprojektin onnistumiselle.

> Paljon puhutaan ketteryydestä suomalaisessa bisnesmaailmassa, mutta samat tyypit sitten sortuvatkin mikromanageroimaan ihan pienimpiäkin juttuja. Se syö luottamusta ja haaskaa sekä johtajan että työntekijän aikaa ja on kaukana ketteristä toimintatavoista. (Haastateltava 6)

> Kyllä meillä sellainen peukalosääntö on, että jos päätökset kyetään tehdä tiimin sisäisesti, niin sitten se tehdään. Koulutuksissa ollaan käyty sitä tosiasiaa läpi, että tiimin sisällä tehdään 80 % päätöksistä, ylemmäs viedään 15 % ja loput 5 % sitten sitäkin ylemmäs. (Haastateltava 7)

Johdon tuen tulisi haastateltavien mukaan olla mahdollistavaa ja valvovaa, eikä kontrolloivaa ja rajoittavaa. Aito johdon tuki on tavoitteiden asettamista, eikä kiinnittämistä huomiota siihen, miten näitä tavoitteita pyritään saavuttamaan projektitiimissä. Projektitiimillä pitäisi olla mahdollisuus valita itse menetelmät ja pyrkiä saavuttamaan tavoitteet alusta loppuun. Johdon tulee siis huolehtia, että projektitiimillä on kaikki tarvittavat resurssit, jotka projektin onnistumiseen tarvitaan.

> Johto toimii hyvin silloin kun scope ja budjetti on määritetty oikein. Ne on oikeestaan ne tärkeimmät asiat. Muuten johdon pitää sitten antaa tiimille työskentelyrauha ja malttaa olla mikromanageroimatta pieniä yksityiskohtia. (Haastateltava 5)

Johdon tulee myös huolehtia projektin ja projektitiimin riippuvuuksista muihin osastoihin, tiimeihin ja sidosryhmiin. Hyvä johtaminen mahdollistaa koko organisaation yhteisten prioriteettien asettamisen osastojen tai tiimien omien prioriteettien edelle. Oman osaston tai tiimin prioriteettien asettaminen koko organisaation prioriteettien edelle on haastateltavien mukaan hyvin usein yksi ketterien toimintatapojen vaikeimpia vastustajia. Ketterän ohjelmistoprojektin onnistumisen näkökulmasta katsottuna organisaation yhteisten prioriteettien arvostaminen oman osaston tai tiimin prioriteettien edelle

johtaa parempaan yhteistyöhön tiimien tai osastojen välille, joka taas mahdollistaa esimerkiksi työvoiman nopean uudelleenallokaation ja helpottaa kommunikaatiota.

> Kyllähän siiloutuminen hidastaa ja vaikeuttaa paljon esimerkiksi silloin, jos ja kun tarvitaan apua jossain projektissa muualta organisaatiosta. (Haastateltava 6)

6.3.3 Asiakasyhteistyö

Asiakasyhteistyöstä ja asiakkaan osallistamisesta ohjelmistoprojekteihin on tehty lukuisia tutkimuksia. Tutkimusten mukaan läheinen keskusteluyhteys ja asiakkaan osallistuminen kehitystyöhön johtaa poikkeuksetta parempiin tuloksiin ohjelmistoprojektissa. Esimerkiksi Lynch & Gregor (2003) toteavat tutkimuksessaan yksimielisesti, että läheisellä asiakasyhteistyöllä sekä asiakkaan osallistamisella on ohjelmistoprojektin onnistumisen kanssa selvä korrelaatio. Myös useimpien haastateltavien mukaan asiakasyhteistyö ja yleisesti asiakaslähtöinen toimintatapa on yksi ketterän ohjelmistokehityksen kulmakivistä. Tätä perusteltiin muun muassa sillä, koska toimitettavan tuotteen tai palvelun ideana on luoda asiakkaalle arvoa, tarpeen ja vaatimusten täytyy olla peräisin asiakkaalta. Asiakkaalle arvoa tuottavien tekijöiden selvittäminen onnistuu haastateltavien mukaan kaikista parhaiten pitämällä hyvä keskusteluyhteys asiakkaaseen koko projektin ajan.

> Tärkeintä on se, että keskitytään simppelisti niihin juttuihin mitkä tuo eniten arvoa asiakkaalle. Monesti koodatessa lähdetään vähän turhaan taiteilemaan ja lisäämää sinne sellasia juttuja mitkä on meidän mielestä siistejä, mutta asiakas ei niitä edes huomioi tai tarvi yhtään mihinkään. Helpoiten se selviää, kun viitsitään keskustella asiakkaan kanssa kunnolla ja koko ajan, että mitä ne tarvii. (Haastateltava 1)

> Kyllähän paras tilanne on se, että asiakas on koko ajan meidän kanssa työskentelemässä ja sanomassa heti, että mennäänkö oikeaan suuntaan vai ei. Totta kai on tärkeä alussa asiakkaan kanssa huolella specsata että mitä tarvitaan ja keskustella heidän vaatimuksista, ja meidän tehtävä on tässä sparrata ja tarjota meidän näkemystä, että mikä on fiksua tai mahdollista ja mikä välttämättä ei. Mutta kyllähän loppupeleissä asiakas päättää mitä me heille tehdään. (Haastateltava 2)

Kuten edellä suorassa lainauksessa mainittiin, asiakkaan osallistaminen kehitystyöhön mahdollistaa välittömän ja suoran palautteenannon, joka taas ehkäisee turhan työn tekemistä ja mahdollistaa olennaiseen keskittymisen. Haastateltavien mukaan asiakasyhteistyön pitäisi olla keskustelunomaista ja siinä määrin kommunikaation suoraa ja avointa, että toisen ideoiden ja suunnitelmien sparraaminen ja haastaminen on hyväksyttävää.

> No tärkeintä on keskittyä siihen, että tuotetaan asiakkaalle mahdollisimman paljon arvoa, eli keskitytään niihin juttuihin ohjelmistossa, mitkä asiakkaalle on kaikista tärkeimpiä, eikä sellaisiin mitkä meidän mielestä hienoimpia juttuja. Tämä tietysti vaatii suoraa ja avointa keskusteluyhteyttä asiakkaan kanssa, että asiakas voi heti sanoa jos mennään väärään suuntaan, heidän tarpeet on muuttunu tai jotain pitäisi muuttaa ohjelmistossa. Parashan se on jos asiakas on mukana koodaajien kanssa, mutta aina se ei onnistu. (Haastateltava 7)

6.3.4 Julkaisustrategia

Useimmat haastateltavat pitivät ketterän ohjelmistokehityksen lähtökohtana nopeasti toimittamista sekä ohjelmistokehitysprojektin toistavaa ja lisäävää luonnetta. Haastateltavat mainitsivat toimittamisen nopeuden olevan yksi ketterän ohjelmistokehityksen isoimmista hyödyistä. Esimerkiksi eräs haastateltava kertoi pitkään ohjelmistokehityskokemukseensa perustuen, että asiakkaalle on tärkeintä saada tuote mahdollisimman nopeasti ja projektitiimin tulisi keskittyä vain asiakkaalle arvoa tuottaviin tekijöihin, jotta asiakas saa haluamansa mahdollisimman nopeasti.

> Yleensä asiakkaalla on aina tarve saada ohjelmisto mahdollisimman nopeasti käyttöönsä. Tärkeintä tässä on se, että koko ajan keskustellaan asiakkaan kanssa ja toimitellaan heille tuoteversioita aina testattavaksi, jota sitten kehitellään aina iteratiivisesti eteenpäin asiakkaan palautteiden ja tarpeiden mukaan. (Haastateltava 1)

Erään haastateltavan mielestä toistava (engl. iterative) ja lisäävä (engl. incremental) julkaisustrategia on koko ketterän ohjelmistokehityksen ytimessä. Haastateltavien mukaan

iteraatiot on syytä pitää suhteellisen lyhyinä, jotta olennaiseen keskittyminen ja toimittamisen nopeus säilyvät.

> Vähän projektista ja sen koosta riippuen meillä iteraatiot kestävät about kahdesta viikosta kuukauteen. Joskus voi olla pitempiäkin. On hyvä, että iteraatiot on mahdollisimman tiukkia, jotta pysytään niin sanotusti asiassa eli kehitellään niitä juttuja, mitkä on asiakkaalle tärkeimpiä ja toimitetaan nopeasti. (Haastateltava 7)

6.3.5 Kommunikaatio

Kaikki haastateltavat nostivat esille kommunikaation merkityksen ketterän ohjelmistokehityksen kriittisenä menestystekijänä. Kuten myös Livermoren (2008: 4) tutkimuksen mukaan, myös tämän tutkimuksen haastateltavat näkivät kommunikaation eräänlaisena ketterän ohjelmistokehityksen kivijalkana. Kommunikaation tulee sekä Livermoren (2008: 4), että useimpien haastateltavien mukaan olla mahdollisimman suoraa, avointa ja mutkatonta projektitiimin sisällä, johdon kanssa sekä asiakkaan kanssa. Kommunikaation merkitys tuli esille erityisesti puhuttaessa projektitiimin tiimihengen ja asiakasyhteistyön merkityksestä ketterän ohjelmistoprojektin onnistumiselle.

> No yks tärkeä edellytys hyvälle tiimihengelle, mitä ei voi tarpeeksi korostaa, on ehdottomasti hyvä kommunikointi. Sen tulee olla mahdollisimman suoraa ja avointa. Meillä paljon korostetaan projekteissa sitä, että kaikki villeimmätkin ideatkin ja palautteet tulee sumeilematta kertoa heti ja välittömästi. (Haastateltava 2)

> Asiakasyhteistyö on toinen juttu, missä kommunikaation merkitys nousee myös vahvasti esille. Jos asiakkaan kanssa on sellanen keskusteluyhteys, että ei oikein saada irti mitä ne haluaa, niin vaikea on löytää yhteistä säveltä. Meillä toki on vastuu kysellä asiakkaalta niin paljon kuin mahdollista ja luoda se keskusteluyhteys. Kommunikaatio on sillee hyvän asiakasyhteistyön edellytys. (Haastateltava 2)

> Kyllä se vaatii hyvää keskusteluyhteyttä asiakkaan kanssa, kun halutaan tehdä heidän tarpeen mukaan ohjelmistoa. Kommunikaation pitää olla tosi rohkeaa, avointa ja suoraa. Ja samalla tavalla meidän koko organisaatiossakin ja projektitiimissä pitää olla edellytykset hyvälle kommunikaatiolle, jotta

> pystytään tekemään tehokkaasti töitä ja kaikki tietää mitä tapahtuu ja mikä kenenkin rooli on. (Haastateltava 8)

Etätyön ollessa pinnalla Covid-19 -viruksen takia, haastateltavat nostivat esille kommunikaation myös etäyhteyksiin liittyen. Haastateltavat mainitsivat, että kommunikointiin käytettävällä menetelmällä ei ole niinkään merkitystä, vaan sillä, että oleellinen tieto välittyy. Haastateltavat kuitenkin mainitsivat etäyhteyksien käytön lisääntymiseen liittyen, että sanaton viestintä vähenee, kun kommunikointia tehdään entistä enemmän esimerkiksi tekstiviestitse tai videokonferenssien välityksellä, mutta silti kommunikointi etäyhteyksien avulla on sujunut kokemusten mukaan pääosin hyvin.

6.3.6 Organisaatiokulttuuri

Noin puolet haastateltavista nostivat esille myös erittäin kiinnostavan ja monisäikeisen menestystekijän, organisaatiokulttuurin. Lyytisen & Rosen (2006) sekä Sheffieldin & Lemétayerin (2013) mukaan organisaatiokulttuuri käsitteenä on hyvin kompleksinen, ja jonka luominen tai muuttaminen tiettyyn suuntaan on vaikeaa erityisesti lyhyellä aikavälillä. Organisaatiokulttuurilla on tutkimusten mukaan valtava vaikutus ketterien ohjelmistokehitysprojektien onnistumiseen Gandomani, Zulzalil, Ghani, Sultan & Nafchi 2013; Sheffieldin & Lemétayerin 2013). Myös haastateltavien mukaan organisaatiokulttuurilla on suuri tai valtava vaikutus ketterien ohjelmistokehitysprojektien onnistumiseen.

Wanin ja Wangin (2010) mukaan ketterän ohjelmistokehityksen soveltamisen vaatimukset organisaatiokulttuurille voidaan jakaa kolmeen osaan, jotka ovat: ylitöitä koskeva kulttuuri, luottamus organisaatioon ja työntekijöiden keskinäisen yhteistyökulttuurin taso. Haastateltavien mukaan organisaatiokulttuuria on suhteellisen vaikea erottaa päivittäisessä tekemisessä, mutta monet haastateltavat kertoivat hyvän organisaatiokulttuurin esimerkiksi tunteen, että heidän päätöksiinsä luotetaan, eikä niistä tuomita, vaikka niistä johtuisi epäonnistumisia tai vastoinkäymisiä.

> Projektitiimillä ja jokaisella jäsenellä pitää olla sellanen tunne, että heihin luotetaan ja on lupa kokeilla kaikkea pelkäämättä epäonnistumisia. Puhutaan sellaisesta käsitteestä ku growth mindset, eli koodaaja testaa, mittaa ja oppii. Eli pitää rohkeasti kokeilla, ja sitten jos homma ei toimi niin muuttaa

> tuotetta niin kauan että homma toimii. Datasta katsotaan että mikä toimii ja mikä ei ja menee sen mukaan. (Haastateltava 8)

Haastateltavien mukaan organisaatiokulttuurin yksi ketterään ohjelmistokehitykseen läheisesti liittyvä ulottuvuus on oppiva organisaatio -käsite, jolla tarkoitetaan sellaista organisaatiota, jossa kokeillaan erilaisia ratkaisuja ja tällä tavoin löydetään jopa tahallaan epäonnistumalla parhaat ratkaisut. Iteratiivinen ja inkrementaalinen lähestymistapa on ketterän ohjelmistokehityksen yksi kulmakivistä, joka tietyllä tavalla edellyttää kokeilevaa kulttuuria sekä palautteista ja epäonnistuneista kokeiluista oppimista. Suurin osa haastateltavista mainitsi myös data-analytiikan merkityksen ketterässä ohjelmistokehityksessä. Dataa seuraamalla ja erilaisia mittareita käyttämällä voidaan vertailla erilaisia tuoteratkaisuja ja valita niistä asiakkaalle paras. Luonnollisesti erilaisia tuoteratkaisuja täytyy kokeilukulttuurin mukaisesti ensin luoda, ennen kuin vertailua voidaan suorittaa.

> Kyllä koko organisaatiossa pitää olla sellainen jatkuvan oppimisen kulttuuri, että ei jarrutella kun epäonnistutaan. Pitää oppia epäonnistumisesta, katsoa mikä meni pieleen, ja kehittää ohjelmaa sitä mukaa. Siinä mielessä ohjelmistoala on turvallinen ala kokeilla ja oppia, että ei menetetä kuin ainoastaan muutama tunti koodarin työaikaa. Jollain muilla teollisuudenaloilla specsaamisen tulee olla alussa paljon tarkempaa, koska tuotantoketjussa takaisinpalaaminen on paljon kalliimpaa. (Haastateltava 7)

> Käytännössä menee niin, että toteutetaan eka tuoteversio hyvin nopeasti ja katsotaan sitten mitä data näyttää tai mitä asiakas on mieltä. Sen mukaan sitten lisätään tai muutetaan komponentteja. (Haastateltava 5)

6.4 Haastateltavien esittämät tekijät, joilla on vähäinen tai olematon vaikutus ketterän ohjelmistokehitysprojektin menestykseen

Ketterän ohjelmistokehityksen kriittisten menestystekijöiden lisäksi haastateltavilta kysyttiin myös tekijöitä, joilla on vähäinen tai olematon vaikutus ketterän ohjelmistoprojektin onnistumiselle. Tällaisia tekijöitä kartoitettiin esittelemällä alan kirjallisuuden mukaiset hypoteettiset menestystekijät haastateltaville, jonka jälkeen haastateltavat punnitsivat tekijöiden kriittisyyttä omien kokemustensa pohjalta. Seuraavaksi

tarkastelemme tarkemmin näitä vähäisen tai olemattoman vaikutuksen tekijöitä, jotka mainittiin monissa eri haastatteluissa.

6.4.1 Projektin luonne

Projektin luonne on yksi Chowin & Caon (2008) määrittelemä hypoteettinen kriittinen menestystekijä. Projektin luonne -menestystekijällä tarkoitetaan sitä, että suoritettavan projektin luonteen tulisi olla ei-elämäkriittinen (engl. non-life critical), jotta projektin suorittamiseksi voisi onnistuneesti soveltaa ketterän ohjelmistokehityksen periaatteita. Elämäkriittisellä projektin luonteella tarkoitetaan käytännössä projektia, jonka tulee onnistua täydellisesti. Elämäkriittinen projektin luonne voi olla esimerkiksi sellaisen projektin kehitystyössä, jossa ohjelmisto tulee valmistuttuaan käytettäväksi lennonjohdossa tai terveydenhoitoalalla. Chowin & Caon (2008) mukaan ketterän ohjelmistoprojektin luonteen hypoteettisesti ei saisi olla elämäkriittinen.

Haastateltavat nostivat projektin luonne -menestystekijän kuitenkin esille siinä mielessä, että heidän mielestään ei ole mitään syytä olla käyttämättä ketteriä ohjelmistokehitysmenetelmiä elämäkriittisissä projekteissa. Kuitenkin elämäkriittisissä projekteissa projektin tavoitteen voidaan olettaa olevan tarkemmin määritelty ja liiketoimintaympäristön vaatimusten voidaan olettaa muuttuvan paljon vähemmän, kuin esimerkiksi tavallisilla B2B-markkinoilla. Toisin sanoen projektin ympäristön voidaan siis olettaa olevan stabiilimpi.

> Vaikka se aika monesti meneekin niinkin, että mitä isompi tai tärkeämpi projekti on, sitä vähemmän muutoksia projektin aikana tulee, niin silti ketterillä menetelmillä saavutetaan esimerkiksi toimitusnopeutta, olennaiseen keskittymistä ja yleisesti hyvää projektinhallintaa. (Haastateltava 3)

> En ite ainakaan näe mitään syytä miksi tällaisia elämäkriittisiä hankkeita ei voisi tehdä ketterästi. Voi tosin olettaa, että alussa specsataan tarkemmin ja muutoksia vaatimuksissa tulee vähemmän kuin ns normaaleissa projekteissa. Mutta onhan ketterästi tekemisessä muitakin etuja kuin muutoksiin reagoiminen. (Haastateltava 8)

Haastateltavien mukaan nopea reagointi liiketoimintaympäristössä tapahtuviin muutoksiin on ketterien ohjelmistokehitysmenetelmien ehkä suurin vahvuus, mutta syitä kyseisten menetelmien käyttöön on muitakin. Moni haastateltava ajatteli ketterien ohjelmistokehitysmenetelmien esimerkiksi auttavan hallitsemaan projektia paremmin ja keskittymään asiakkaalle arvoa tuottaviin toimintoihin, vaikka asiakkaan tarpeiden voidaankin olettaa muuttuvan harvemmin. Myös toimitusnopeus ja asiakasyhteistyö on yleisesti ketterissä ohjelmistokehitysprojekteissa kehittyneempää kuin esimerkiksi perinteisillä ohjelmistokehitysmenetelmillä toteutetuissa projekteissa. Haastateltavat eivät siis nähneet mitään syytä olla käyttämättä ketteriä ohjelmistokehitysmenetelmiä myös elämäkriittisen projektin luonteen omaavissa projekteissa.

6.4.2 Tiimin maantieteellinen jakauma

Misran ym. (2009) tutkimuksen mukaan fyysisesti samassa paikassa sijaitseva projektitiimi on hypoteettinen kriittinen menestystekijä ketterälle ohjelmistokehitysprojektille. Projektitiimin työskentely samassa tilassa helpottaa sekä alan kirjallisuuden, että haastateltavien mukaan tiimihengen muodostumista tiimin jäsenten välille, joka taas puolestaan parantaa kommunikaatiota. Haastateltavien mukaan yksi suurimmista miinuksista tiimin samassa paikassa työskentelyssä kommunikaatiohaasteiden lisäksi on epävirallisten kohtaamisten, kuten esimerkiksi lounas- ja kahvitaukojen puuttuminen. Tällaisten epävirallisten kohtaamisten mainittiin olevan erinomaisia hetkiä jakaa niin sanottua hiljaista tietoa tai epävirallista informaatiota tiimin jäsenten välillä.

> Kyllähän se tiimihenkeä ja kommunikaatiota parantaa, kun porukka on samassa tilassa, mutta varmaan suurin osa meidänkin työntekijöistä koodaa mieluummin kotona rauhassa. Nettiyhteydet kun vielä kehittyy entisestään niin sekään ei sitten tuota hankaluuksia. (Haastateltava 5)

> Kyllä meillä kaikki on etänä suurimman osan työajasta. Ei sitä enää edes ajattele, kun ollaan niin totuttu kommunikoimaan etänä. Meille on tärkeintä se, että minkälaisen työskentelyn meidän työntekijät kokee parhaaksi. (Haastateltava 1)

Useat haastateltavat kuitenkin mainitsivat, että modernien viestintäteknologioiden kehittyminen mahdollistaa työn tekemisen etänä mainiosti, eikä kommunikaatio tai hiljaisen tiedon välittyminen ole kärsinyt verrattuna fyysisesti samassa tilassa työskentelyyn. Moni haastateltava nosti myös esille sen, että työn tekeminen on rauhallisempaa etänä, kuin esimerkiksi meluisessa avokonttorissa.

Yhteenvetona todettakoon, että tässä tutkimuksessa toteutettujen teemahaastatteluiden tulosten mukaan tiimin maantieteellisellä sijainnilla ei havaittu olevan ainakaan kovin suurta negatiivista vaikutusta ketterän ohjelmistokehitysprojektin onnistumiselle.

6.5 Haastattelutulosten vertailu

Seuraavassa taulukossa (ks. taulukko 3) esitellään haastateltavien kertomat kriittiset menestystekijät sekä tekijät, joilla on vähäinen tai olematon vaikutus ketterän ohjelmistokehitysprojektin onnistumiselle.

Kuten aiemmin jo mainittiin, ketterää ohjelmistokehitystä voi tehdä monella eri tapaa, jolloin todennäköisesti myös kriittisten menestystekijöiden tärkeysjärjestykset vaihtelevat. Haastatteluissa nousi kuitenkin ilmi menestystekijöitä, jotka haastateltavat kokevat olevan kriittisiä kaikille ketterille ohjelmistokehitysprojekteille.

Taulukko 3. Yhteenveto haastattelujen tuloksista

Haastateltava	Kriittinen menestystekijä	Tekijä, jolla ei ole vaikutusta
H1	• Tiimin kyvykkyys • Johdon tuki • Asiakasyhteistyö ja asiakkaan osallistuminen • Kommunikaatio • Julkaisustrategia ja toimittamisen nopeus	• Tiimin maantieteellinen jakauma
H2	• Tiimin kyvykkyys • Johdon tuki • Asiakasyhteistyö ja asiakkaan osallistuminen • Kommunikaatio • Julkaisustrategia ja toimittamisen nopeus	
H3	• Tiimin kyvykkyys • Asiakasyhteistyö ja asiakkaan osallistuminen • Kommunikaatio • Julkaisustrategia ja toimittamisen nopeus	• Projektin luonne • Tiimin maantieteellinen jakauma
H4	• Tiimin kyvykkyys • Asiakasyhteistyö ja asiakkaan osallistuminen • Kommunikaatio	
H5	• Tiimin kyvykkyys • Johdon tuki • Asiakasyhteistyö ja asiakkaan osallistuminen • Kommunikaatio • Julkaisustrategia ja toimittamisen nopeus • Organisaatiokulttuuri	• Tiimin maantieteellinen jakauma
H6	• Tiimin kyvykkyys • Johdon tuki • Asiakasyhteistyö ja asiakkaan osallistuminen • Kommunikaatio • Julkaisustrategia ja toimittamisen nopeus	
H7	• Tiimin kyvykkyys • Johdon tuki • Asiakasyhteistyö ja asiakkaan osallistuminen • Kommunikaatio • Organisaatiokulttuuri	• Projektin luonne
H8	• Tiimin kyvykkyys • Asiakasyhteistyö ja asiakkaan osallistuminen • Kommunikaatio • Julkaisustrategia ja toimittamisen nopeus • Organisaatiokulttuuri	• Projektin luonne

6.5.1 Mainituimmat kriittiset menestystekijät – tiimin kyvykkyys, asiakasyhteistyö ja kommunikaatio

Haastatteluissa esiin nousseista ketterän ohjelmistokehityksen kriittisistä menestystekijöistä mainituimmat olivat tiimin kyvykkyys, asiakasyhteistyö ja kommunikaatio, jotka mainittiin jokaisen haastateltavan toimesta. Tiimin kyvykkyydellä tarkoitetaan siis projektitiimin jäsenten kompetenssiosaamista ja henkilökohtaisia ominaisuuksia. Haastateltavat mainitsivat, että tiimistä tulisi löytyä syvää kompetenssiosaamista, jotta haastaviin vaatimuksiin esimerkiksi ohjelmoinnin -ja testauksen parissa voitaisiin vastata. Syvän osaamisen lisäksi projektitiimistä tulisi löytyä myös laajaa osaamista, jotta projektissa vältyttäisiin mahdollisilta pullonkauloilta, eli huonolta resurssien allokoinnilta. Edellä mainitun kompetenssiosaamisen lisäksi henkilökohtaiset ominaisuudet nostettiin myös esille lukuisissa haastatteluissa, joilla tarkoitettiin muun muassa itseohjautuvaa ja ennakkoluulotonta työotetta, joka on erään haastateltavan mukaan jopa vaatimuksena kyseiseen yritykseen rekrytoidessa. Haastateltavat painottivat, että tärkeää on myös tiimin yhteistyön taso, koska monen haastateltavan mukaan hyvin yhdessä työskentelevä tiimi on paljon arvokkaampi kuin joukko taitavia yksilöitä.

Asiakasyhteistyön merkitystä korostettiin myös jokaisen haastateltavan toimesta. Asiakasyhteistyön merkitystä perusteltiin haastateltavien toimesta muun muassa sillä, koska toimitettavan tuotteen tai palvelun ideana on luoda asiakkaalle arvoa, tarpeen ja vaatimusten täytyy olla peräisin asiakkaalta. Läheinen asiakasyhteistyö varmistaa perusteellisen tarpeiden ja vaatimusten selvittäessä projektin vaatimusmäärittelyvaiheen lisäksi esimerkiksi myös projektin kehitystyön aikana.

Tiimin kyvykkyyden ja asiakasyhteistyön lisäksi tärkeimpänä ja mainituimpana kriittisenä menestystekijänä mainittiin kommunikaatio, jonka tärkeyttä perusteltiin monen haastateltavan sanoin ketterän ohjelmistokehityksen kulmakivenä sekä muiden menestystekijöiden edellytyksenä. Esimerkiksi projektitiimin ja asiakkaan välisen asiakasyhteistyön edellytyksenä on suora ja avoin kommunikaatio, jotta asiakkaan kaikki tarpeet saadaan käsiteltyä. On myös tärkeä muistaa, että kommunikaatiolla ei viitata ainoastaan projektitiimin sisäiseen kommunikaatioon, vaan myös kommunikaatioon esimerkiksi

projektitiimin ja asiakkaan välillä, tai kommunikaatioon projektitiimin ja organisaation johdon välillä.

6.5.2 Muut haastatteluissa mainitut kriittiset menestystekijät

Haastatteluissa seuraavaksi mainituin ketterän ohjelmistokehityksen kriittinen menestystekijä oli ketterän ohjelmistokehityksen mukainen julkaisustrategia, joka mainittiin kuuden haastateltavan toimesta. Julkaisustrategialla tarkoitetaan tässä yhteydessä ketterän ohjelmistokehityksen mukaista julkaisustrategiaa, jonka pääasiallisina elementteinä on strategian toistava ja lisäävä luonne. Julkaisustrategian tärkeyttä perusteltiin haastateltavien toimesta muun muassa sillä, että toistavan ja lisäävän luonteen ansiosta projektitiimin niin sanotusti täytyy keskittyä vain arvoa tuottaviin toimiin, koska kyseisen strategian mukaisesti kehitystyössä on edettävä hyvin nopeasti. Sprinttejä toteutetaan peräjälkeen ja tuotetta parannetaan sekä toiminnallisuuksia lisätään jokaisessa sprintissä. Kyseiseen julkaisustrategiaan kuuluu läheisesti myös nopea toimittaminen, joka haastateltavien mukaan jo itsessään luo arvoa asiakkaalle.

Seuraavaksi eniten mainittu ketterän ohjelmistokehityksen kriittinen menestystekijä oli johdon tuki projektille, jonka mainitsi kahdeksasta haastateltavasta viisi haastateltavaa. Johdon tuella haastateltavat tarkoittivat pääasiassa johdon luottamusta ja vastuunantoa projektia sekä projektitiimiä kohtaan. Haastateltavien mukaan johdon tulee sitoutua projektiin ja vaalia itseohjautuvaa yrityskulttuuria sekä toimia enemmänkin mahdollistavana tekijänä, eikä rajoittavana tekijänä.

Myös organisaatiokulttuurin vaikutusta painotettiin ketterän ohjelmistokehityksen kriittisenä menestystekijänä. Organisaatiokulttuuri mainittiin yhteensä kolmen haastateltavan toimesta kriittisenä menestystekijänä. Organisaatiokulttuuri on haastateltavien mukaan vaikeasti määriteltävissä ja sen vaikutus projekteille vaikeasti rajattavissa, mutta vaikutus kuitenkin on olemassa. Haastateltavien mukaan organisaatiokulttuuri toimii menestystekijänä ketterälle ohjelmistoprojektille erityisesti silloin, kun se kannustaa ketterään, itseohjautuvaan ja matalahierarkkiseen kulttuuriin, jossa uskalletaan edetä oppimisen ja kokeilemisen kautta asiakkaan kokema arvo pääasiallisena ajurina.

Organisaatiokulttuurin kehittäminen tai vaaliminen nähtiin haastateltavien mukaan hyvin vaikeana erityisesti lyhyellä aikavälillä, mutta tärkeänä tavoitteena pitkällä aikavälillä.

Edellä mainittujen lisäksi haastateltavien kanssa käytiin läpi myös sellaisia kriittisiä menestystekijöitä, joilla ei heidän mukaansa ole vaikutusta ketterän ohjelmistoprojektin onnistumiseen. Tällaisia tekijöitä olivat tiimin maantieteellinen jakauma ja projektin luonne, jotka molemmat mainittiin kolmen haastateltavan toimesta. Haastateltavat katsoivat, että erityisesti nykyaikana etäyhteyksien kehityttyä tiimin maantieteellisellä jakaumalla ei katsottu olevan vaikutusta ketterän ohjelmistoprojektin onnistumiseen. Projektin luonteella ei myöskään katsottu olevan merkitystä ketterän ohjelmistoprojektin onnistumiseen, toisin sanoen haastateltavien mukaan ketterät ohjelmistokehitysmenetelmät toimivat riippumatta projektin luonteesta.

6.6 Ketterän ohjelmistokehityksen malli

Tässä luvussa esitellään tutkielman löydösten pohjalta kehitelty malli tai työkalu, joka on suunniteltu erityisesti käytännön avuksi ketteriä ohjelmistokehitysmenetelmiä soveltaville organisaatioille. Ketterä ohjelmistokehitys on toistava ja lisäävä ohjelmistokehitysmenetelmä, jota voidaan soveltaa monilla eri tavoin. Alla olevassa kuvassa (ks. kuvio 8) on kuvattu kuuden askeleen ketterä projektikaavio tehdä ketterää ohjelmistokehitystä. Kyseinen projektikaavio on vain suuntaa antava esimerkki yhdestä tavasta viedä eteenpäin ketterää ohjelmistokehitysprojektia. Tässä luvussa käydään läpi yksityiskohtaisesti ketterän ohjelmistokehityksen kriittisten menestystekijöiden avulla, mitä ketterässä ohjelmistoprojektissa tulisi varmistaa, jotta projektilla olisi parhaat mahdollisuudet onnistua.

Kuvio 6. Ketterän ohjelmistokehityksen projektikaavio kriittisillä menestystekijöillä (mukaillen Anurina, 2019)

Seuraavaksi käydään läpi kyseisen projektikaavion (ks. kuvio 8) kukin vaihe ja niissä erityisesti korostuvat kriittiset menestystekijät.

6.6.1 Yksityiskohtainen vaatimusmäärittely

Yksityiskohtaisessa vaatimusmäärittelyssä käydään tiiviissä yhteistyössä asiakkaan kanssa läpi vaatimukset ja tavoitteet tuotteelle, sekä erityisesti ketterän ohjelmistokehityksen julkaisustrategian mukaan ne tekijät, jotka tuovat eniten arvoa asiakkaalle. Yksityiskohtaisessa vaatimusmäärittelyssä kommunikaation tulisi olla suoraa ja avointa sekä asiakasyhteistyön hedelmällistä, jotta kaikki vaatimukset ja tarpeet saadaan projektitiimin tietoon. Projektitiimiltä vaaditaan yksityiskohtaisessa vaatimusmäärittelyssä täten myös kyvykkyyttä ymmärtää ohjelmistokehityksen mahdollisuudet vastata asiakkaan tarpeisiin sekä kykyä ymmärtää asiakkaan liiketoimintaa, jotta projektitiimi ymmärtäisi asiakkaan todelliset tarpeet ja arvoa tuottavat tekijät tuotteessa. Yksityiskohtaisessa

vaatimusmäärittelyssä korostuu myös johdon tuen ja sitoutumisen merkitys, koska projektitiimillä on välttämätöntä olla tiedossaan esimerkiksi ne resurssit ja rajoitteet, jotka projektin tai organisaation johto asettaa projektille. Lisäksi, mikäli ohjelmistoa tarjoavan yrityksen organisaatiokulttuurissa on syvälle juurtunut tapa tehdä vaatimusmäärittelyä keskustelevasti ja hakea asiakkaan kanssa yhdessä ratkaisuja perusteellisesti ennen kehitystyöhön siirtymistä, on projektitiimillä erinomaiset lähtökohdat onnistua vaatimusmäärittelyssä. (Haastateltava 3; Haastateltava 5; Haastateltava 8)

6.6.2 Suunnittelu

Suunnitteluvaiheessa projektitiimi suunnittelee ohjelmiston arkkitehtuuria yksityiskohtaisessa vaatimusmäärittelyssä selvitettyjen vaatimusten ja tavoitteiden pohjalta. Tässä vaiheessa projektitiimiltä vaaditaan kompetenssia suunnitella ja kykyä aikatauluttaa kehitystyötä. Tiimin yhteensopivuudella ja kommunikaatiolla on myös suuri merkitys suunnitellessa projektitiimin sisäisesti kehitystyötä niin, että projekti etenee ketterän ohjelmistokehityksen mukaisen julkaisustrategian mukaan nopeasti ja pullonkauloja välttämällä. Kokeilunhaluisen ja ketterän organisaatiokulttuurin mukaisesti projektitiimillä tulisi olla aikataulussa tilaa edetä myös kokeilun ja oppimisen kautta, jotta asiakkaalle arvoa eniten tuottavat asiat löydetään. Myös johdon tuella on tähän suuri merkitys, mikäli johto tukee projektitiimiä parhaansa mukaan, projektitiimillä on rohkeutta ja turvallisuutta kokeilla rohkeitakin lähestymistapoja asiakkaan ongelmien ratkaisemiseksi. (Haastateltava 3; Haastateltava 5; Haastateltava 8)

6.6.3 Kehittäminen ja testaus

Itse kehitystyössä ohjelmistoa koodataan ja yleensä myös testataan samaan aikaan, joten kehitys- ja testausvaiheessa pätevät Vaativa ohjelmistokehitys- ja testaustyö vaatii projektitiimiltä luonnollisesti kyvykkyyttä suorittaa vaativia ohjelmistokehitystehtäviä sekä myös hyvää yhteistyötä ja kommunikaatiota, jotta kehitystyö etenee sujuvasti ja pullonkauloilta vältytään. Myös asiakasyhteistyön ja kommunikaation merkitys korostuu kehitystyössä, koska usein kehitystyössä nousee esiin tarkentavia kysymyksiä asiakkaan

suuntaan. Ketterän ohjelmistokehityksen mukaisen toistavan ja lisäävän julkaisustrategian mukaan kehitystyössä projektitiimin on ensiarvoisen tärkeä pitää mielessä keskittyminen ensiarvoisesti vain asiakkaalle eniten arvoa tuottaviin toimiin. (Haastateltava 3; Haastateltava 5; Haastateltava 8)

6.6.4 Julkaisu

Julkaisuvaiheessa kyseisen iteraation tuotos toimitetaan asiakkaalle, jonka jälkeen kyseisessä iteraatiossa tuotteeseen ei enää tehdä muutoksia. Julkaisuvaiheessa korostuu erityisesti ketterän ohjelmistokehityksen mukainen julkaisustrategia, jonka mukaan ketterän ohjelmistokehityksen yksi kulmakivistä on toimittaa tuote asiakkaalle mahdollisimman nopeasti resurssien säästämiseksi. (Haastateltava 3; Haastateltava 5; Haastateltava 8)

6.6.5 Katselmus

Katselmus -vaiheessa läpikäydyn iteraation kehitystyön kulku ja erityisesti iteraation tuotos käydään yhdessä läpi koko projektitiimin ja asiakkaan edustajan kanssa. Katselmuksessa arvioidaan iteraation tavoitteiden täyttymistä ja aloitetaan seuraavan iteraation ideointi, jota sitten jatketaan seuraavan iteraation ensimmäisessä vaiheessa, joka siis on yksityiskohtainen vaatimusmäärittely. Katselmuksessa kommunikaation tulee olla suoraa ja avointa projektitiimin ja asiakkaan välillä, eli toisin sanoen asiakasyhteistyön tulee olla hedelmällistä, jotta asiakkaalta saadaan vastaanotettua palautetta kehitystyön suunnasta tai tavoitteiden muuttumisista sekä palautteen käsittely on asianmukaista ja etenee hyvässä yhteistyössä tavoitteena ratkaista asiakkaan ongelmat. (Haastateltava 3; Haastateltava 5; Haastateltava 8)

7 Diskussio

Tämän tutkielman tavoitteena oli selvittää teemahaastattelujen avulla ketterän ohjelmistokehityksen kriittisiä menestystekijöitä sekä vertaamalla löydöksiä aiempiin tutkimuksiin ketterän ohjelmistokehityksen kriittisistä menestystekijöistä. Tavoite saavutettiin, ja teemahaastattelujen avulla löytyi yhteensä kuusi kriittistä menestystekijää, jotka osaltaan vahvistavat aiempien alan tutkimusten johtopäätöksiä, mutta tuovat osaltaan myös uusia havaintoja tutkimuskenttään, joihin palaamme tarkemmin seuraavissa luvuissa. Kriittisillä menestystekijöillä tarkoitetaan siis sellaisia onnistumisen edellytyksiä tai ominaisuuksia, jotka projektitiimillä tulisi ainakin olla käytössään, jotta ketterä ohjelmistokehitysprojekti olisi onnistunut. Tämän tutkimuksen tutkimusongelmana oli:

- Mitkä ovat ketterän ohjelmistokehitysprojektin onnistumisen kriittiset menestystekijät?

Teemahaastattelujen avulla löydettiin yhteensä kuusi ketterän ohjelmistokehityksen kriittistä menestystekijää: tiimin kyvykkyys, johdon tuki, asiakasyhteistyö, kommunikaatio, julkaisustrategia ja organisaatiokulttuuri. Haastateltavien mainitsemia tekijöitä, joilla on olematon tai vähäinen vaikutus ketterän ohjelmistokehitysprojektin onnistumiselle löydettiin kaksi: tiimin maantieteellinen jakauma ja projektin luonne. Kriittisten menestystekijöiden löytämisen lisäksi tutkimuksen hyötyä erityisesti käytännön näkökulmasta pyrittiin nostamaan kehittämällä ketterän ohjelmistokehityksen malli, johon ketterää ohjelmistokehitystä soveltavat organisaatiot voivat peilaamaan omaa toimintaansa ja tätä kautta auttavat organisaatioita kehittämään omaa toimintaansa ketterämpään suuntaan. Seuraavaksi kyseisiä löydöksiä verrataan aikaisempiin aiheeseen liittyvän kirjallisuuden löydöksiin.

7.1 Tulosten vertailu aikaisempiin löydöksiin

Tässä kohtaa on tärkeä jälleen kerran nostaa esille, että ketterän ohjelmistokehityksen kriittisten menestystekijöiden painotukset ja tärkeysjärjestykset ovat hyvin

projektikohtaisia. Menestystekijöiden kriittisyys ja niiden tärkeysjärjestys projektin onnistumiselle riippuu hyvin paljon erityisesti projektin tavoitteista, vaatimuksista ja liiketoimintaympäristöstä.

Tiimin kyvykkyyden kriittisyydestä menestystekijänä ketterän ohjelmistokehitysprojektin onnistumiselle on monia eri tuloksia. Cockburnin & Highsmithin (2001) tutkimuksen mukaan tiimin kyvykkyys on tärkein, ja Chowin & Caon (2008) mukaan toiseksi tärkein kriittinen menestystekijä ketterän ohjelmistokehitysprojektin onnistumiselle ketterien ohjelmistokehitystekniikoiden hallitsemisen ohessa. Myös Aldahmashin ym. (2017) sekä Tam ym. (2020) kertovat tutkimuksissaan, että tiimin kyvykkyys on kriittinen menestystekijä ketterän ohjelmistokehitysprojektin onnistumiselle. Tässä tutkielmassa tarkastelluista tutkimuksista ainoastaan Stankovic ym. (2013) osoittaa määrällisessä tutkimuksessaan, että tiimin kyvykkyydellä on vain vähäinen vaikutus ketterän ohjelmistoprojektin onnistumiselle, ja Françan ym. (2010) sekä Misran ym. (2009) tutkimusten mukaan tiimin kyvykkyys on kriittisen menestystekijän sijaan vain tärkeä menestystekijä. Suurin osa alan arvostetuista tutkimuksista on siis kuitenkin päätynyt siihen tulokseen, että tiimin kyvykkyys on vähintään tärkeä menestystekijä, ellei kriittinen menestystekijä ketterän ohjelmistoprojektin onnistumiselle. Myös Beldarrain (2019) osoittaa tuoreessa tutkimuksessaan, että tiimin kyvykkyydellä ja yhteensopivuudella on valtavan suuri merkitys ketterässä ohjelmistoprojektissa onnistumiselle, ja siihen voidaan vaikuttaa suuresti onnistuneella rekrytoinnilla ja perehdytyksellä organisaation sisäisesti. Tämän tutkielman ja alan aiempien tutkimusten löydösten välillä suurimman eron tiimin kyvykkyyttä tarkastellessa tekee kuitenkin se, että haastateltavat korostivat tiimin kompetenssiosaamisen lisäksi tiimin yhteishengen merkitystä hyvin paljon. Haastateltavat perustelivat tätä sillä, että tiimi tietysti rakentuu osaavista työntekijöistä, mutta hyvin monet asiantuntijat kertoivat, että hyvin yhteen toimiva tiimi on paljon tehokkaampi kuin huonosti kommunikoivat ja huonon yhteishengen omaavat yksilöt. Tämä löydös näihin kyseisiin alan tutkimuksiin verrattuna on siis uusi löydös ketterän ohjelmistoprojektin kriittisten menestystekijöiden tutkimuskenttään.

Johdon tuesta ja sitoutumisesta ketterään ohjelmistoprojektiin puhuttaessa kriittisenä menestystekijänä täytyy ottaa huomioon, että johdon tuen muoto ja vaikutus

vaihtelevat paljon riippuen projektin tavoitteista, vaatimuksista ja liiketoimintaympäristöstä. (Chow & Cao, 2008) Keskimäärin johdon tukea ja sitoutumista ei olla pidetty kriittisenä, eikä edes tärkeänä menestystekijänä alan arvostetuissa tutkimuksissa. Johdon tuki ja sitoutuminen ketterälle ohjelmistokehitysprojektille sekoittuukin hyvin usein organisaatiokulttuuriin aiheesta puhuttaessa, joka puolestaan käsitetään alan tutkimuksissa tärkeänä tai kriittisenä tekijänä ketterän ohjelmistoprojektin onnistumiselle. Esimerkiksi Misra ym. (2009) pitävät organisaatiokulttuuria kriittisenä menestystekijänä, mutta ei johdon tukea ja sitoutumista. Alan arvostetuista tutkimuksista vain Aldahmash ym. (2017) pitävät johdon tukea ja sitoutumista kriittisenä menestystekijänä. Johdon tuesta ja sitoutumisesta puhuttaessa tämän tutkielman löydökset siis vahvistavat Aldahmashin ym. (2017) löydöksiä, mutta on siis eri mieltä muiden tässä tutkielmassa esiteltyjen arvostettujen alan tutkimusten kanssa.

Asiakasyhteistyötä ja asiakkaan osallistumista pidettiin kaikissa tässä tutkielmassa esitellyissä tutkimuksissa kriittisenä, tai vähintään tärkeänä menestystekijänä ketterän ohjelmistoprojektin onnistumiselle. Esimerkiksi Misra ym. (2009) mainitsevat tutkimuksessaan, että asiakasyhteistyön ja asiakkaan osallistumisen merkitys ketterälle ohjelmistoprojektille on helppo hyväksyä kriittisenä menestystekijänä, koska ketterän ohjelmistoprojektityöskentelyn yhtenä keskeisenä ideana on keskustella jatkuvasti ja kehittää tuotetta yhdessä asiakkaan kanssa pohjautuen asiakkaan muuttuviin kehitysehdotuksiin tai tarpeisiin kehitystyön edetessä. Myös tämän empiirisen tutkimuksen mukaan asiakasyhteistyö on kriittinen menestystekijä juuri niillä perusteluilla, että ketterän ohjelmistokehityksen perusideana on vastata muutoksiin nopeasti ja jopa luoda muutosta, jotka siis ovat lähtöisin asiakkaalta, heidän tarpeistaan ja niiden alituisista muutoksista.

Kaikki haastateltavat nostivat esille kommunikaation merkityksen muiden menestystekijöiden kulmakivenä. Kommunikaation kriittisyyttä ketterän ohjelmistoprojektin onnistumiselle perusteltiin muun muassa sillä, että jos kommunikaatio on huonoa, esimerkiksi tiimin jäsenet eivät pahimmassa tapauksessa tiedä mitä projektissa pitää tehdä missäkin vaiheessa. Myös tässä tutkielmassa esiteltyjen tutkimusten mukaan kommunikaatio on eräänlainen avaintekijä muille menestystekijöille. Esimerkiksi Misra ym. (2009) mainitsevat, että kommunikaatio ikään kuin sisältyy muihin menestystekijöihin, koska se on

niiden edellytys. Kuitenkin esimerkiksi Aldahmashin ym. (2017) tutkimuksessa kommunikaatio on mainittu itsessään kriittisenä menestystekijänä.

Julkaisustrategiaa pidetään kriittisenä menestystekijänä kaikissa aiemmin esitellyissä tutkimuksissa, paitsi Stankovicin ym. (2013) määrällisessä tutkimuksessa. Julkaisustrategian kriittisyyttä ketterän ohjelmistoprojektin onnistumiselle on perusteltu muun muassa sillä, että ketterän ohjelmistokehityksen perusperiaatteisiin kuuluvat toistava ja lisäävä kehitystyö, jotta tuotteen kehittäminen etenee asiakkaan palautteiden ja jatkuvan kehittämisen ideologian mukaan. Jotta tuotetta voidaan kehittää toistavasti ja lisäävästi, asiakkaan täytyy vastaanottaa tuotetta jatkuvasti, eli ketterän ohjelmistokehityksen julkaisustrategian mukaisesti. Julkaisustrategia on monissa alan tutkimuksissa mainittu olevan ketterän ohjelmistokehityksen kulmakivi, ja samoin termein myös tämän tutkielman haastateltavat kuvasivat kyseisen menestystekijän merkitystä ketterälle ohjelmistokehitysprojektille.

Organisaatiokulttuuria pidettiin kriittisenä menestystekijänä ainoastaan Aldahmashin ym. (2017) sekä Misran ym. (2009) tutkimuksissa. Organisaatiokulttuuria tutkiessa ketterän ohjelmistoprojektin kriittisenä menestystekijänä on tärkeä muistaa, että organisaatiokulttuurin vaikutuksen mittaaminen on hyvin vaikeaa, kuten moni haastateltavakin nosti esille. Kuitenkin organisaatiokulttuurin vaikutusta pidetään suhteellisesti yhtä mittavina vertaillessa aikaisempien tutkimusten löydöksiä ja tämän tutkielman haastattelujen tuloksia. Tässä tutkielmassa tarkastelluista tutkimuksista organisaatiokulttuuria pidetään kriittisenä menestystekijänä kahdessa tutkimuksessa viidestä tutkimuksesta ja haastatteluissa kolme haastateltavaa kahdeksasta haastateltavasta pitävät organisaatiokulttuuria kriittisenä menestystekijänä.

7.2 Tulosten merkitys teorialle

Tämän tutkielman löydökset osaltaan vahvistavat ja osaltaan tuovat uutta näkemystä tutkimuskenttään. Osaltaan ristiriitaiset tulokset aikaisempiin alan tutkimuksiin oli odotettavissa, koska ketterien ohjelmistokehitysmenetelmien soveltaminen on hyvin moninaista projektien monipuolisuuden ja erilaisten liiketoimintaympäristöjen takia. Tämän

tutkimuksen tutkimusasetelma koostui myös pääosin Chowin & Caon (2008) ja Misran ym. (2009) tutkimusasetelmien yhdistelmänä. Tämän lisäksi ohjelmointialan kehittyessä ja muuttuessa oletettavasti nopeasti, uusien tutkimusten merkitys on hyvin suuri huolimatta löydösten mahdollisesta ristiriitaisuudesta toistensa kanssa.

Johtopäätöksenä on myös huomionarvoista edelleen mainita, että muun muassa tässä tutkielmassa esitellyt ketterät ohjelmistokehitysmenetelmät ja niiden menestystekijät sekä menestystekijöiden tärkeydet ovat hyvin paljolti riippuvaisia projektin tavoitteista, vaatimuksista ja liiketoimintaympäristöstä, joten niitä tulisi soveltaa ja hyväksikäyttää tilanteen vaatimilla tavoilla, eikä niinkään niin sanotusti oppikirjamaisesti, kuten moni haastateltava tämän tutkielman teemahaastatteluissa nosti esille.

Edellisestä johdettuna tärkeä johtopäätös on myös se, että ketterän ohjelmistokehityksen menestystekijöitä tarkastellessa tulisi keskittyä tarkastelemaan yksittäisten menestystekijöiden sijaan kaikkia projektiin liittyviä menestystekijöitä, jotka vaikuttavat suuresti toisiinsa ja tätä kautta ketterän ohjelmistoprojektin onnistumiseen. Toisin sanoen, ketterän ohjelmistokehityksen menestystekijöihin tulisi suhtautua kokonaisuutena, eikä tarkastella menestystekijöitä vain yksittäin. Esimerkiksi ketterän ohjelmistokehityksen mukaisesta toistavasta ja lisäävästä julkaisustrategiasta menestystekijänä on huomattavasti enemmän hyötyä, kun kommunikaatio asiakkaan ja projektitiimin välillä on suoraa ja avointa, ja täten asiakkaan tarpeita voidaan päivittää projektin edetessä palautteiden muodossa. Toisena esimerkkinä voidaan mainita esimerkiksi projektitiimin kompetenssiosaamisen merkitys puhuttaessa itseohjautuvasta ja matalahierarkkisesta päätöksenteosta.

7.3 Tulosten merkitys käytännölle

Käytännön näkökulmasta mietittynä tämä tutkielma antaa ketteriä ohjelmistokehitysmenetelmiä soveltaville organisaatioille merkitystä erityisesti siitä näkökulmasta, että todennäköisimmin digitalisaation edetessä ja myös tätä kautta kilpailun alati tiukentuessa asiakkaalle arvoa tuottavien asioiden merkitys korostuu entisestään, joka luonnollisesti vaatii toimittavalta taholta ymmärrystä ja ketteryyttä vastata asiakkaan monimutkaisiin

ja alati muuttuviin tarpeisiin entistä paremmin. Tässä tutkielmassa esitellyt kuusi ketterän ohjelmistokehityksen menestystekijää sekä ketterän ohjelmistokehityksen malli antavat organisaatioille mahdollisuuden peilata omaa toimintaansa ja tätä kautta auttavat kehittämään omaa toimintaansa ketterämpään suuntaan. Kuitenkin tässä yhteydessä on syytä mainita, että ketteryys ja ketterät toimintatavat ovat työkaluja ja menetelmiä, eikä ketteryyden tulisi niinkään itsessään olla organisaation tavoite.

Tutkielmassa havaittujen tulosten yhteenvetona ja johtopäätöksenä erityisesti käytännöllisestä näkökulmasta voidaan todeta, että toimiakseen mahdollisimman ketterästi, projektitiimien tulee varmistaa ensi kädessä asiakasyhteistyön sujuvuus ja keskittyä vain niihin asioihin, jotka tuovat asiakkaalle eniten arvoa, jota kutsutaan myös turhan poistoksi (engl. leanness). Jotta projektissa voitaisiin keskittyä asiakkaalle arvoa tuottaviin toimiin, vaaditaan perusteellista vaatimusten selvittämistä projektin alkuvaiheessa sekä myös niiden päivittämistä asiakkaan vaatimusten muuttuessa ja tavoitteiden kirkastuessa kehitystyön aikana. Suurena apuna tässä on ketterän ohjelmistokehityksen toistava (engl. iterative) ja lisäävä (engl. incremental) julkaisustrategia, joka tarkoittaa projektin tuotoksen versioiden toimittamista asiakkaalle tasaisin ja säännöllisin väliajoin, jotta asiakas voi antaa palautetta kehitystyön suunnasta, mahdollisesta vaatimusten muuttumisesta tai tarpeiden täyttymisestä. Jotta palautteen antaminen ja vastaanottaminen olisi hedelmällistä, projektitiimin ja asiakkaan välillä tulee vallita läpinäkyvä, suora ja avoin keskusteluyhteys eli kommunikaation tulee olla mahdollisimman laadukasta. Versioita tulee toimittaa sekä palautetta vastaanottaa ja käsitellä nopeissa ja lyhyissä sykleissä, jotta kehitystyössä minimoidaan turhan työn tekeminen ja projekti etenee nopeasti. Jotta projektitiimillä olisi mahdollista vastata edellä mainittuihin tehtäviin ja vaatimuksiin, projektitiimiltä vaaditaan syvää osaamista. Projektitiimistä tulisi siis löytyä esimerkiksi huipputason ohjelmoijia ja testaajia. Syvän osaamisen lisäksi täytyy ottaa huomioon myös laajan osaamisen merkitys, jotta vältytään niin sanotuilta pullonkauloilta kehitystyössä. Syvää ja laajaa osaamista kuitenkin jopa tärkeämpi tekijä on projektitiimin kyky ja mahdollisuus edetä kehitystyössä kokeilemisen ja oppimisen kautta, jotta asiakkaalle eniten arvoa tuottavat asiat löydetään. Projektitiimin sekä koko organisaation tulisi vaalia ja kasvattaa yrityskulttuuria, joka kannustaa työn tekemiseen kokeilunhalun ja

oppimishalukkuuden (engl. growth mindset) kautta, itseohjautuvaan työskentelyyn sekä matalahierarkkiseen päätöksentekoon.

Kyseisten huomioiden lisäksi luvussa 6 esitellään ketterän ohjelmistokehityksen malli ja sellaisia tekijöitä, jotka ketteryyteen pyrkivien organisaatioiden olisi hyvä ottaa huomioon omaksuakseen sitä kautta ketteryyttä ja saavuttaakseen onnistuneita ohjelmistoprojekteja. Tässä kohtaa on myös huomionarvoista nostaa esille, että ketterän ohjelmistokehityksen, sen menestystekijöiden tutkimus sekä tässä tutkielmassa että koko tutkimuskentässä esiin nousseet työskentelymenetelmät ja toimintatavat ovat mahdollisesti myös sovellettavissa jossain määrin myös muille toimialoille saavuttaakseen onnistumisia projekteissa sekä työelämässä yleensä.

Voidaan olettaa, että ketterän ohjelmistokehityksen kriittiset menestystekijät ovat mielenkiintoinen tutkimusaihe myös tulevaisuudessa, koska edelleen kiihtyvän digitalisaation asettamien vaatimusten myötä ohjelmistoyritysten on entistäkin tärkeämpi ymmärtää, mitkä tekijät mahdollistavat menestyksen saavuttamisen ketterissä ohjelmistoprojekteissa.

Lähteet

Abbas, N., Gravell, A. & Wills, G. (2008). Historical Roots of Agile Methods: Where did "Agile Thinking" Come from? [online]. Teoksessa P. Abrahamsson, R. Baskerville, K.Conboy, B.Fitzgerald & X.Wang. Proceedings of the 9th International Conference on Agile Processes in Software Engineering and Extreme Programming, 94-103. Berliini: Springer. [12.7.2020]. Saatavissa: https://www.researchgate.net/publication/39996418_Historical_Roots_of_Agile_Methods_Where_Did_Agile_Thinking_Come_From

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2002). Agile Software Development Methods: Review and analysis [online]. Espoo, Finland: VTT Publication 478. [30.7.2020]. Saatavissa: https://www.vtt.fi/inf/pdf/publications/2002/P478.pdf

Aldahmash, A. & M., Gravell, A. & Howard, Y. (2017). A Review on the Critical Success Factors of Agile Software Development [online]. Teoksessa Systems, Software and Services Process Improvement: 24th European Conference, 504-512. Eds. Stolfa, J., Štolfa, S., O'Connor, R., & Messnarz, R. [23.7.2020]. Ostrava, Czech Republic: EuroSPI 2017. Saatavissa: https://www.researchgate.net/publication/319062098_A_Review_on_the_Critical_Success_Factors_of_Agile_Software_Development

Anderson, D.J. (2010). Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press, Sequim, WA. ISBN 978-0984521401.

Anurina, O. (2019). Agile SDLC: Skyrocketing Your Project with Agile Principles [online]. MLSDev Blog. [31.8.2020] Saatavissa: https://mlsdev.com/blog/agile-sdlc

Beck, K. (1999a). Extreme Programming Explained: Embrace Change. Boston: Addison-Wesley Professional. ISBN 0-32-127865-8.

Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001). Manifesto for Agile Software

Development [online]. Agilemanifesto.org. [1.6.2020]. Saatavissa: http://www.agilemanifesto.org/

Beldarrain, Y. (2019). Developing an Agile Global Workforce. In The Wiley Handbook of Global Workplace Learning [online]. [10.7.2020]. Saatavissa: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119227793.ch3

Boehm, B. & Turner, R. (2005). Management challenges to implementing agile processes in traditional development organizations. IEEE Software 22(5), 30–39.

Bullen, C.V. & Rockart, J.F. (1981). A Primer on critical success factors, Information Systems Research no. 69 [online]. [23.7.2020]. Saatavissa: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.465.3321&rep=rep1&type=pdf

Cao, L., Mohan, K., Xu, P. & Ramesh, B. (2009). A framework for adapting agile development methodologies [online]. European Journal of Information Systems 18, 332–343. [10.6.2020]. Saatavissa: https://www.researchgate.net/publication/31957499_A_framework_for_adapting_agile_development_methodologies

Chow, T. & Cao, D. B. (2008). A survey study of critical success factors in agile software projects [online]. Journal of Systems and Software, 81(6), 961–971. [15.5.2020]. Saatavissa: https://www.sciencedirect.com/science/article/abs/pii/S0164121207002208

Cobb, C.G. (2011). Becoming More Agile. In Making Sense of Agile Project Management, C.G. Cobb (Ed.) [online]. [20.6.2020]. Saatavissa: https://onlinelibrary.wiley.com/doi/10.1002/9781118085950.ch3

Cockburn, A. & Highsmith, J. (2001). Agile software development, the people factor [online]. Computer 34(11), 131–133. [14.7.2020]. Saatavissa: https://www.researchgate.net/publication/2955526_Agile_software_development_The_people_factor

Cockburn, A. & Williams, L. (2003). Agile Software Development: It's about Feedback and Change [online]. IEEE Computer. [26.6.2020]. Saatavissa: https://ieeexplore.ieee.org/document/1204373

Cohen, D., Lindvall, M. & Costa, P. (2004). An introduction to Agile Methods [online]. In: Advances in Computers, 61-66. Eds. Zelkowitz, M.V. Amsterdam, Netherlands: Elsevier. [22.6.2020]. Saatavissa: http://www.cse.chalmers.se/~feldt/courses/agile/cohen_2004_intro_to_agile_methods.pdf

Conboy, K. (2009). Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development [online]. Information Systems Research. [15.6.2020]. Saatavissa: https://www.researchgate.net/publication/47442715_Agility_From_First_Principles_Reconstructing_the_Concept_of_Agility_in_Information_Systems_Development

Dingsøyr, T., Nerur, S., Balijepally, V. & Moe, N. B. (2012). A Decade of Agile Methodologies Towards Explaining Agile Software Development [online]. Journal of Systems and Software. [9.6.2020]. Saatavissa: https://www.researchgate.net/publication/236211358_A_decade_of_agile_methodologies_Towards_explaining_agile_software_development

Dong, C., Chuah, K. B., & Zhai, L. (2004). A study of critical success factors of information system projects in China [online]. Paper presented at PMI® Research Conference: Innovations, London, England. Newtown Square, PA: Project Management Institute. [21.8.2020]. Saatavissa: https://www.pmi.org/learning/library/study-critical-success-factors-information-system-projects-8291

Dvir, D., Lipovetsky, S., Shenhar, A. & Tishler, A. (1998). In search of project classification: a nonuniversal approach to project success factors [online]. Research Policy 27(9), 915–935. [4.7.2020]. Saatavissa: https://www.sciencedirect.com/science/article/abs/pii/S0048733398000857

Dvir, Dov & Lipovetsky, Stan & Shenhar, Aaron & Tishler, Asher. (2003). What is really important for project success? [online]. A refined, multivariate, comprehensive

analysis. International Journal of Management and Decision Making - Int J Manag Decis Making. 4. [21.8.2020]. Saatavissa: 10.1504/IJMDM.2003.004001.

Erickson, J., Lyytinen, K., & Siau, K. (2005). Agile Modeling, Agile Software Development, and Extreme Programming: The State of Research [online]. Journal of Database Management. [5.6.2020]. Saatavissa: https://www.researchgate.net/publication/220373708_Agile_Modeling_Agile_Software_Development_and_Extreme_Programming_The_State_of_Research

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. Proceedings of the 2010 [online]. ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. 37–41. Bolzano: ACM. [17.7.2020]. Saatavissa: https://www.researchgate.net/publication/221494925_An_empirical_study_on_the_relationship_between_the_use_of_agile_practices_and_the_success_of_Scrum_projects

Gandomani, T., Zulzalil, H., Ghani, A.A., Sultan, A.B. & Nafchi, M.Z. (2013). Obstacles in moving to agile software development methods; at a glance [online]. J. Comput. Sci. 9(5), 620. [12.7.2020]. Saatavissa: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.679.7296&rep=rep1&type=pdf

Greer, D. & Hamon, Y. (2011). Agile Software Development [online]. Software Practice and Experience, 41: 943-944. [11.7.2020]. Saatavissa: https://onlinelibrary.wiley.com/doi/full/10.1002/spe.1100

Guntur, M., Purwandari, B., Raharjo, T., Solichah, I., and Kumaralalita, L. 2018. Critical success factors for information systems development: A case study in e-government [online]. In ACM International Conference Proceeding Series, 29--33. [21.8.2020]. Saatavissa: https://doi.org/10.1145/3278252.3278288

Hart, M.A. (2011). Agile Product Management with Scrum: Creating Products that Customers Love by Roman Pichler [online]. Journal of Product Innovation Management, 28: 615-615. [3.7.2020]. Saatavissa:

https://play.google.com/store/books/details/Roman_Pichler_Agile_Product_Management_with_Scrum?id=aLSu0P0EojEC

Kotimaisten kielten keskus. (2018). Kielitoimiston sanakirja [online]. Helsinki: Kotimaisten kielten keskus. [2.6.2020]. Saatavissa: https://www.kielitoimistonsanakirja.fi/

Kinnunen, H. (2015). Ohjelmistokehityksen ketteryys ja sen mittaaminen [online]. Jyväskylän yliopisto: Tietojärjestelmätieteen pro gradu -tutkielma. [13.6.2020]. Saatavissa: https://jyx.jyu.fi/bitstream/handle/123456789/48063/URN%3aNBN%3afi%3ajyu-201512113991.pdf?sequence=1&isAllowed=y

Kniberg, H. & Skarin, M. (2010). Kanban and Scrum – Making the Most of Both [online]. Lulu.com. [25.7.2020] Saatavissa: http://www.agileinnovation.eu/wordpress/wp-content/uploads/2010/09/KanbanAndScrum_MakingTheMostOfBoth.pdf

Larman, C., & Basili, V.R. (2003). Iterative and Incremental Developments. A Brief History [online]. IEEE Computer. [28.6.2020]. Saatavissa: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1204375

Livermore, J.A. (2007). Factors that impact implementing an agile software development methodology [online]. Teoksessa Proceedings of 2007 IEEE SoutheastCon, pp. 82–86. [13.7.2020]. Saatavissa: https://www.researchgate.net/publication/42804520_Factors_that_Significantly_Impact_the_Implementation_of_an_Agile_Software_Development_Methodology

Lynch, T. & Gregor, S. (2004). User participation in decision support systems development: influencing system outcomes [online]. Euro. J. Information Systems 13, 286–301. [12.6.2020]. Saatavissa: https://www.researchgate.net/publication/220393171_User_participation_in_decision_support_systems_development_Influencing_system_outcomes

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

[12.6.2020]. Saatavissa: https://orsociety.tandfonline.com/doi/abs/10.1057/palgrave.ejis.3000604#.XyqW5ygzaUk

McLeod, L. & MacDonell, S.G. (2011). Factors that affect software systems development project outcomes [online]. ACM Computing Surveys 43 (4), 1–56. [19.7.2020]. Saatavissa: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.214.9834&rep=rep1&type=pdf

Miles, M. B., Huberman, A. M., & Saldaña, J. (2014). Qualitative data analysis: a methods sourcebook [online]. Thousand Oaks, Kalifornia: SAGE Publications Inc. [28.6.2020]. Saatavissa: https://www.researchgate.net/publication/272566756_Qualitative_Data_Analysis_A_Methods_Sourcebook

Misra, S. C., Kumar, V. & Kumar, U. (2009). Identifying some Important Success Factors in Adopting Agile Software Development Practices [online]. Journal of Systems and Software, 82(11), 1869–1890. [1.7.2020]. Saatavissa: https://www.sciencedirect.com/science/article/pii/S016412120900123X

Moe, N. B., Dingsøyr, T., & Dybå, T. (2010). A Teamwork model for Understanding an Agile Team: A Case Study of a Scrum Project [online]. Teoksessa Information and Software Technology, 480-491. Eds. Information and Software Technology. Amsterdam, Netherlands: Elsevier. [12.7.2020]. Saatavissa: https://www.researchgate.net/publication/220610393_A_teamwork_model_for_understanding_an_agile_team_A_case_study_of_a_Scrum_project

Myers, M. D. & Newman, M. (2007). The qualitative interview in IS research: Examining the craft [online]. Information and Organization, 17(1), 2–26. [27.7.2020]. Saatavissa: https://www.sciencedirect.com/science/article/abs/pii/S1471772706000352

Petersen, K., & Wohlin, C. (2011). Measuring the flow in Lean software development [online]. Software: Practice and experience, 41(9), 975–996. [25.6.2020]. Saatavissa: https://www.researchgate.net/publication/228451017_Measuring_the_flow_in_Lean_software_development

Peterson, D. (2015). What is Kanban? [online]. Kanban Blog. [22.6.2020]. Saatavissa: https://kanbanblog.com/

Poppendieck, M. & Cusumano, M. A. (2012). Lean Software Development: A Tutorial [online]. IEEE Software. [28.6.2020]. Saatavissa: https://ieeexplore.ieee.org/document/6226341

Poppendieck, M. & Poppendieck, T. (2003). Lean Software Development: An Agile Toolkit [online]. Boston: Addison-Wesley. [28.6.2020]. Saatavissa: http://ptgmedia.pearsoncmg.com/images/9780321150783/samplepages/0321150783.pdf

Poth, A., Sasabe, S., Mas, A. & Mesquida, A. (2019). Lean and agile software process improvement in traditional and agile environments [online]. Journal of Software: Evolution and Process. 2019. 31. e1986. [27.7.2020]. Saatavissa: https://www.researchgate.net/publication/327018101_Lean_and_agile_software_process_improvement_in_traditional_and_agile_environments

Pyörälä, Eeva. (2002). Mitä ja millaista on laadullinen tutkimus? Osa kokonaisuudessa Kvalitatiivisen tutkimuksen perusteista [online]. Johdantoa luentosarjaan. Laadulliset tutkimusmenetelmät yhteiskuntatieteissä -luentosarja. Helsingin yliopisto. [2.6.2020]. Saatavissa: http://www.valt.helsinki.fi/yleope/kvali/kvali1.htm

Radujković, M. & Sjekavica, M. (2017). Project Management Success Factor. Procedia Engineering. 196. 607-615. [21.8.2020]. Saatavissa: https://www.sciencedirect.com/science/article/pii/S1877705817331740

Rolstadås, A., Tommelein, I., Morten Schiefloe, P. and Ballard, G. (2014). Understanding project success through analysis of project management approach [online]. International Journal of Managing Projects in Business, Vol. 7 No. 4, pp. 638-660. [21.8.2020]. Saatavissa: https://doi.org/10.1108/IJMPB-09-2013-0048

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering [online]. Empirical Software Engineering, 14(2), 131–164. [29.6.2020]. Saatavissa: https://link.springer.com/article/10.1007/s10664-008-9102-8

Saaranen-Kauppinen & Puusniekka. (2006). KvaliMOTV - Menetelmäopetuksen tietovaranto [online]. Tampere: Yhteiskuntatieteellinen tietoarkisto. [12.6.2020]. Saatavissa: https://www.fsd.tuni.fi/menetelmaopetus/

Sanchez, O., Terlizzi, M., & de Moraes, H. (2016) Factors of Successful Management of Information Systems Development Projects [online]. International Research Workshop on IT Project Management 2016. 14. [21.8.2020]. Saatavissa: https://aisel.aisnet.org/irwitpm2016/14

Schwaber, K. & Sutherland, J. (2017). The Scrum Guide [online]. ScrumGuides.org. [27.6.2020] Saatavissa: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

Sheffield, J. & Lemétayer, J. (2013). Factors associated with the software development agility of successful projects. International Journal of Project Management. 31(3), 459–472. [17.7.2020]. Saatavissa: https://www.sciencedirect.com/science/article/abs/pii/S0263786312001251

Shenhar, A.J., Dvir, D., Levy, O. & Maltz, A.O. (2001). Project success: a multidimensional strategic concept [online]. Long Range Plan. 34(6), 699–725. [27.7.2020]. Saatavissa: https://www.semanticscholar.org/paper/Project-Success%3A-A-Multidimensional-Strategic-Shenhar-Dvir/524dbb2c6592fd7889e96f0745b6ec5ce9c5adc8

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 [online]. Journal of Systems and Software. 86(6), 1663–1678. [9.5.2020]. Saatavissa: https://www.researchgate.net/publication/256991972_A_survey_study_of_critical_success_factors_in_agile_software_projects_in_former_Yugoslavia_IT_companies

Sutherland, J. (2010). Scrum Handbook [online]. Sommerville: Scrum Training Institute Press. [23.5.2020]. Saatavissa: https://www.researchgate.net/publication/301685699_Jeff_Sutherland's_Scrum_Handbook/download

Tam, C., Moura, E. J. D. C., Oliveira, T., & Varajão, J. (2020). The factors influencing the success of on-going agile software development projects [online]. International Journal of Project Management, 38(3), 165-176. [21.8.2020]. Saatavissa: https://doi.org/10.1016/j.ijproman.2020.02.001

Tuomi, J. & Sarajärvi, A. (2018). Laadullinen tutkimus ja sisällönanalyysi. Helsinki: Kustannusosakeyhtiö Tammi. ISBN 9789513199531.

Walker, R. (2004). Getting and analysing of qualitative data [online]. The PREST Training Resources. Commonwealth of Learning. [27.6.2020]. Saatavissa: http://oasis.col.org/bitstream/handle/11599/87/A4.pdf?sequence=1&isAllowed=y

Wan, J. & Wang, R. (2010). Empirical Research on Critical Success Factors of Agile Software Orocess Improvement [online]. Journal of Software Engineering and Applications 3(12):1131-1140. [25.5.2020]. Saatavissa: https://www.researchgate.net/publication/220204390_Empirical_Research_on_Critical_Success_Factors_of_Agile_Software_Process_Improvement/download

Wang, X., Conboy, K., & Cawley, O. (2012). “Leagile” software development: An experience report analysis of the application of lean approaches in agile software development [online]. Journal of Systems and Software, 85(6), 1287–1299. [25.5.2020]. Saatavissa: https://www.sciencedirect.com/science/article/abs/pii/S0164121212000404