Tampereen yliopisto

Erik Keitaanniemi

KETTERIEN OHJELMISTOKEHITYSMENETELMIEN SOVELTUVUUS STARTUP-YRITYKSIIN

Tekniikan ja luonnontieteiden tiedekunta
Kandidaatintyö
Maaliskuu 2019

TIIVISTELMÄ

ERIK KEITAANNIEMI: Ketterien ohjelmistokehitysmenetelmien soveltuvuus startup-yrityksiin
Tampereen yliopisto
Kandidaatintyö, 30 sivua
Maaliskuu 2019
Teknis-taloudellinen kandidaatin tutkinto-ohjelma
Pääaine: Tietojohtaminen
Tarkastaja: TkT Pasi Hellsten

Avainsanat: ketterät menetelmät, startup-yritys, Scrum, Extreme Programming, DSDM, FDD

Kilpailun jatkuvan kiristymisen vuoksi myös ohjelmistoalalla on jouduttu kehittämään tehokkaampia työskentelytapoja. Ohjelmistoalalla tämä on tarkoittanut uusien menetelmien ja toimintatapojen kehittämistä. Ketterät ohjelmistokehitysmenetelmät ovat yksi tapa, jolla ohjelmistoalan yritykset ovat pyrkineet tehostamaan toimintaansa. Tässä kandidaatintyössä käsiteltiin ketteriä menetelmiä yleisellä tasolla, sekä neljää eri menetelmää tarkemmin. Työssä keskityttiin erityistesti ohjelmistoalan 2-5 henkilön startup-yrityksiin.

Kandidaatintyö toteutettiin kirjallisuuskatsauksena. Suuri enemmistö käytetyistä lähteistä on tieteellisiä artikkeleita, mutta myös kirjalähteitä on käytetty jonkin verran. Lähteistä suurin osa on 2000-luvulta. Kirjallisuudesta löytyy melko vähän suoraan aiheeseen liittyvää tutkimusta, mutta työssä pyrittiin yhdistelemään eri tutkimuksissa löydettyjä tuloksia ja näin löytämään tietoa ketterien menetelmien sopivuudesta 2-5 henkilön startup-yrityksiin. Kunkin käsitellyn ketterän menetelmän soveltuvuuden kohdalla keskityttiin lähinnä sen tarjoamaan prosessimalliin, toimintatapoihin ja rooleihin. Näitä kolmea ketterän menetelmän ominaisuutta tutkimalla pyrittiin löytämään sekä seikkoja, jotka tukevat ketterien menetelmien soveltuvuutta, että seikkoja, jotka saattavat aiheuttaa haasteita.

Työn keskeisinä tuloksina löydettiin, että käsitellyissä ketterissä menetelmissä on paljon samankaltaisuuksia, mutta myös melko paljon eroavaisuuksia. Osa käsitellyistä menetelmistä soveltuu startup-yrityksiin melko hyvin, mutta osan löydettiin esimerkiksi prosessinsa tai rooliensa kautta tuottavan suoraan sovellettuna ongelmia. Ketterien menetelmien räätälöinnillä todettiin olevan suuri vaikutus niiden soveltamisessa startup-yrityksiin. Prosessien sopivuudessa suurimmat löydetyt erot liittyivät niiden joustavuuteen. Roolien osalta suurimmat erot olivat roolien määrässä ja siinä, kuinka joustavia eri menetelmien roolit ovat. Työssä saatiin selville hyvin tietoa yksittäisten menetelmien sopivuudesta startup-yrityksiin ja lisäksi yhdenkään käsitellyn ketterän menetelmän ei voida nähdä sopivan erityisen hyvin startup-yrityksiin varsinkaan tarkasti seurattuna.

ABSTRACT

ERIK KEITAANNIEMI: Suitability of agile software development methods in startup companies
Tampere University
Bachelor of Science Thesis, 30 pages
March 2019
Major: Information and Knowledge Management
Examiner: Pasi Hellsten

Keywords: agile software development, startup company, Scrum, Extreme Programming, DSDM, FDD

The constant tightening of competition in the software industry has led to businesses developing more effective ways of working. In the software industry this has meant the development of new methods and practices. Agile software development methods are one way software companies have tried to improve their efficiency. In this Bachelor of Science Thesis agile software development methods were reviewed in general, as well as four methods were reviewed more specifically. Only startup companies that are in the area of software development and have from two to five employees were discussed.

This thesis was performed as a literature review. Majority of the references reviewed are scientific articles, but also some books were also used. Most of the references were written after 2000. There is only little literature available of the suitability of agile software development methods in startup companies, but findings of previous studies were put together to find out more about using agile software development methods in software startup companies of two to five employees. On each agile software development method, three things were discussed, process, practices and roles. By combining these three features of different agile methods I tried to find facts that support the use of agile methods in startup companies, as well as facts that might pose challenges.

Key results of this Bachelor of Science Thesis were that reviewed agile software development methods had many similarities, but also quite a lot of differences. Some of the methods reviewed are suitable to be used in startup companies, but some were found to pose challenges through their process or roles if followed rigorously. Tailoring of agile software development methods was found to have a big effect on their use in startup companies. Biggest differences in processes of the different methods reviewed were found to be in their flexibility. Biggest differences in roles of the different methods reviewed were in their number and how flexible they are. Many findings about the suitability of agile software development methods were done, but none of the reviewed methods were found to be particularly suitable in startup companies, at least not if followed rigorously.

ALKUSANAT

Tämä kandidaatintyö on tehty Tampereen yliopiston tietojohtamisen koulutusohjelmaan keväällä 2019. Idean kandidaatintyön aiheesta sain tehdessäni esitutkimusta mahdollisista aiheista, jolloin huomasin, että ketterien menetelmien soveltamisesta startup-yrityksiin on olemassa melko vähän tutkimusta. Aihe yhdistää kaksi mielenkiinnon kohdettani, ohjelmistotekniikan ja startup-yritykset.

Haluan kiittää työni ohjaajaa ja tarkastajaa Pasi Hellsteniä ohjauksesta ja palautteesta työn aikana. Kiitos myös pienryhmälleni hyvistä ideoista ja palautteesta kurssin aikana.

Tampereella, 31.3.2019

Erik Keitaanniemi

SISÄLLYSLUETTELO

KESKEISET KÄSITTEET

Dynamic Systems Development method (DSDM) on ketterä ohjelmistokehitysmenetelmä, jonka mukaan ”paras kaupallinen arvo syntyy, kun projekteilla on selvät kaupalliset tavoitteet, nopea toimitustahti ja kun motivoidut ihmiset työskentelevät yhdessä” (Stapleton 1997 s. 11).

Extreme Programming on ketterä ohjelmistokehitysmenetelmä, joka perustuu asiakastyytyväisyyteen ja iteraatioon. Se sisältää ydinarvoja, joiden pohjalta on kehitetty menetelmälle ominaisia aktiviteetteja (Wells 2013).

Feature-driven Development (FDD) on ketterä ohjelmistokehitysmenetelmä, jonka periaate on kehittää ohjelmistoja asiakkaan arvostamien ominaisuuksien pohjalta. Se on ketterien menetelmien perustan mukaisesti iteratiivinen kehitysmenetelmä. (Nawaz et al. 2017)

Iteratiivisuus tarkoittaa tietyn prosessin toistamista paremman ratkaisun saavuttamiseksi. (Oxfrod Dictionary)

Ketterät ohjelmistokehitysmenetelmät (agile software development methods) ovat joukko menetelmiä, joiden avulla pyritään tuottamaan ohjelmistoja nopeasti ja ottamaan huomioon jatkuvasti muuttuvat vaatimukset. Ketterille ohjelmistokehitysmenetelmille yhteistä on esimerkiksi jatkuva, iteratiivinen ohjelmistojen toimittaminen asiakkaalle sekä asiakkaan kanssa läheisessä yhteistyössä tapahtuva ohjelmistotuotanto ja vaatimusten keruu. (Greer et al. 2011)

Ohjelmistotuotanto on prosessi, jossa suunnitellaan, tuotetaan ja ylläpidetään sovellusta. (Oxford Dictionary).

Scrum on ketterä ohjelmistokehitysmenetelmä, joka perustuu ennalta määrättyihin rooleihin ja toimintatapoihin. Scrum on ketterien menetelmien perustan mukaisesti myös iteratiivinen kehitysmenetelmä (Schwaber & Beedle 2004, Vuorinen 2011 mukaan).

Startup-yritys on uusi yritys, joka pyrkii toimimaan uusilla markkinoilla pienillä resursseilla (Paternoster et al. 2014). Tehtävässä tutkimuksessa startup-yrityksellä tarkoitetaan nimenomaan ohjelmistoalan startup-yrityksiä.

1. JOHDANTO

Jatkuva kilpailun kiristyminen on aikojen saatossa johtanut yritysten jatkuvaan tarpeeseen tehostaa toimintaansa. Tehostamista on tapahtunut sekä perinteisen, valmistavan teollisuuden että myöhemmin myös muiden alojen toimintatavoissa. Ohjelmistoalalla tämä on tarkoittanut uusien toimintatapojen ja menetelmien kehittämistä, joista myös ketterät ohjelmistokehitysmenetelmät ovat yksi esimerkki.

Ketterät ohjelmistokehitysmenetelmät (agile software development methods) ovat joukko menetelmiä, joiden avulla pyritään tuottamaan ohjelmistoja nopeasti ja ottamaan huomioon jatkuvasti muuttuvat vaatimukset (Greer et al. 2011). Tämän kandidaatintyön aihe liittyy ketterien ohjelmistokehitysmenetelmien käyttöön startup-yrityksissä. Laajemmin ajateltuna tutkimuksen aiheen voidaan mielestäni nähdä liittyvän yritysten välisen kilpailun jatkuvan kiristymisen tuomaan jatkuvaan tehostustarpeeseen. Aivan kuten perinteisessä valmistavassa teollisuudessa, myös ohjelmistoalalla tuottavampien työskentelytapojen etsiminen on jatkuvaa. Ketterät ohjelmistokehitysmenetelmät ovat yksi tapa, joilla on pyritty vastaamaan juuri yritysten tarpeeseen luoda tehokkaampia työskentelytapoja. Aihe on valittu, koska se on mielenkiintoinen ja siitä löytyy myös melko vähän tutkimusta.

Startup-yritys -termiä käytettiin ensimmäistä kertaa 1990-luvun puolivälissä kuvaamaan uusia, innovatiivisia ja menestyksekkäitä yrityksiä. Ohjelmistoalan startup-yritykset ovat uusia yrityksiä, jotka pyrkivät kehittämään ohjelmistoja nopeasti kasvaville ja helposti skaalautuville markkinoille pienillä resursseilla. (Paternoster et al. 2014) Sutton (2000) kuvailee artikkelissaan ohjelmistoalan startup-yrityksiä neljällä ominaisuudella. Ensimmäinen on yrityksen toimintahistoria, joka näillä yrityksillä on lyhyt tai sitä ei ole ollenkaan. Toinen on rajatut resurssit, jotka johtavat siihen, että yritykset pyrkivät saamaan tuotteensa markkinoille nopeasti. Kolmas ohjelmistoalan startup-yrityksille yhteinen ominaisuus on useat sidosryhmät, jotka pyrkivät kaikki vaikuttamaan yrityksen toimintaan. Näitä sidosryhmiä ovat esimerkiksi sijoittajat, asiakkaat ja kilpailijat. Neljäs ominaisuus on ohjelmistoalan startup-yritysten käyttämät uudet teknologiat. Suttonin (2000) mukaan yritysten pitää toimia uusilla teknologioilla, jotta ne voisivat pärjätä uusilla markkinoilla. (Sutton 2000)

Aiheen sisältämistä elementeistä, eli ketteristä ohjelmistokehitysmenetelmistä ja startup-yrityksistä löytyy paljon tutkimusta, mutta näitä yhdistäviä tutkimuksia löytyy kuitenkin melko rajatusti. Ohjelmistoalan startup-yritysten nykyisistä toimintatavoista ja käytännöistä löytyy melko kattavasti tutkimuksia, joita voidaan käyttää vertaillessa nykyisten

toimintatapojen suhdetta ketteriin menetelmiin. Näiden tutkimusten avulla saadaan hyvä kuva siitä, miten toimintaa voitaisiin tehostaa.

Tutkimus on tärkeä tehdä, koska sen avulla voidaan mahdollisesti löytää tehokkaampia työskentelytapoja ohjelmistoalan startup-yrityksiin. Tutkimuksen avulla mahdollisesti löydettäviä lopputuloksia voidaan pyrkiä soveltamaan suoraan yritysmaailmaan. Tutkimuksen avulla voidaan myös pystyä löytämään mahdollisia jatkotutkimuskohteita, sillä aihetta on tutkittu melko vähän. Jatkotutkimukset voivat myös edesauttaa ohjelmistoalan startup-yritysten toimintaa jatkossa.

Tutkimusongelmaa on pyritty rajaamaan keskittymällä eri ketterien menetelmien ominaisuuksien löytämiseen ja niiden soveltamiseen startup-yrityksiin. Tutkimalla eri menetelmiä ja soveltamalla niitä startup-yritysten ominaisuuksiin voidaan pyrkiä löytämään suotuisia ominaisuuksia tai menetelmiä, jotka sopivat erityisesti käytettäväksi startup-yrityksissä. Tutkimusongelmaa on rajattu myös sen osalta, mitä ketteriä menetelmiä tutkitaan. Valitut menetelmät ovat Scrum, Extreme Programming, Dynamic Systems Development Method sekä Feature-driven Development. Ketterillä menetelmillä on paljon yhteisiä ominaisuksia ja erillisten menetelmien lisäksi tullaan tutkimaan ketterien ohjelmistokehitysmenetelmien sopivuutta startup-yrityksiin yleisellä tasolla. Aihe on myös rajattu käsittelemään vain 2-5 henkilön ohjelmistoalan startup-yrityksiä. Rajaus vain ohjelmistoalan yrityksiin on luonnollista, koska ne ovat juuri niitä yrityksiä, joissa erilaisia ohjelmistokehityksen menetelmiä käytetään. Alle viiden hengen yrityksiin päädyttiin, koska tutkimuksen tavoitteena on löytää sopivia menetelmiä juurikin pienille yrityksille. Päätutkimuskysymyksenä on ”Mitkä ketterän ohjelmistokehityksen ominaisuudet soveltuvat erityisesti startup-yrityksiin?”.

Työn toinen luku käsittelee tutkimusmenetelmää ja -aineistoa. Työn kolmannessa luvussa käsitellään ketteriä ohjelmistokehitysmenetelmiä yleisellä tasolla sekä käsitellään neljää ketterää menetelmää ja niiden ominaisuuksia tarkemmin. Neljännessä luvussa käsitellään ohjelmistoalan startup-yrityksiä ja niiden tämänhetkisiä toimintatapoja. Viides luku yhdistää kaksi ensimmäistä lukua ja siinä käsitellään ketterien menetelmien soveltuvuutta startup-yrityksiin. Viidennessä luvussa pyritään myös etsimään tietoa tällä hetkellä käytössä olevista ketteristä menetelmistä startup-yrityksissä ja etsimään mahdollisesti muita toimivia ketteriä menetelmiä tai näiden ominaisuuksia.

2. TUTKIMUSMENETELMÄT JA -AINEISTO

Työ toteutetaan kirjallisuuskatsauksena. Tiedonhakua suoritetaan ketteristä ohjelmistokehitysmenetelmistä yleisellä tasolla, valituista menetelmistä sekä startup-yritysten nykyisistä toimintatavoista ja yleisistä ominaisuuksista. Suurin osa työssä käytettävistä lähteistä ovat tieteellisiä, vertaisarvioituja artikkeleita ja kirjallisuuslähteitä. Lähteinä tullaan käyttämään myös luotettaviksi arvoitujen internet-sivujen sisältöä. Nämä sivut ovat esimerkiksi ketteriä menetelmiä kehittävien organisaatioiden internet-sivuja. Joitakin työssä käytettäviä kuvia, kuten prosessikaavioita, etsitään myös suoraan internetistä. Ketterät ohjelmistokehitysmenetelmät ovat syntyneet vasta 2000-luvun alkupuolella, joten kovinkaan vanhoja lähteitä työhön ei käytetä.

Työssä vertaillaan sekä yhdistellään eri lähteistä saatua tietoa. Lähteistä saatujen tietojen välissä on myös jonkin verran omaa pohdintaa, jolla pyritään sekä yhdistämään lähteitä että tuomaan esille niiden erilaisuuksia. Tärkeässä osassa työtä on luvut kolme ja neljä, joissa käsitellään ketterien menetelmien teoriaa sekä startup-yrityksiä. Näissä luvuissa pyritään tuomaan esille ketterien menetelmien ja startup-yritysten sisältämiä ominaisuuksia. Luvussa viisi käsitellään näiden teoriaosuuksien pohjalta löydettyjä ominaisuuksia, ja pyritään löytämään startup-yrityksiin sopivia ketteriä menetelmiä tai näiden osia.

Tutkimusaineistoa etsitään eri tietokantojen avulla. Käytettäviä tietokantoja ovat Andor, Scopus, Google Scholar ja IEEE Xplore. Tietokantojen lisäksi tullaan käyttämään Tampereen Yliopiston kirjastoa. Hakuja kirjaston materiaaleista tullaan suorittamaan Tutcat-hakukoneella. Yhdistämällä hakusanat ”startup” ja ”agile” löytyy Andorista 1207 osumaa, Scopuksesta 109 osumaa, IEEE Xploresta 33 osumaa ja Google Scholarista 2750 osumaa. Iso osa artikkeleista käsittelee alan ja tutkimuksen nykytilaa, eikä suoraan samasta aiheesta tehtyjä tutkimuksia löydy. Seuraavaksi on esitelty joitakin tärkeimpiä työssä käytettäviä lähteitä:

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2017). Agile software development methods: Review and analysis.

Artikkeli käsittelee useita eri ketteriä menetelmiä ja näiden tutkimuksen tilaa. Siinä on myös yleisemmin kuvattu ketterien menetelmien syntyä ja yleisiä ominaisuuksia.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R. and Kern, J., 2001, Manifesto for agile software development.

Vaikkakin tämä lähde on pituudeltaan melko lyhyt, se sisältää kuvauksen tai ”manifestin”, jota pidetään koko ketterän ohjelmistokehityksen alkulähteenä. Manifestista käy ilmi kaikki ketterän ohjelmistokehittämisen pääpiirteet ja keskeiset asiat.

Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T. and Abrahamsson, P., 2014, Software development in startup companies: A systematic mapping study. Information and Software Technology, 56(10), pp.1200-1218.

Artikkeli sisältää tutkimuksen, jossa käsitellään laajasti ohjelmistoalan startup-yritysten toimintaa ja käytössä olevia käytäntöjä. Artikkelista saa hyvän kuvauksen siitä, mikä on ohjelmistoalan startup-yritysten tämänhetkinen tilanne, ja tätä voidaan verrata ketterien ohjelmistokehitysmenetelmien tuomiin etuihin.

Misra, Subhas C., Kumar, Vinod and Kumar, Uma, 2009, Identifying some important success factors in adopting agile software development practices

Artikkeli sisältää tutkimuksen, jossa on tutkittu ketterien ohjelmistokehitysmenetelmien käyttöönottoa yrityksissä. Tutkimuksen avulla on pyritty selvittämään tärkeimpiä tekijöitä, jotka ovat vaikuttaneet yritysten menestykseen niiden ottaessa käyttöön ketterät ohjelmistokehitysmenetelmät. Lähteestä saatavaa tietoa voidaan vertailla startup-yritysten ominaisuuksiin, ja näin selvittää, ovatko ohjelmistoalan startup-yritykset yleisellä tasolla soveltuvia ketterien ohjelmistomenetelmien käyttöön.

3. KETTERÄT OHJELMISTOKEHITYSMENETELMÄT

Ohjelmistoalalla on jo pitkään keskusteltu siitä, miten ohjelmistokehitys voitaisiin organisoida niin, että se olisi nopeampaa, laadukkaampaa ja halvempaa (Dybå & Dingsøyr 2008). Ketterät ohjelmistokehitysmenetelmät (Agile software development methods) ovat joukko menetelmiä, joiden avulla pyritään tuottamaan ohjelmistoja nopeasti ja ottamaan huomioon jatkuvasti muuttuvat vaatimukset. Ketterille ohjelmistokehitysmenetelmille yhteistä ovat esimerkiksi jatkuva, iteratiivinen ohjelmistojen toimittaminen asiakkaalle sekä asiakkaan kanssa läheisessä yhteistyössä tapahtuva ohjelmistotuotanto ja vaatimusten keruu. (Greer et al. 2011) Ketterän ohjelmistokehityksen julistuksessa, Agile Manifestossa (Beck et al. 2001), kirjoittajat määrittelevät ketterän ohjelmistokehityksen periaatteet seuraavasti:

1. Tärkein tavoitteemme on tyydyttää asiakas toimittamalla tämän tarpeet täyttäviä versioita ohjelmistosta aikaisessa vaiheessa ja säännöllisesti.
2. Otamme vastaan muuttuvat vaatimukset myös kehityksen myöhäisessä vaiheessa. Ketterät menetelmät hyödyntävät muutosta asiakkaan kilpailukyvyn edistämiseksi.
3. Toimitamme versioita toimivasta ohjelmistosta säännöllisesti, parin viikon tai kuukauden välein, ja suosimme lyhyempää aikaväliä.
4. Liiketoiminnan edustajien ja ohjelmistokehittäjien tulee työskennellä yhdessä päivittäin koko projektin ajan.
5. Rakennamme projektit motivoituneiden yksilöiden ympärille. Annamme heille puitteet ja tuen, jonka he tarvitsevat ja luotamme siihen, että he saavat työn tehtyä.
6. Tehokkain ja toimivin tapa tiedon välittämiseksi kehitystiimille ja tiimin jäsenten kesken on kasvokkain käytävä keskustelu.
7. Toimiva ohjelmisto on edistymisen ensisijainen mittari.
8. Ketterät menetelmät kannustavat kestävään toimintatapaan. Hankkeen omistajien, kehittäjien ja ohjelmiston käyttäjien tulisi pystyä ylläpitämään työtahtinsa pitkään tulevaisuuteen.
9. Teknisen laadun ja ohjelmiston hyvän rakenteen jatkuva huomiointi edesauttaa ketteryyttä.
10. Yksinkertaisuus, eli tekemättä jätettävän työn maksimointi, on oleellista.
11. Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat syntyvät itseorganisoituvissa tiimeissä.
12. Tiimi tarkastelee säännöllisesti, kuinka parantaa tehokkuuttaan, ja mukauttaa toimintaansa sen mukaisesti.

Agile Manifestossa (Beck et al. 2001) määritellään myös neljä arvoa, joihin ketterän kehityksen tulisi keskittyä:

1. Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja
2. Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota
3. Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja
4. Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa

Nämä ketterän kehityksen periaatteet ovat synnyttäneet useita toimintatapoja, joiden tavoitteena on tarjota asiakkaalle enemmän arvoa. Niiden ydinajatuksena on, että itseorganisoituvat tiimit pystyvät toimimaan tehokkaasti, ja että asiakkaiden jatkuva osallistuminen kehitysprosessiin sekä varsinkin palautteen antaminen johtaa parempaan lopputulokseen. (Dingsøyr et al., 2012) Erickson et al. (2005) määrittelevät artikkelissaan ketteryyden ydinajatukseksi kaiken ylimääräisen karsimisen. Tämä johtaa artikkelin mukaan siihen, että kaikenlaisiin muutoksiin esimerkiksi ympäristössä, käyttäjävaatimuksissa ja aikarajoissa pystytään vastaamaan nopeammin kuin perinteillä ohjelmistokehitysmenetelmillä. Misra (2012) tuo yhtenä tärkeimpänä pointtina esille asiakkaan korostamisen ketterissä menetelmissä. Asiakas on usein kiinnostunut juurikin lopputuloksena olevasta toimivasta tuotteesta, eivätkä ketterät menetelmät tämän takia anna juurikaan arvoa sivutuotteille, kuten suunnitteludokumenteille. Ketterät menetelmät keskittyvät Misran (2012) mukaansa nopeaan ja osittaiseen ohjelmiston toimittamiseen asiakkaalle.

Ketterän kehityksen pohjana voidaan pitää Agile Manifestoa (Beck et al. 2001), mutta sen historia ulottuu vuoteen 1991, jolloin Lean-metodologia sai jalansijaa valmistavassa teollisuudessa. Ketterän kehityksen kaltaisia menetelmiä onkin ollut käytössä jo ennen niiden varsinaista esittelyä vuonna 2001 (Poppendieck & Poppendieck 2013). Ketterän kehityksen liikkeellä ollut suuri vaikutus ohjelmistokehitykseen maailmanlaajuisesti (Dybå & Dingsøyr 2008). Erickson et al. (2005) huomauttavat artikkelissaan, että kehitysmenetelmän valinta on tullut yhä tärkeämmäksi, koska perinteiset kehitysmenetelmät ovat erittäin monimutkaisia ja vaikeakäyttöisiä. Ketterät menetelmät kehittyivät vastaamaan nykypäivän dynaamisten organisaatioiden haasteisiin (Misra 2012). Ohjelmistoalan kilpailun kiristyessä jatkuvasti yhä useammat yritykset pyrkivät tehostamaan ohjelmistokehitystään ja tässä vaiheessa ketterät menetelmät ovat usein isossa roolissa.

Vaikkakin ketterät menetelmät ovat vallanneet alaa nopeasti ja olleet erittäin pidettyjä ammattilaisten keskuudessa, ne ovat kohdanneet myös kritiikkiä. Ketterien menetelmien hyväksyminen varsinkin suurien yritysten tai valtiollisten toimijoiden projekteihin on ollut hidasta (Misra 2012). Joitakin tunnistettuja ongelmia ovat esimerkiksi se, että ketteriä menetelmiä käyttävissä projekteissa on usein vain vähän mahdollisuuksia käyttää alihankintaa ja se, että ketterät menetelmät eivät usein tue kovinkaan suuria tiimejä. (Turk & France 2002) Ketterissä menetelmissä on kuitenkin paljon eroja, sillä jotkut keskittyvät enemmän projektinhallinnan tai ohjelmistokehityksen aspekteihin, kun taas jotkut ovat kokonaisvaltaisempia (Misra & Misra 2011). Tästä johtuen ketterien menetelmien

haasteissa on myös eroja, ja yhden menetelmän sisältämiä ongelmia ei välttämättä esiinny toisessa ketterässä menetelmässä.

Ennen ketteriä menetelmiä yleisesti käytössä ollut ohjelmistokehitysmenetelmä oli vesiputousmalli, joka on vieläkin laajasti käytössä ohjelmistoalalla (Petersen et al. 2009). Vesiputousmallin esitteli ensimmäisenä Winston Royce vuonna 1987, joten se on syntynyt yli vuosikymmenen ennen ketterien menetelmien syntyä. Vesiputousmallissa on tarkkaan määritellyt, toisiaan seuraavat vaiheet, eikä siinä ole iteratiivisuutta kuten ketterissä menetelmissä. (Yau & Murphy 2013) Vesiputousmallissa on kuusi eri vaihetta (Yau & Murphy 2013):

1. Vaatimusmäärittely
2. Suunnittelu
3. Toteutus
4. Testaus
5. Asennus
6. Ylläpito

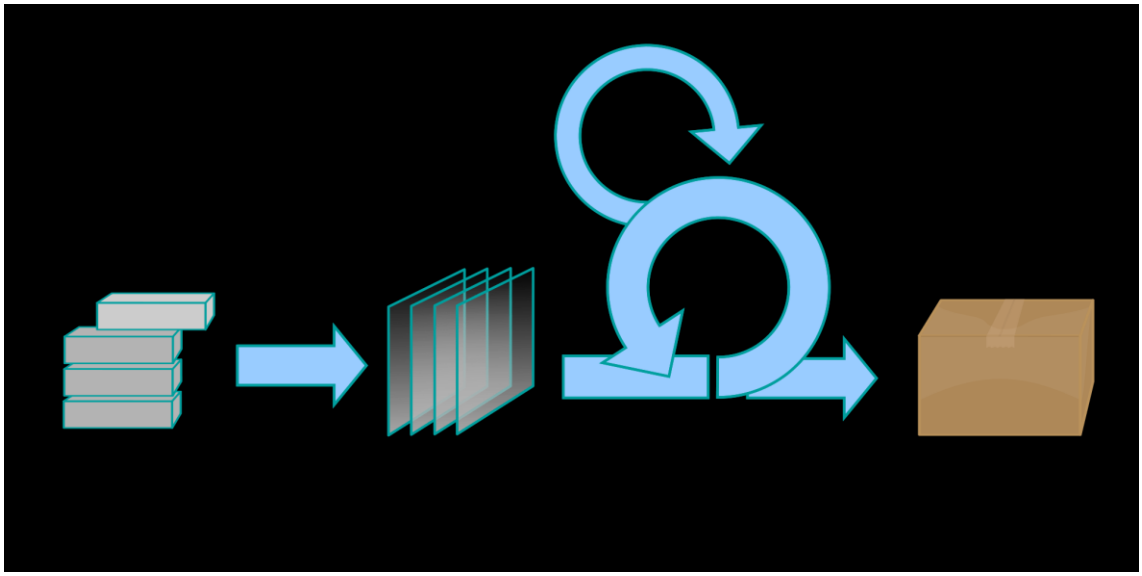
Vesiputousmallissa on kuitenkin useita tunnistettuja puutteita. Näistä on esimerkkinä se, että toisin kuten ketterien menetelmien avulla, vesiputousmallilla ei pystytä vastaamaan muuttuviin asiakasvaatimuksiin. Tämä johtuu suurilta osin siitä, että toisin kuin ketterissä menetelmissä, vesiputousmallissa ei ole itetariivisuutta. Vesiputousmallia käyttämällä tuotettu ohjelmisto ei myöskään välttämättä ole laadultaan yhtä hyvä kuin ketterien menetelmien avulla tuotettu ohjelmisto, sillä testaus tapahtuu vesiputousmallissa vasta myöhäisessä vaiheessa. (Petersen et al. 2009)

Seuraavaksi esitellään neljä ketterää ohjelmistokehitysmenetelmää, jotka on valittu tähän tutkimukseen. Valituista menetelmistä Scrum ja Extreme Programming on valittu niiden yleisyyden takia, sillä ne ovat kaksi yleisintä käytössä olevaa menetelmää (Dingsøyr et al. 2012). Dynamic Systems Development Method ja Feature-driven Development valikoituivat käsiteltäviksi menetelmiksi, koska ne eroavat prosessiltaan ja rooleiltaan paljon edellä mainituista Scrumista ja Extreme Programmingista. Käsittelemällä useita erilaisia ketteriä menetelmiä pyritään tuomaan esille niiden startup-yrityksiin soveltuvuuksien eroja ja löytämään startup-yrityksiin sopivia toimintatapoja useista eri menetelmistä.

3.1 Scrum

Scrum on ketterä ohjelmistokehitysmenetelmä, joka on yhdessä Extreme Programmingin (XP) kanssa yksi yleisimpiä käytössä olevia menetelmiä. Scrum oli myös yksi ensimmäisistä ketteristä menetelmistä, ja sen voidaan nähdä syntyneen Agile Manifeston (Beck et al. 2001) ydinperiaatteiden pohjalta. (Dingsøyr et al. 2012) Scrum perustuu ennalta määrättyihin rooleihin ja toimintatapoihin. Sen toimintaperiaate koostuu tuotteen toiminnallisuuksien pohjalta kehitetystä tuotteen työlistasta. Scrum on iteratiivinen ketterä

menetelmä, joka tarkoittaa sitä, että tuotteen suunnittelua, kehitystä ja testaamista tehdään pienissä osissa. Yhdestä iteraatiosta, eli noin kahdesta neljään viikon kehitysjaksosta käytetään Scrumissa nimeä ”sprintti” ja yhden sprintin päätteeksi valmiina on aina yksi tuotteen osa, joka voidaan toimittaa asiakkaalle. (Schwaber & Beedle 2004, Vuorinen 2011 mukaan) Alla olevassa kuvassa 1 on kuvattu Scrumin toimintaperiaate, eli vasemmalta oikealle tuotteen työlista, sprintin työlista, sprintti ja osittain valmis tuote.



Kuva 1. Scrum-prosessi (Powers 2014)

Yllä olevassa kuvassa 1 tuotteen työlistalla (Product Backlog) tarkoitetaan tuotteen ominaisuuksien perusteella laadittua listaa tuotteen toiminnallisuuksista. Sprintin työlista (Sprint Backlog) taas on yhden tuotteen työlistan tehtävä jaettuna pienempiin osiin, jotka kuvan kohdassa sprintti (Sprint) suunnitellaan ja toteutetaan. Kuvan viimeisenä kohtana olevassa laatikossa (Working increment of the software) on kuvattu valmis tuotteen osa.

Scrumin sisältämiä rooleja ovat Scrum master, tuotteen omistaja (product owner) ja kehitystiimi. Scrum masterin pääasialliset tehtävät ovat Scrumin mukaisten toimintatapojen tukeminen varmistamalla, että kaikki projektissa mukana olevat ymmärtävät Scrumin teorian, toimintatavat, säännöt ja arvot. Hänen tehtäviinsä kuuluu yhteydenpito kehitystiimin ja muun organisaation välillä. Scrum master toimii myös tuotteen omistajan kanssa ja yhdessä he varmistavat projektin sujuvuuden järjestämällä tuotteen työlistan mahdollisimman tehokkaaksi. (Schwaber & Sutherland 2017) Abrahamsson et al. (2002) esittävät artikkelissaan Scrumin rooleiksi myös asiakasta, käyttäjää ja yrityksen johtoa. Heidän mukaansa asiakas osallistuu Scrumin prosessissa esimerkiksi tuotteen työlistan laatimiseen. Yrityksen johdon roolia Scrumissa ei myöskään tule sivuuttaa, sillä he ovat vastuussa oikeiden ihmisten valinnasta muihin rooleihin (Abrahamsson et al. 2002). Scrum on rooleiltaan selvä kokonaisuus ja jokaiselle roolille on määritelty tarkat vastuualueet, mikä on varmasti hyödyllistä nopeasti muuttuvassa toimintaympäristössä.

Tuotteen omistajan rooli on olla vastuussa kehitystiimin tekemän työn arvon maksimoinnista. Hänen tärkein tehtävänsä on tuotteen työlistan ylläpito. Työlistan ylläpitoon kuuluu muun muossa työlistan järjestäminen mahdollisimman tehokkaaseen järjestykseen ja sen varmistaminen, että kehitystiimi ymmärtää tuotteen työlistassa olevat työtehtävät. (Schwaber & Sutherland 2017) Tuotteen omistajan roolin voidaan nähdä olevan tärkeä, sillä tuotteen työlista on koko Scrumin sisältämän prosessin aloituspiste, jonka perusteella määräytyy myös koko prosessin eteneminen.

Kehitystiimin vastuualueena on jokaisen sprintin aikana tehdä sprintin tehtäväjonossa ennalta määritellyt tehtävät. Kehitystiimin tavoitteena on jokaisen sprintin päätteeksi saada valmiiksi mahdollisesti toimitusvalmis tuote, joka sisältää ennalta määrätyt ominaisuudet. Scrumin periaatteiden mukaan kehitystiimit ovat itseorganisoituvia ja heillä on täysi päätäntävalta siitä, miten tuotteen kehitysjonon sisältävät palaset muutetaan toimivaksi ohjelmistoksi. (Schwaber & Sutherland 2017) Kehitystiimin tehtäviin kuuluu myös olla mukana tuotteen työlistan sisältämien töiden vaativuuden arvioinnissa (Abrahamsson et al. 2012). Scrumin sisältämistä rooleista juuri kehitystiimin työn jälki näkyy parhaiten asiakkaalle, joten se on myös koko yrityksen näkökulmasta erittäin tärkeä rooli.

3.2 Extreme Programming

Extreme Programming on yksi ensimmäisistä ketteristä ohjelmistokehitysmenetelmistä ja sen voidaan nähdä pohjautuvan Agile Manifeston (Beck et al. 2001) ydinperiaatteisiin (Dingsøyr et al. 2012). Tästä johtuen se sisältää hyvin paljon samanlaisia elementtejä kuin edellä esitelty Scrum. Extreme Programmingin ytimessä on asiakastyytyväisyyden korostaminen. Tähän päästään Wellsin (2013) mukaan nopeilla iteraatioilla, joissa ohjelmistoa toimitetaan asiakkaalle pienissä osissa lisäämällä uusia ominaisuuksia jokaisessa iteraatiossa. Lyhyiden iteraatioiden avulla pystytään saavuttamaan myös Agile Manifeston (Beck et al. 2001) kuvailema ketterien menetelmien ydinajatus siitä, että ketterien menetelmien pitäisi pystyä vastaamaan muuttuviin asiakasvaatimuksiin.

Extreme Programmingin neljä ydinarvoa ovat yhteydenpito, yksinkertaisuus, palaute ja rohkeus. Näiden neljän ydinarvon pohjalta on kehitetty 12 aktiviteettia, joita ovat (Beck 1999, Erickson et al. 2005 mukaan)

1. suunnittelupeli
2. pienet julkaisut
3. metaforat
4. yksinkertainen suunnittelu
5. testaus
6. koodin uudelleenkäyttö
7. pariohjelmointi
8. yhteisomistus
9. jatkuva integrointi

10. 40-tuntiset työviikot
11. paikan päällä oleva asiakas
12. ohjelmointistandardit

Yllä olevassa listassa suunnittelupelillä tarkoitetaan suunnittelua, jossa ovat mukana sekä ohjelmiston kehittäjät että asiakkaat. Pienet julkaisut taas perustuvat Agile Manifeston (Beck et al. 2001) esittämään periaatteeseen lyhyistä iteraatioista. Aktiviteeteista metaforien käytöllä tarkoitetaan tuotteen määrittämistä metaforien avulla, jolloin asiakas voi paremmin ymmärtää tuotteen toimintaa ja osallistua tämän kautta myös sen suunnitteluun. Yksinkertaisen suunnittelun ideana on yksinkertaisesti tuottaa mahdollisimman kevyt, mutta toteutettavissa oleva suunnitelma. Testaus on merkitty yhdeksi aktiviteetiksi, koska Extreme Programmingin yhtenä ajatuksen on kirjoittaa yksikkötestit enne koodia. Koodin uudelleenkäyttö on tärkeä aktiviteetti, sillä se yksinkertaistaa koodia ja lisää sen joustavuutta. Pariohjelmointi taas tarkoittaa kahden ohjelmoijan työskentelyä samalla koneella. Yhteisomistus on myös tärkeä osa Extreme Programmingin periaatteita, sillä sen mukaan jokainen projektissa mukana oleva voi muuttaa mitä tahansa osaa koodista. Jatkuvan integroinnin ideana on lisätä uutta koodia järjestelmään pienissä paloissa ja heti sen ollessa valmiina. Aktiviteettina 40-tuntiset työviikot tarkoittavat sitä, että tuotteen parissa ei tulisi työskennellä ylitöitä, ja mikäli näin tapahtuu, tulee ongelma korjata. Paikan päällä oleva asiakas tarkoittaa sitä, että Extreme Programmingin periaatteiden soveltamiseksi asiakkaan tulisi fyysisesti olla samassa tilassa kuin kehitystiimi. Viimeisenä aktiviteettina oleva koodausstandardit määrittelevät tietyt säännöt, joita tuotteen kehittäjien pitää seurata koko projektin ajan. (Abrahamsson et al. 2002) Vaikkakin Extreme Programming sisältää paljon erilaisia aktiviteetteja, ovat ne kaikki helposti ymmärrettäviä ja selvästi ketterän kehityksen periaatteita tukevia.

Wellsin (2013) mukaan Extreme Programming perustuu tiimityöhön, jossa johtajat, asiakkaat ja kehittäjät toimivat tasa-arvoisina kumppaneina projektissa. Scrumin kaltaisesti myös Extreme Programming uskoo itseorganisoituvien tiimien tuottavan tehokkaimman mahdollisen ratkaisun. Extreme Programming sisältää tiimityön painottamisen lisäksi ydinarvoihin ja aktiviteetteihin perustuvia sääntöjä. Säännöt sisältävät toimintatapoja, jotka liittyvät projektin suunnitteluun, johtamiseen sekä ohjelmiston suunnitteluun, kehittämiseen ja testaamiseen. Projektin suunnitteluun ja johtamiseen liittyviä sääntöjä ovat esimerkiksi lyhyiden iteraatioiden sekä avoimen työtilan käyttäminen, iteraatiosuunnitelmien laatiminen ja projektin etenemisen mittaaminen. Ohjelmistojen suunnitteluun, kehittämiseen ja testaamiseen liittyviä sääntöjä ovat esimerkiksi yksinkertaisuus, toiminnallisuuksien lisääminen vain, kun niitä tarvitaan, koodin uudelleenkäyttö, asiakkaan nopea tavoitettavuus ja yksikkötestien kirjoittaminen ennen koodia. (Wells 2013) Extreme Programming sisältää enemmän erilaisia toimintatapoja ja sääntöjä kuin esimerkiksi edellä esitelty Scrum, ja tästä syystä myös sen käyttöönoton ja ylläpidon voidaan nähdä olevan Scrumia haastavampaa.

3.3 Dynamic Systems Development Method

Dynamic Systems Development Method (DSDM) on ketterä ohjelmistokehitysmenetelmä, joka kehitettiin jo vuonna 1994 Rapid Application Developmentin (RAD) pohjalta. Voittoa tavoittelemattoman Agile Business Consortiumin mukaan DSDM:n filosofia on ”paras kaupallinen arvo syntyy, kun projekteilla on selvät kaupalliset tavoitteet, nopea toimitustahti ja kun motivoidut ihmiset työskentelevät yhdessä”. (Stapleton 1997 s. 11). Yleisemmällä tasolla se perustuu terveen järjen käyttöön ja pragmatismiin. Ketterien menetelmien perustan, Agile Manifeston (Beck et al. 2001), mukaisesti DSDM:n yksi kulmakivistä on projektin jakaminen pieniin osiin ja inkrementaalinen ohjelmistokehitys. (Agile Business Consortium 2014) Abrahamsson et al. (2002) mukaan DSDM:n perimmäinen idea on se, että yrityksen on projektia suunnitellessaan järkevämpää pitää resurssit sekä käytettävä aika vakiona ja muuttaa tuotettavan ohjelmiston toiminnallisuuksia niiden mukaan.

Agile Business Consortium (2014) puolestaan määrittelee DSDM:n kahdeksan periaatetta seuraavanlaisesti:

1. Keskity liiketoiminnan tarpeisiin
2. Toimita ajallaan
3. Tee yhteistyötä
4. Älä koskaan tingi laadusta
5. Rakenna inkrementaalisesti vakaista perustuksista
6. Kehitä iteratiivisesti
7. Kommunikoi jatkuvasti ja selvästi
8. Kontrolloi kokonaisuutta

Nämä periaatteet peilaavat hyvin myös Agile Manifeston (Beck et al. 2001) määrittelemiä ketterän kehityksen periaatteita.

Verrattuna esimerkiksi Scrumiin, DSDM sisältää melko paljon erilaisia ennalta määriteltyjä rooleja työntekijöille. Agile Business Consortium (2014) määrittelee käsikirjassaan ainakin 13 eri roolia, joista osa on määritelty teknisimmiksi ja osa käsittelee joko projektin liiketoiminnallista tai projektinhallinnallista puolta. Rooleja ovat esimerkiksi projektipäällikkö, tekninen neuvonantaja, liiketoiminnallinen neuvonantaja, kehittäjä, testaaja ja DSDM valmentaja.

Roolien ja periaatteiden lisäksi DSDM sisältää hyvin tarkkaan määritellyn prosessin, jonka mukaan liiketoiminnallisesti tärkeimmät asiat pitäisi hoitaa ajoissa, kun taas vähemmän tärkeät ominaisuudet hoidetaan myöhemmässä vaiheessa (Agile Business Consortium 2014). Prosessin ensimmäisenä vaiheena on toteutettavuuden arviointi, joka sisältää arvioinnin siitä, onko projekti teknisesti ja liiketoiminnallisesti järkevä toteuttaa (Stapleton 1997 s. 3). Toinen vaihe on projektin perustan rakentaminen, jolloin tulisi

selvittää projektin liiketoiminnalliset perusteet ja toteuttaa suunnitelma mahdollisesta ratkaisusta. Tässä vaiheessa projektia ei kuitenkaan suunnitella liian tarkasti, jotta ratkaisujen muuttaminen on mahdollista myöhemmässä vaiheessa. (Agile Business Consortium 2014)

Suunnittelun jälkeisistä prosessin vaiheista on kirjallisuudessa useita versioita. Stapleton (1997 s. 3) esittelee DSDM:n prosessin viisiosaisena, kun taas Agile Business Consortium (2014) määrittelee prosessiin useamman osan. Jälkimmäinen on myös kirjallisuudessa useimmin käytetty versio DSDM:n prosessista. Suunnittelun jälkeisiä vaiheita ovat sen mukaan kehitysvaihe, käyttöönotto, yhteenveto, projektin arvostelu, projektin päättäminen ja projektin tulosten tarkastelu (Agile Business Consortium 2014). DSDM on siis kokonaisuudessaan raskaampi ja kokonaisvaltaisempi ketterä menetelmä kuin esimerkiksi Scrum, sillä se sisältää paljon sekä projektinhallinnallisia että itse ohjelmiston tuottamiseen liittyviä periaatteita ja toimintatapoja.

3.4 Feature-driven Development

Viimeisenä tarkasteltavana ketteränä menetelmänä on Feature-driven development (FDD). Se on ketterä menetelmä, joka on Scrumin ja Extreme Programmingin kaltaisesti syntynyt Agile Manifeston (Beck et al. 2001) ydinajatusten pohjalta (Dingsøyr et al. 2012). Sen periaate on kehittää ohjelmistoja asiakkaan arvostamien ominaisuuksien pohjalta. Se on ketterien menetelmien perustan mukaisesti iteratiivinen kehitystapa. Se keskittyy pääasiallisesti ohjelmiston suunnitteluun ja toteutukseen, projektinhallinnallisten aspektien sijaan. (Nawaz et al. 2017) FDD keskittyy myös laadun ylläpitämiseen prosessin edetessä ja painottaa projektin etenemisen mittaamisen tärkeyttä (Abrahamsson et al. 2002). Yleisellä tasolla ketterien menetelmien yhtenä huonona puolena on nähty se, että niiden avulla ei pystytä kehittämään kriittisiä ohjelmistoja, mutta Abrahamssonin et al. (2002) mukaan tätä ongelmaa ei ole FDD:n kanssa.

FDD:n periaatteisiin kuuluu viisiosainen prosessi sekä toimintatavat ja tekniikat. Prosessin ensimmäinen osa on kokonaismallin rakentaminen. Sen tarkoituksena on saada projektin jäsenille hyvä kokonaisymmärrys projektista. Tässä vaiheessa projekti myös jaetaan pienempiin osiin ja päätetään miten sitä pitäisi lähestyä. Prosessin toinen vaihe on ominaisuuslistan laatiminen. Tässä vaiheessa esitellään kaikki asiakkaalle arvokkaat ominaisuudet. Loppukäyttäjät ja muut sidosryhmät saavat ominaisuuslistan hyväksyttäväkseen. Prosessin kolmantena vaiheena on jokaisen ominaisuuden suunnittelu yleisellä tasolla. Tässä vaiheessa suunnitellaan myös projektin aikataulu ja prosessin ensimmäisessä vaiheessa määritellyt yksittäiset luokat osoitetaan yksittäisten ohjelmoijien työksi. Prosessin neljännessä ja viidennessä vaiheessa yksittäiset ominaisuudet suunnitellaan tarkemmin ja toteutetaan. Tämä vaihe on FDD:n prosessin iteratiivinen osa. Jokaiseen iteraatioon valitaan tietty määrä ominaisuuksia toteutettavaksi. Yhdessä iteraatiossa voidaan eri ominaisuuksia toteuttaa eri tiimeissä. Jokaisen iteraation jälkeen valmiiksi toteutetut

ominaisuudet lisätään tuotannossa olevaan ohjelmistoon. (Palmer & Felsing 2001, Abrahamsson et al. 2002 mukaan)

Muiden edellä esiteltyjen ketterien menetelmien tapaan myös FDD sisältää ennalta määriteltyjä rooleja. Ensimmäiseen kategoriaan eli avainrooleihin kuuluu projektipäällikkö, pääarkkitehti, pääohjelmoija, luokan omistaja ja vaatimusasiantuntija. Toiseen kategoriaan eli tukeviin rooleihin kuuluu julkaisupäällikkö, kielijuristi, ohjelmointi-insinööri, apuohjelmien kehittäjä ja järjestelmän ylläpitäjä. Kolmanteen kategoriaan kuuluu esimerkiksi testaajat ja käyttöönottajat. FDD ei kuitenkaan rajoita sitä, montako roolia yhdellä henkilöllä on, vaan yksi ihminen saattaa toimia useammassa roolissa. (Palmer & Felsing 2001, Abrahamsson et al. 2002 mukaan) Roolien puolesta FDD on siis huomattavasti joustavampi kuin esimerkiksi Scrum.

Periaatteiden ja roolien lisäksi FDD sisältää myös parhaaksi todettuja toimintatapoja, joita FDD:tä käyttävien projektien tulisi soveltaa. Näitä ovat esimerkiksi toiminnallisuuksien mukaan kehittäminen, joka mahdollistaa projektin valmiusasteen seuraamisen tarkasti, ja luokkaomistajuus, jossa yhdelle luokalle on määritelty vain yksi vastuuhenkilö, joka on vastuussa luokan yhtenäisyydestä. Muita toimintatapoja ovat ominaisuuksien mukaan järjestäytyvät koodaustiimit, koodin tarkastaminen, säännölliset kokonaisuuden testaamiset ja asetusten hallinta. (Palmer & Felsing 2001, Abrahamsson et al. 2002 mukaan)

3.5 Yhteenveto

Ketteristä menetelmistä käsiteltiin erikseen neljä eri menetelmää: Scrum, Extreme Programming, Dynamic Systems Development Method ja Feature-driven Development. Näistä Scrum ja Extreme Programming ovat eniten käytettyjä ja myös tunnetuimpia ketteriä menetelmiä. Nämä kaksi ovat myös käsitellyistä menetelmistä ne, jotka tarkimmin perustuvat Agile Manifeston (Beck et al. 2001) määrittelemiin periaatteisiin. Menetelmistä DSDM on kokonaisvaltaisempi, kun taas esimerkiksi Scrum keskittyy lähinnä ohjelmiston tuottamiseen liittyviin prosesseihin ja rooleihin.

Kaikki esitellyt menetelmät sisältävät prosessimallin sekä tiettyjä rooleja. Raskaimpina käsitellyistä menetelmistä voidaan pitää DSDM:a ja FDD:tä, sillä nämä määrittelevät tarkimmin roolinsa, kun taas Scrum määrittelee vain kolme roolia. Extreme Programming määrittelee roolinsa melko epätarkasti ja FDD mahdollistaa useamman roolin asettamisen yhdelle ihmiselle.

Roolien lisäksi kaikki esitellyt ketterät menetelmät sisältävät myös prosessimallin. Scrumin prosessimalli on muihin verrattuna melko yksinkertainen, sillä siinä projekti on jaettu pienempiin osiin, joita tuotetaan iteraatioissa. Extreme Programmingin prosessiin liittyy useita periaatteita ja sääntöjä, joita tulee seurata ja prosessi sisältää Scrumin tapaisesti iteratiivista ohjelmistokehitystä. DSDM:n prosessi on muihin esiteltyihin menetelmiin verrattuna kokonaisvaltaisempi ja se sisältää vaiheita, joita toteutetaan myös ennen

projektin aloitusta, kuten toteutettavuuden arvioinnin. FDD:n prosessi on selvästi määritelty ja viisiosainen. Se jakaa projektin Scrumin tapaisesti pieniin osiin ja jokainen osa sisältää yhden toiminnallisuuden.

Käsitellyt menetelmät eroavat myös siinä, paljonko ne sisältävät suunnittelua ennen toteutusvaihetta. Ketterien menetelmien yhtenä pääideana pidetään sitä, että muuttuviin asiakasvaatimuksiin pystytään vastaamaan nopeasti. Käsitellyistä menetelmistä esimerkiksi FDD sisältää melko paljon suunnittelua ennen kuin ohjelmaa aletaan ohjelmoimaan, kun taas Scrumin prosessimallin mukaan suunnittelun tulee tapahtua pääasiallisesti pienissä osissa iteraatioiden aikana.

4. OHJELMISTOALAN STARTUP-YRITYKSET

Ohjelmistoalan startup-yritysten voidaan nähdä eroavan monilta osin alan suurista yrityksistä. Tämä aiheuttaa niille haasteita, mutta avaa myös mahdollisuuksia kilpailla suuria yrityksiä vastaan. Eroavaisuuksia suuriin yrityksiin on varsinkin Suttonin (2000) kuvailemalla neljällä osa-alueella, joita ovat toimintahistoria, resurssit, sidosryhmät ja käytettävät teknologiat.

4.1 Nykyiset toimintatavat

Ohjelmistoalan startup-yritykset kohtaavat toiminnassaan monimutkaisia ongelmia ja haasteita (Paternoster et al. 2014). Yksi ohjelmistoalan startup-yritysten suurimmista haasteista on kehitysprosessin luominen ja johtaminen (Coleman & O'Connor 2008). Tutkimuksessaan Coleman & O'Connor (2008) kertovat, että tyypillisesti ohjelmistoalan startup-yritykset käyttävät kehitysmenetelmänään perinteistä vesiputousmallia tai joissain tapauksissa Extreme Programmingia. Näitä menetelmiä on kuitenkin yrityksissä muokattu toimintaympäristöön sopivaksi (Coleman & O'Connor 2008). Ohjelmistomenetelmien räätälöinti saattaa olla suuressa roolissa yritysten menestyksen määrittelyssä, sillä yksi tietty menetelmä tarkasti seurattuna tuskin sopii useille, eri tyyppisille startup-yrityksille.

Ohjelmistoalan startup-yritykset eroavat alan suurista yrityksistä monin tavoin. Organisaatioiden koko, erilaiset lähestymistavat, metodit, tavoitteet ja resurssit vaikuttavat startup-yritysten toimintaan markkinoilla. Eroavaisuudet luovat startup-yrityksille haasteita, mutta osaltaan parantavat myös niiden kilpailukykyä isoja toimijoita vastaan. Startup-yritykset keskittyvät usein pieniin markkina-alueisiin, joihin suuret ohjelmistoalan yritykset eivät syystä tai toisesta ole lähteneet mukaan. Niiden joustava ja mukautuva organisaatiorakenne sopii hyvin epävarmoihin ja dynaamisiin toimintaolosuhteisiin. (Richardson & von Wangenheim 2007)

Startup-yritysten on pienen kokonsa takia usein vaikeaa keskittyä itse yrityksen tuotekehityksen ulkopuolisiin asioihin (Richardson & von Wangenheim 2007). Niiden on myös dynaamisuutensa vuoksi melko vaikeaa suunnitella ennalta määrätty prosessimalli, jonka mukaan voitaisiin toimia kaikissa projekteissa. Prosessien vakiinnuttaminen on vaikea myös siksi, että uusissa yrityksissä ei ole vielä tarpeeksi paljon tietämystä ja rutiineja. Haasteita aiheuttaa usein myös työntekijöiden vähäinen määrä. (Sutton 2000) Ohjelmistoalan startup-yritykset omaksuvat erilaisia menetelmiä yleensä vasta melko myöhäisessä vaiheessa elinkaartaan, sillä aikaisessa vaiheessa niiden päähuomio keskittyy lähinnä yrityksen selviytymiseen (Coleman & O´Connor 2008).

Coleman & O'Connorin (2008) mukaan useat startup-yritysten johtajat päätyvät käyttämään johtamisessaan hyväksi kokemiaan prosesseja ja toimintatapoja. Heidän mukaansa ohjelmistoyritykset ovat hyvinkin riippuvia juuri johtajien omasta taustasta ja kyvyistä luoda teknologisia ratkaisuja pienillä resursseilla ja rajoitetussa ajassa. Coleman & O'Connor (2008) mainitsevat myös, että ohjelmistoalan startup-yrityksissä johtajat pyrkivät usein saamaan yrityksen kaikki työntekijät osallistumaan ohjelmistokehityksen jokaiseen vaiheeseen. Niissä on myös tapahtunut muutosta sen suhteen, kuinka ennalta saneltua ohjelmistokehitys on, johtaen siihen, että ohjelmiston kehittäjät pääsevät nykyään osallistumaan yhä enemmän heitä koskeviin päätöksiin. (Coleman & O'Connor 2008)

Paternoster et al. (2014) mukaan ohjelmistoalan startup-yritysten nykyisin käyttämät ohjelmistokehitysmenetelmät perustuvat usein monista eri menetelmistä otettuihin osiin. Sekä perinteisten, että ketterien menetelmien käyttö sellaisenaan on usein vaikeaa yllä mainituista syistä. Vaikeaa ketterien menetelmien käytöstä tekee myös se, että yritykset toimivat usein epävarmassa ja nopeasti muuttuvassa toimintaympäristössä. Useiden eri menetelmien käyttö päällekkäin edistää yritysten kykyä reagoida muuttuvaan ympäristöön. Käytettävät menetelmät keskittyvät usein vain itse ohjelmistojen tuottamiseen, eikä johtamiseen tai organisaation toimintaan liittyviin toimintatapoihin kiinnitetä juurikaan huomiota. (Paternoster et al. 2014)

Perinteisen vesiputousmallin käytössä ohjelmistoalan startup-yrityksissä on sekä hyviä että huonoja puolia. Hyviä puolia ovat esimerkiksi sen yksiselitteisyys ja helppo ymmärrettävyys. Vesiputousmallin ensimmäinen vaihe, vaatimusmäärittely, tarjoaa hyvin tehtynä selvän vision, jonka mukaan startup-yritys voi lähteä kehittämään ohjelmistoaan. (Yau & Murphy 2013) Klotins et al. (2018) mukaan ohjelmistoalan startup-yritykset ovat kuitenkin usein markkinavetoisia, jolloin asiakkaan vaatimusten tarkka määrittely saattaa olla vaikeaa. Vesiputousmallin hyvänä puolena voidaan nähdä myös se, että sen käyttö vähentää tarvetta kommunikaatiolle tuotantovaiheessa ja näin vähentää mahdollisuutta väärinkäsityksille. Tämä johtuu tarkasti tehdystä vaatimusmäärittelystä sekä laajasta dokumentaatiosta. (Yau & Murphy 2013)

Vesiputousmallin käyttämisen huonona puolena Yau & Murphy (2013) pitävät sitä, että muuttuviin asiakasvaatimuksiin ei pystytä vastaamaan yhtä nopeasti kuin ketteriä menetelmiä käyttämällä. Startup-yritykset toimivat usein hyvin dynaamisilla markkinoilla, jolloin niiden tuotteiden visio ja laajuus saattaa vaihdella rajustikin. Vesiputousmallin prosessin vaiheet ovat hyvin riippuvaisia prosessin edellisen vaiheen tuloksista, jolloin muuttuviin vaatimusmäärittelyihin vastaaminen ohjelmiston tuotannossa on vaikeaa. (Yau & Murphy 2013) Edellä mainittu dynaamisuuden puuttuminen on yleisesti tunnistettu ongelma vesiputousmallin käytössä myös muissa kuin startup-yrityksissä ja yksi suurimmista syistä siihen, että ketterät menetelmät on alun perin kehitetty.

4.2 Ketterät menetelmät startup-yrityksissä nykyään

Klotins et al. (2018) mukaan startup-yritysten nykyisin käyttämät kehitysmenetelmät ovat osittain melko lähellä ketteriä menetelmiä, sillä käytettävät menetelmät sisältävät usein iteraatiota ja jatkuvaa suunnittelua. Ketterien menetelmien haasteena startup-yrityksissä on kuitenkin se, että ne toimivat markkinavetoisessa ympäristössä, jossa yrityksille on vaikeaa määritellä tiettyä asiakasta. Ketterän kehityksen yksi kulmakivistä on juurikin asiakkaan läsnäolo ja tiivis yhteydenpito koko kehitysprosessin ajan. (Klotins et al. 2018) Paternoster et al. (2014) tiivistävät startup-yrityksissä hyviksi todetuiksi menetelmiksi kevyet menetelmät, jotka osaltaan ylläpitävät yrityksen kykyä pysyä joustavana ja nopeasti muutoksiin sopeutuvana. Kirjoittajat pitävät myös tärkeänä prototyyppien kehittämistä käyttäjiltä kerätyn palautteen perusteella, sillä tämä lisää yrityksen kykyä toimia epävarmoilla markkinoilla. (Paternoster et al. 2014) Coleman & O'Connor (2008) löysivät tutkimuksessaan, että startup-yritykset käyttävät ohjelmistokehityksessään usein mitä tahansa prosessia, joka tukee yrityksen liiketoiminnan tavoitteita. Tämä johtuu kirjoittajien mukaan siitä, että startup-yrityksillä ei yksinkertaisesti ole riittävästi resursseja pyrkiä löytämään mahdollisimman tehokasta tapaa kehittää ohjelmistoja.

5. KETTERIEN MENETELMIEN SOVELTUVUUS STARTUP-YRITYKSIIN

Ohjelmistoalan startup-yritysten toimintaympäristö eroaa monilta osin suurempien yritysten toimintaympäristöstä ja tämä aiheuttaa sen, että toimintamalli, joka toimii suurilla yrityksillä ei välttämättä toimi startup-yrityksillä. Ketterillä menetelmillä on monia etuja verrattuna perinteisiin ohjelmistokehitysmenetelmiin, mutta varsinkin pienillä yrityksillä edut ovat kiistanalaisia, sillä niillä ei välttämättä edes esiinny ongelmia, joihin ketterät menetelmät ollaan suunniteltu vastaamaan. (Yau & Murphy 2013) Klotins et al. (2018) huomauttavat, että on tärkeää tutkia, mitkä monista menetelmistä ja toimintatavoista sopivat juuri startup-yrityksiin. Coleman & O'Connor (2008) mukaan ohjelmistoalan startup-yritysten yksi suurimmista haasteista on vakiinnuttaa menetelmät, joita yrityksen ohjelmistokehityksessä käytetään. Heidän mukaansa ketterillä ohjelmistokehitysmenetelmillä on paljon annettavaa startup-yrityksille, sillä niillä on samanlaisia piirteitä, kuten se, että molemmissa painotetaan ohjelmoijien asemaa kehityksessä. Myös Paternoster et al. (2014) tunnistavat startup-yritysten toimintaympäristön monimutkaisuuden ja heidän mukaansa startup-yritysten käyttämien ohjelmistokehitysmenetelmien pitäisi tukea yritysten dynaamisuutta toimintaympäristössään.

Startup-yritysten käyttämät ohjelmistokehitysmenetelmät muovaantuvat usean tekijän summana, mutta Coleman & O'Connor (2008) pitävät suurimpana tekijänä yrityksen ohjelmistokehityksen johtajan asemaa. Heidän mukaansa on tärkeää, että ohjelmistokehityksestä vastuussa oleva johtaja ymmärtää hyvin myös liiketoiminnan tavoitteita ja markkinoita, joihin yritys pyrkii. Johtamistyylillä on myös Coleman & O'Connor (2008) mukaan suuri vaikutus yrityksen menestykseen. Myös Sutton (2000) tunnistaa startup-yrityksen johtavissa rooleissa olevien ihmisten suuren vaikutuksen yrityksen menestykseen. Hänen mukaansa jatkuvasti muuttuvassa ympäristössä on tärkeää, että startup-yrityksessä on ihmisiä, jotka pystyvät jatkuvasti viemään yritystä oikeaan suuntaan ja samalla toimimaan johtavassa asemassa.

Ketterien menetelmien tuottamat edut startup-yrityksiin riippuvat monesta tekijästä. Näistä tärkeimpiä ovat se, kuinka tarkasti ketterien menetelmien periaatteita ja toimintatapoja seurataan, sekä se, missä vaiheessa tarkasteltava startup-yritys on. Yritykset yleisesti käyttävät ketteriä menetelmiä hyvin eri tavalla, ja usein menetelmistä sekä niiden toimintatavoista tehdään erilaisia yhdistelmiä. Startup-yrityksen koko vaikuttaa ketterien menetelmien vaikutukseen, sillä sen tuottamat edut muuttuvat startup-yrityksen kasvaessa. Myös ketterien menetelmien käyttöönoton kustannukset muuttuvat startup-yrityksen kasvun seurauksena. (Yau & Murphy 2013) Fagerholm et al. (2015) mukaan myös yrityksen sisäiset tekijät vaikuttavat ohjelmistomenetelmien tuottamiin etuihin. Sisäisiä tekijöitä voivat olla esimerkiksi kommunikaatio, tiimihenki ja arvot. Panostamalla näihin

voidaan kirjoittajien mukaan vaikuttaa positiivisesti tiimin tehokkuuteen. Klotins et al. (2018) mukaan Fagerholm et al. (2015) tekemän tutkimuksen tulokset ovat myös yleistettävissä startup-yrityksissä toimiviin tiimeihin. Coleman & O'Connor (2008) mukaan prosessin yksilöllinen suunnittelu tai räätälöinti yrityksen tilanteen mukaan vaikuttaa myös menetelmien tuottamiin etuihin.

Ketterien menetelmien etuja on kirjallisuudessa käsitelty laajasti. Ensimmäinen suuri etu on se, että Agile Manifeston (Beck et al. 2001) mukaan toimiessa yritykset ovat lähempänä asiakastaan ja palautteen vastaanottaminen asiakkaalta on helppoa. Tämä johtaa laadukkaampaan ja paremmin asiakkaan toiveisiin vastaavaan tuotteeseen. Itseorganisoituvien tiimien nähdään toimivan tehokkaammin ja tuottavan laadukkaampia ohjelmistoja, kuin jos tiimien kokoonpano ja toimintatavat olisi määritelty ennalta. (Dingsøyr et al. 2012) Toinen yleisesti tunnistettu ketterien menetelmien etu on se, että ketteriä menetelmiä käyttämällä yritykset pystyvät vastaamaan nopeasti esimerkiksi toimintaympäristössä, asiakasvaatimuksissa tai aikarajoissa tapahtuviin muutoksiin. (Erickson et al. 2005) Molemmat edellä mainitut edut ovat kiistattomia ja hyvin sovellettavissa myös startup-yrityksiin, sillä myös startup-yritykset hyötyvät läheisestä asiakasyhteistyöstä sekä mahdollisuudesta vastata muuttuviin projektin aspekteihin nopealla aikataululla. Startup-yritykset itsessään toimivat myös dynaamisessa ympäristössä ja niille on tärkeää pystyä muuttamaan yrityksen kurssia nopealla aikataululla tilanteen niin vaatiessa.

Ohjelmistoalan startup-yrityksen käyttämällä ohjelmistokehitysmenetelmällä tai -prosessilla on suuri merkitys niiden toimintaan. Ohjelmistokehitys tapahtuu aina jonkun prosessin kautta, oli se sitten suunniteltua tai ei. Prosessin käyttämisen lisäksi myös sen toistettavuudella on suuri vaikutus yrityksen toimintaan, sillä jos yritys pystyy toistamaan prosessinsa projektista toiseen, se pystyy myös tehokkaasti suunnittelemaan tulevien projektien aikatauluja sekä tehostamaan käyttämiään prosessejaan oppimisen kautta. Aloittelevan startup-yrityksen ei kuitenkaan tulisi määritellä prosessejaan liian tarkasti, sillä tämä saattaa johtaa siihen, että prosesseja ei seurata tarkasti. (Sutton 2000)

Yau & Murphy (2013) muistuttavat, että ketterät menetelmät eivät välttämättä ainakaan tarkasti noudatettuna sovi ohjelmistoalan startup-yrityksiin, sillä niillä ei esiinny samanlaisia ongelmia esimerkiksi organisaation sisäisessä viestinnässä kuin suuremmilla yrityksillä. Heidän mukaansa startup-yritysten pitäisi prosessinmallin kehittämisessä keskittyä ohjelmistojen kehittäjien tukemiseen, eikä yrittää liian voimakkaasti pakottaa jonkinlaista ennalta määriteltyä prosessia tai menetelmää. Startup-yritysten ohjelmistokehittäjät eivät usein vaadi tarkkoja tehtävälistoja tai ketterien menetelmien tuomia etuja motivaatioon. (Yau & Murphy 2013) Coleman & O'Connor (2008) mukaan yrityksen prosessin luomisessa yksi osa on jonkin valmiin prosessin muovaaminen omaan käyttöön. Tämä tarkoittaa sitä, että yritykset saattavat karsia huomattaviakin määriä valmiista ketterästä menetelmästä löytyviä ominaisuuksia tai toimintatapoja. Prosessin räätälöintiä tapahtuu sekä prosessin luomisen yhteydessä että koko yrityksen elinkaaren ajan. (Coleman & O'Connor 2008)a

5.1 Scrum

Aliluvussa 3.1 esitellyn Scrumin voidaan nähdä keskittyvän siihen, miten tiimin jäsenien pitäisi toimia, jotta ohjelmistoja voidaan tuottaa mahdollisimman joustavasti jatkuvasti muuttuvassa ympäristössä (Abrahamsson et al. 2002). Scrum sisältää ennalta määrätyn prosessin, joka perustuu tuotteen ominaisuuksien perusteella tehtyyn tuotteen työlistaan sekä iteratiiviseen ohjelmiston suunnitteluun, kehittämiseen ja testaamiseen. Jokaisessa sprintissä eli iteraatiossa päätetään, mitä pyritään saamaan valmiiksi ja näiden perusteella toteutetaan ohjelmistoa eteenpäin. Jokaisen sprintin päätteeksi valmiina on ohjelmiston osa, joka voidaan toimittaa asiakkaalle. Scrum keskittyy prosessin lisäksi myös ennalta määrättyihin rooleihin, joita on kolme kappaletta, Scrum master, tuotteen omistaja ja ohjelmistokehittäjä.

Scrumin prosessin sopivuutta startup-yrityksiin voidaan tarkastella monelta kannalta. Suttonin (2000) mukaan joustavuus on tärkeä osa startup-yrityksen ohjelmistokehitystä, jotta yritys pystyy vastaamaan jatkuviin muutoksiin henkilöstössä, asiakasvaatimuksissa, resursseissa ja aikatauluissa. Abrahamsson et al. (2002) mukaan Scrum on juurikin joustavuuteen keskittyvä ketterä ohjelmistokehitysmenetelmä. Yau & Murphy (2013) määrittelevät artikkelissaan Scrumin eduiksi startup-yrityksissä päivittäiset tiimin kokoontumiset, jotka heidän mukaansa vähentävät mahdollisuutta väärinkäsityksille ja näiden mukana tuleville ylimääräisille kuluille. Toinen Scrumin prosessiin keskittyvä etu startup-yrityksissä on sprintin tehtävälistan luominen ja seuraaminen. Tämä auttaa startup-yrityksen ohjelmistokehittäjiä keskittymään kehitettävän ohjelmiston tärkeisiin ominaisuuksiin. Scrumin prosessi kokonaisuudessaan ja varsinkin sen iteratiivinen luonne auttaa startup-yritystä kulkemaan kohti tavoitteitaan. (Yau & Murpy 2013) Kokonaisuudessaan Scrumin sisältämän prosessin voidaan siis nähdä tukevan hyvin nimenomaan startup-yritysten toimintaa, sillä se sisältää paljon startup-yrityksissä hyödyllisiä aspekteja. Scrum on yksi ensimmäisiä ketteriä menetelmiä ja se nojaa vahvasti Agile Manifeston (Beck et al. 2001) periaatteisiin (Dingsøyr et al. 2012). Startup-yritykset voisivat hyötyä myös tästä, sillä Scrumista on olemassa paljon tutkimustietoa ja se on laajasti käytössä teollisuudessa.

Scrum sisältää prosessin lisäksi ennalta määrättyjä rooleja, joita ovat Scrum master, tuotteen omistaja ja ohjelmistokehittäjät. Kaikkien roolien voidaan nähdä olevan tärkeitä Scrumin periaatteiden toteutumiseksi. Tarkastelemassani alueessa, eli 2-5 henkilön startup-yrityksissä tämä voi tarkoittaa sitä, että roolien tarkka soveltaminen yritykseen ei välttämättä ole täysin mahdollista. Scrum tiimien pitäisi olla kooltaan viidestä yhdeksään jäsentä (Schwaber & Beedle 2002 Abrahamsson et al. 2002 mukaan). Scrumin sisältämän prosessimallin toteuttaminen pienemmällä määrällä olisi myös vaikeaa, sillä ohjelmistokehittäjien pieni määrä vaikuttaa luonnollisesti negatiiviesti siihen, paljonko tiimin on mahdollista saada aikaiseksi. Sprinttien sisältämiä tehtäviä jouduttaisiin varmasti karsimaan ja projekti pitkittyisi verrattuna tilanteeseen, jossa Scrumin sisältämiä rooleja ei tarkasti noudatettaisi ja useampi tiimin jäsen osallistuisi itse ohjelmiston kehitykseen.

5.2 Extreme Programming

Luvussa 3.2 esitellyn Extreme Programmingin pääajatuksen voidaan nähdä olevan melko lailla sama kuin edellä käsitellyssä Scrumissa, sillä se on Scrumin tapaan syntynyt myös Agile Manifeston (Beck et al. 2001) perusperiaatteiden pohjalla samoihin aikoihin kuin Scrum. Toimintatapojen keskiössä ovat Scrumin tapaisesti ohjelman iteratiivinen suunnittelu, kehittäminen ja testaaminen. Sen sisältämät toimintatavat eivät ole mitenkään ainutlaatuisia, mutta yhdessä käytettynä ne luovat toimivan kokonaisuuden (Abrahamsson et al. 2002). Extreme Programmingin sisältämiä toimintatapoja ovat esimerkiksi pariohjelmointi, avoimet työtilat, yksikkötestien kirjoittaminen ennen koodia, lyhyet iteraatiot ja koodin uudelleenkäyttö (Wells 2013). Extreme Programming ei sisällä Scrumin tapaisesti tarkasti määriteltyjä rooleja, joka saattaa olla etu sen soveltamisessa startup-yrityksiin.

Extreme Programmingin toimintatapojen soveltuvuudesta startup-yrityksiin on olemassa melko vähän tutkimusta. Yau & Murphy (2013) esittävän artikkelissaan kysymyksen siitä, ovatko Extreme Programmingin ratkaisemat ongelmat relevantteja startup-yrityksissä. Tämä on heidän mukaansa myös yleisesti ketteriin ohjelmistokehitysmenetelmiin liittyvä kysymys, sillä monet ketteristä menetelmistä on alun perin kehitetty vastaamaan suuremmissa yrityksissä esiintyviä ongelmia, kuten kommunikointia tiimin sisällä. Kommunikointi alle viiden hengen startup-yrityksessä tuskin on kovinkaan oleellinen ongelma. Extreme Programmingin keskiössä olevan iteratiivisen toimintatavan edut ovat samat kuin Scrumin kohdalla, sillä sen avulla pystytään vastaamaan muuttuviin asiakasvaatimuksiin (Abrahamsson et al. 2002).

Toisin kuin Scrumissa, Extreme Programmingissa ei ole tarkkaa prosessia, jota yrityksen tulisi projektissaan seurata, vaan keskiössä ovat juurikin hyväksi todetut käytännön toimintatavat (Beck et al. 1999 Abrahamsson et al. 2002 mukaan). Yau & Murphy (2013) määrittelevät Extreme Programmingin suurimmaksi eduksi yksikkötestien kirjoittamisen ennen ohjelmointia ja tämän toimintatavan mukanaan tuoman laadun lisäyksen. He kuitenkin esittävät vastaväitteen siitä, vähentääkö yksikkötestien kirjoittaminen etukäteen startup-yritysten mahdollisuutta vastata muuttuviin asiakasvaatimuksiin. Joustavuus resurssien, asiakasvaatimusten, henkilöstön ja aikataulujen suhteen on Suttonin (2000) mukaan tärkeää varsinkin startup-yrityksille. Pariohjelmointi on myös tärkeä osa Extreme Programmingin toimintatapoja ja sen soveltuvuus startup-yrityksiin on kyseenalaista. Varsinkin tarkasteltavissa alle viiden henkilön yrityksissä resurssit ovat usein hyvin minimissä, mikä saattaa johtaa siihen, että yrityksen jäsenet joutuvat kaikki työskentelemään ohjelmiston eri alueiden parissa (Yau & Murphy 2013)

Yau & Murphy (2013) esittävät artikkelissaan, että vaikkakin Extreme Programming on hyvä projektinhallinnan työkalu, eivät sen ratkaisemat ongelmat välttämättä ole relevantteja pienissä startup-yrityksissä. Varsinkin tarkasti seurattuna Extreme Programming saattaa lähinnä hidastaa startup-yritysten ohjelmistokehitystä. Näistä syystä Extreme

Programmingin käyttöä tulee mielestäni harkita tarkkaan, ja mahdollisuuksien mukaan käyttää Coleman & O'Connor (2008) esittelemää menetelmän räätälöintiä tilanteen mukaan. Extreme Programmingin sisältämien yksittäisten toimintatapojen, kuten iteratiivisen ohjelmistokehityksen, edut ovat merkittävämpiä kuin koko ketterän menetelmän soveltaminen startup-yritykseen sellaisenaan. Extreme Programmingin toimintatapojen räätälöinnistä tilanteen mukaan on myös näyttöä teollisuudesta (Coleman & O'Connor 2008).

5.3 Dynamic Systems Development Method

Dynamic Systems Development Methodin (DSDM) filosofia on Agile Business Consertiumin (2014) mukaan "paras kaupallinen arvo syntyy, kun projekteilla on selvät kaupalliset tavoitteet, nopea toimitustahti ja kun motivoidut ihmiset työskentelevät yhdessä". Se perustuu yhdeksään periaatteeseen, joiden pohjana on maalaisjärjen ja pragmatismin käyttö. Kuten edellä esitellyissä Scrumissa ja Extreme Programmingissa, myös DSDM:n yksi pääasiallisista toimintatavoista on projektin jakaminen pieniin osiin ja iteratiiviseen ohjelmistokehitykseen. (Agile Business Consertium 2014) Abrahamsson et al. (2002) mukaan DSDM eroaa Scrumista ja Extreme Programmingista siinä, että se kattaa kaikki projektin vaiheet. DSDM on siis luonteeltaan raskaampi kuin Scrum ja Extreme Programming, joka voi aiheuttaa vaikeuksia sen soveltamisessa rakenteeltaan jokseenkin kevyisiin ohjelmistoalan startup-yrityksiin.

Dynamic Systems Development Methodin esittelemän prosessin sopivuutta voidaan tarkastella monelta kannalta. Se tuo muiden ketterien menetelmien tapaisesti etuja iteratiivisuudellaan, mutta saattaa olla tarkasti seurattuna raskas ylläpitää. Tämä voi aiheuttaa ongelmia varsinkin pienissä startup-yrityksissä, sillä resurssien ollessa vähissä prosessin vaiheiden seuraaminen tarkasti ei välttämättä ole mahdollista. DSDM:n prosessin sisältämät kattavat selvitykset ennen projektin aloitusta eivät juurikaan tue startup-yritysten pyrkimystä joustavuuteen, sillä niihin saattaa kulua paljonkin aikaa ja resursseja ennen kuin varsinaista ohjelmistoa aletaan edes kehittämään. Toisaalta, Yau & Murphy (2013) mukaan ketterät menetelmät tarkasti seurattuna parantavat valmiin tuotteen laatua. Huonona puolena on heidänkin mukaansa kasvaneelle laadulle syntyvät kustannukset ketterän menetelmän ylläpidon muodossa. Suuremmissa yrityksissä ketterän menetelmän aiheuttamat kulut voidaan Yau & Murphy (2013) mukaan jakaa tasaisemmin kuin startup-yrityksissä, jolloin ketterän menetelmän käytölle on enemmän perusteita. Coleman & O'Connor (2008) esittelemä ketterän menetelmän räätälöinti tilanteen mukaan voisi olla DSDM:n kohdalla mahdollista, sillä se sisältää hyväksi todettuja toimintatapoja monille eri projektin alueille (Abrahamsson et al. 2002).

Edellä käsitellyistä Scrumista ja Extreme Programmingista DSDM eroaa selvimmin siinä, että ennalta määriteltyjä rooleja on huomattavasti enemmän (Agile Business Consertium 2014). Abrahamsson et al. (2002) mukaan DSDM tiimien koko voi kuitenkin vaihdella kahdesta kuuteen henkilöön. Tämä tarkoittaa käytännössä sitä, että DSDM:n rooleja

voidaan yhdistellä samoille ihmiselle toisin kuin esimerkiksi Scrumissa (Mishra & Mishra 2011). Abrahamsson et al. (2002) mukaan DSDM:a on käytetty onnistuneesti sekä pienissä että suurissa projekteissa, joten eri roolien osalta sen soveltaminen ohjelmistoalan startup-yrityksiin olisi varmasti mahdollista.

5.4 Feature-driven Development

Feature-driven Development on Scrumin ja Extremen Programmingin tapaan Agile Manifeston (Beck et al. 2001) ydinajatusten pohjalta syntynyt ketterä ohjelmistokehitysmenetelmä, joka nimensä mukaisesti keskittyy ohjelmiston kehittämiseen asiakkaan arvostamien ominaisuuksien pohjalta (Nawaz et al. 2017). FDD sisältää muiden edellä käsiteltyjen ketterien menetelmien tapaan iteratiivisuutta, joka tuo startup-yritykselle kilpailuetua joustavuuden muodossa (Yau & Murphy 2013). FDD perustuu viiteen prosessiin sekä toimintatapoihin ja tekniikoihin, joiden tehtävänä on painottaa asiakkaan arvostamien ominaisuuksien kehittämistä. Prosessin osista ominaisuuksien suunnittelu ja toteutus ovat iteratiivisia. Ennalta määrätyn prosessin lisäksi FDD sisältää Scrumin ja DSDM:n kaltaisesti rooleja. (Palmer & Fleshing 2001, Abrahamsson et al. 2002 mukaan) FDD:n suurin ero verrattuna esimerkiksi Dynamic Systems Development Methodiin on se, että FDD kattaa pääasiallisesti vain ohjelmiston suunnittelun ja toteutuksen, kun taas DSDM sisältää myös projektinhallinnallisia aspekteja (Nawaz et al. 2017). Muista ketteristä menetelmistä FDD erottuu myös sillä, että sen avulla voidaan kehittää kriittisiä ohjelmistoja. (Abrahamsson et al. 2002)

Feature-driven Developmentin sisältämä viisiosainen prosessi on melko lineaarinen, sillä iteraatiota tapahtuu vain prosessin kahdessa viimeisessä osassa, eli ominaisuuksien suunnittelussa ja kehittämisessä. Projektin toisena vaiheena oleva ominaisuuslistan laatiminen saattaa sovellettuna startup-yrityksiin vähentää niiden joustavuutta vastata muuttuviin asiakasvaatimuksiin. FDD:n voidaan nähdä osaltaan sisältävän ketterien menetelmien ydinajatusten lisäksi osia myös perinteisemmästä, vesiputousmallin tapaisesta lähestymistavasta ohjelmistoprojekteihin, sillä suunnittelua ennen projektin aloitusta tapahtuu muihin ketteriin menetelmiin verrattuna huomattavasti enemmän. Yau & Murphy (2013) pitävät ketterien menetelmien etuna juuri sitä, että suunnittelua tapahtuu etukäteen melko vähän, joka lisää startup-yritysten mahdollisuutta vastata muuttuviin asiakasvaatimuksiin tai tavoitteisiin. Myös Nawaz et al. (2017) tunnistavat FDD:n prosessimallin asettamat rajoitteet liittyen projektin joustavuuteen ja soveltuvuuteen pieniin tai keskisuuriin projekteihin.

FDD sisältää 14 erilaista roolia, jotka jaetaan tärkeyden mukaan eri luokkiin. Toisin kuin esimerkiksi Scrum, FDD mahdollistaa roolien jakamisen eri ihmisten kesken tai useamman roolin asettamisen yhdelle ihmiselle. (Palmer & Felsing 2001, Abrahamsson et al. 2002 mukaan) Pienissä, alle viiden hengen startup-yrityksissä tämä tarkoittaa kuitenkin sitä, että yhdelle ihmiselle saattaa tulla melko monta roolia, joka saattaa vaikeuttaa FDD:n periaatteiden seuraamista. FDD sisältää myös rooleja kuten projektipäällikkö,

ohjelmistokehityspäällikkö ja lakimies, jotka eivät välttämättä ole täysin relevantteja pienessä startup-yrityksessä. Roolien lisäksi myös FDD: sisältämät toimintatavat eivät juurikaan ole suunniteltu käytettäviksi pienissä, alle viiden hengen startup-yrityksissä, sillä toimintatapojen mukaan jokaiselle kehitettävälle ominaisuudelle pitäisi olla osoitettu oma tiiminsä. (Palmer & Felsing 2001, Abrahamsson et al. 2002 mukaan)

Coleman & O'Connor (2008) esittämää menetelmän räätälöintiä ei voida Misra & Misra (2011) mukaan tehdä FDD:n kohdalla, sillä heidän mukaansa kaikkia FDD:n sisältämiä prosesseja ja toimintatapoja tulisi noudattaa yhdessä. Tämä saattaa osaltaan vähentää mahdollisuutta soveltaa sitä startup-yrityksiin. FDD:n soveltamisesta startup-yrityksiin on olemassa melko vähän tutkimustietoa, mutta edellä mainittujen seikkojen pohjalta soveltamisen voidaan nähdä olevan haastavaa.

5.5 Yhteenveto

Ketteristä menetelmistä Scrumin todettiin sisältämänsä prosessin puolesta olevan hyvin soveltuva startup-yrityksiin, mutta sen sisältämät ennalta määrätyt roolit saattavat tuottaa haasteita 2-5 henkilön startup-yrityksissä, sillä rooleja ei välttämättä voida noudattaa kovinkaan tarkasti. Mikäli Scrumin sisältämiin rooleihin liittyvät haasteet saadaan ratkaistua, sopii Scrum hyvin startup-yritysten ohjelmistokehitysmenetelmäksi, sillä se on hyvin tunnettu ja tukee startup-yritysten tavoitetta joustavuuteen.

Extreme Programmingin todettiin sopivan startup-yrityksiin lähinnä räätälöitynä, sillä useat sen sisältämät toimintatavat on suunniteltu suuremmille organisaatioille. Toimintatavat itsessään ovat hyödyllisiä, mutta ongelmia, joita niiden avulla pyritään ratkaisemaan eivät ole juurikaan relevantteja 2-5 henkilön startup-yrityksissä. Extreme Programmingista ei iteratiivisuutta lukuun ottamatta löydetty samantapaisia, startup-yritysten joustavuutta edistäviä ominaisuuksia kuten Scrumista. Extreme Programmingin yksittäisistä toimintatavoista voi kuitenkin olla hyötyä startup-yrityksille. Roolien puolesta Extreme Programming sopii startup-yrityksiin myös hyvin, sillä roolit eivät ole niin tarkasti määriteltyjä kuin esimerkiksi Scrumissa.

Dynamic Systems Development Methodin todettiin niin ikään sopivan startup-yrityksiin lähinnä räätälöitynä, sillä sen melko raskas rakenne saattaa olla startup-yrityksille liian kallis ylläpitää. Raskas rakenne saattaa myös vaikuttaa startup-yrityksen joustavuuteen negatiivisesti, ja varsinkin DSDM:n sisältämät kattavat esiselvitykset ovat startup-yritysten kilpailuvaltin, eli joustavuuden vastaisia. Toisaalta, DSDM:n menestyksekkäästä soveltamisesta myös pieniin yrityksiin ja projekteihin on näyttöä, ja sen sisältämät roolit ovat hyvin joustavia ja yhdisteltävissä.

Feature-driven Developmentin todettiin asettavan rajoitteita startup-yritysten pyrkimykselle joustavuuteen, sillä sen prosessi sisältää melko paljon suunnittelua ennen varsinaista toteutusta. FDD:n sisältämien roolien voidaan sanoa sopivan pieniin startup-yrityksiin

melko hyvin, sillä ne ovat hyvin joustavia ja yksi ihminen voi toimia useammassa roolissa. FDD:n kohdalla rajoitteita asettaa kuitenkin se, että se ei ole räätälöitävissä.

6. PÄÄTELMÄT

Tässä kandidaatintyössä käsiteltiin ketterien ohjelmistokehitysmenetelmien soveltuvuutta startup-yrityksiin. Ensin esiteltiin ketteriä menetelmiä yleisesti, jonka jälkeen esiteltiin tarkemmin neljä ketterää menetelmää, Scrum, Extreme Programming, Dynamic Systems Development Method ja Feature-driven Development. Jokaisesta käsitellystä ketterästä menetelmästä tuotiin esille ominaisuuksia, jotka saattavat vaikuttaa menetelmän soveltamiseen startup-yrityksiin. Menetelmien käsittelyn jälkeen esiteltiin ohjelmistoalan startup-yrityksiä, niiden ominaispiirteitä ja ohjelmistokehityksessä käytettäviä menetelmiä ja toimintatapoja nykyään. Viidennessä luvussa tarkasteltiin ketterien menetelmien sopivuutta startup-yritykseen yleisellä tasolla, sekä jokaisen eri käsitellyn menetelmän sopivuutta tarkemmin.

Työssä löydettiin useita ketterien menetelmien piirteitä, jotka vaikuttavat niiden soveltamiseen startup-yrityksiin. Löydöksiä tehtiin myös startup-yritysten luonteesta verrattuna esimerkiksi suuriin yrityksiin, ja siitä miten nämä erilaisuudet vaikuttavat ketterien menetelmien soveltamiseen startup-yrityksissä. Käsitellyistä neljästä ketterästä menetelmästä löydettiin ominaisuuksia, jotka sopivat startup-yrityksiin, mutta myös ominaisuuksia, jotka eivät sovi. Ketterät ohjelmistokehitysmenetelmät yleisellä tasolla todettiin soveltuvaksi ohjelmistoalan startup-yrityksiin, sillä ne tukevat startup-yritysten pyrkimystä joustavuuteen. Menetelmistä käsiteltiin sekä niiden sisältämää prosessia että menetelmien sisältämien määriteltyjen roolien soveltuvuutta startup-yrityksiin.

Tutkimuksessa löydettyjä tuloksia voidaan käyttää startup-yritysten ohjelmistokehitysmenetelmien kehittämisessä. Startup-yritykset noudattavat vain harvoin tarkasti tiettyä ketterää menetelmää ja useimmin käytössä onkin yhdistelmä hyväksi havaituista toimintatavoista. Tutkimuksessa löydettiin useita startup-yrityksiin hyvin soveltuvia toimintatapoja eri ketterien menetelmien sisältä, joten varteenotettavaa voisi olla näiden yhdistely startup-yritykselle sopivaksi kokonaisuudeksi.

6.1 Työn arviointi

Tutkimus antoi hyvän yleiskuvan ketteristä menetelmistä ja ohjelmistoalan startup-yrityksistä. Työssä saatiin hyvin esille startup-yritysten rajoittavia tekijöitä ja valittujen ketterien menetelmien soveltuvuutta startup-yrityksiin. Työn otsikkoa suoraan vastaavia tutkimuksia löytyi hyvin rajatusti, joten tietoa päädyttiin yhdistelemään useista lähteistä ja myös päätelmiä tehtiin työn loppupuolella melko paljon. Työssä saavutetut lopputulokset ovat melko odotettuja, mutta varsinkin ketterien menetelmien räätälöinti nousi ennalta arvioitua suurempaan rooliin niiden soveltamisessa startup-yrityksiin

6.2 Jatkotutkimusideat

Startup-yritysten nykyisin käyttämistä menetelmistä ja toimintatavoista löytyy melko paljon tutkimusta, mutta vähemmälle huomiolle on jäänyt juuri ketterien menetelmien soveltuvuus pieniin, 2-5 henkilön startup-yrityksiin. Myös eri ketterien menetelmien sisältämien toimintatapojen yhdistelystä on vain vähän tutkimusta. Yksi varteenotettava aihe jatkotutkimukselle voisi siis olla esimerkiksi empiirinen tutkimus ketterien menetelmien käytöstä startup-yrityksissä nykyään. Tutkimuksen avulla saataisiin laajempi kuva alan nykytilasta.

Tutkimusta yksittäisten toimintatapojen soveltuvuudesta startup-yrityksiin löytyi niin ikään melko rajatusti. Yksittäisten toimintatapojen soveltuvuutta tutkimalla voitaisiin mahdollisesti pystyä rajaamaan tietty joukko toimintatapoja, jotka soveltuvat erityistesti startup-yrityksiin. Toimintatapojen tarkempi tutkiminen tukisi myös käytännön työtä ohjelmistoalan startup-yrityksissä.

LÄHTEET

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2017). Agile software development methods: Review and analysis.

Agile Business Consortium (2014). DSDM Handbook. Saatavilla verkossa: https://www.agilebusiness.org/resources/dsdm-handbooks. Viitattu 1.3.2019.

Baseer, K.K., Reddy, M., Rama, A. & Bindu, C.S. (2015). A Systematic Survey on Waterfall Vs. Agile Vs. Lean Process Paradigms, i-Manager's Journal on Software Engineering, Vol. 9(3), pp. 34-59.

Beck, K. (1999). Extreme programming explained: embrace change, addison-wesley professional.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R. & Kern, J. (2001). Manifesto for agile software development. Saatavilla verkossa: https://agilemanifesto.org/iso/fi/manifesto.html. Viitattu 1.3.2019.

Coleman, G. & O'Connor, R.V. (2008). An investigation into software development process formation in software start-ups, Journal of Enterprise Information Management, Vol. 21(6), pp. 633-648.

Dahlstedt, A.G., Karlsson, L., Persson, A., NattochDag, J. & Regnell, B. (2003). Market-driven requirements engineering processes for software products–a report on current practices.

Dingsøyr, T., Nerur, S., Balijepally, V. & Moe, N. (2012). A decade of agile methodologies: Towards explaining agile software development . in: Journal of Systems and Software, pp. 1213-1221.

Greer, D. & Hamon, Y. (2011). Agile Software Development. Softw: Pract. Exper., 41: 943-944.

Dybå, T. & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review (2008). in: Information and Software Technology, pp. 833-859.

Erickson, J., Lyytinen, K. & Siau, K. (2005). Agile Modeling, Agile Software Development, and Extreme Programming: The State of Research, Journal of Database Management, Vol. 16(4), pp. 88-100.

Fagerholm, F., Ikonen, M., Kettunen, P., Münch, J., Roto, V. & Abrahamsson, P. (2015). Performance Alignment Work: How software developers experience the continuous adaptation of team performance in Lean and Agile environments, Information and Software Technology, Vol. 64 pp. 132-147.

Klotins, E., Unterkalmsteiner, M. & Gorschek, T. (2018). Software engineering in start-up companies: an analysis of 88 experience reports, Empirical Software Engineering, pp. 1-35.

Mishra, D. & Mishra, A. (2011). Complex software project development: agile methods adoption, Journal of Software Maintenance and Evolution: Research and Practice, Vol. 23(8), pp. 549-564.

Misra, S., Kumar, V. & Kumar, U. (2009). Identifying some important success factors in adopting agile software development practices. in: Journal of Systems and Software, pp. 1869-1890.

Misra, S. (2012). Agile software development practices: evolution, principles, and criticisms, The International Journal of Quality & Reliability Management, Vol. 29(9), pp. 972-980.

Nawaz, Z., Shabib Aftab & Anwer, F. (2017). Simplified FDD Process Model, International Journal of Modern Education and Computer Science, Vol. 9(9), pp. 53.

Palmer, S.R. & Felsing, M. (2001). A practical guide to feature-driven development, Pearson Education.

Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T. and Abrahamsson, P. (2014). Software development in startup companies: A systematic mapping study. Information and Software Technology, pp. 1200-1218.

Petersen, K., Wohlin, C. & Baca, D. (2009). The waterfall model in large-scale development, Springer, pp. 386-400.

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

Powers, S. (2014). Scrum – An Overview. Saatavilla verkossa: https://www.adventures-withagile.com/2014/06/10/scrum-an-overview/. Viitattu 11.3.2019.

Richardson, I. & von Wangenheim, C.G. (2007). Guest Editors' Introduction: Why are Small Software Organizations Different? IEEE Software, Vol. 24(1), pp. 18-22.

Royce, W.W. (1987). Managing the development of large software systems: concepts and techniques, IEEE Computer Society Press, pp. 328-338.

Schwaber, K. & Beedle, M. (2002). Agile software development with Scrum. Upper Saddle River: Prentice Hall.

Schwaber, K. & Sutherland, J. (2013). The scrum guide-the definitive guide to scrum: The rules of the game.

Shama, P.S. & Shivamanth, A. (2015). A Review of Agile Software Development Methodologies, International Journal of Advanced Studies in Computers, Science and Engineering, Vol. 4(11), pp. 1-6.

Stapleton, J. (1997). DSDM, dynamic systems development method: the method in practice, Cambridge University Press.

Sutton, S.M. (2000). The role of process in a software start-up, IEEE Software, Vol. 17(4), pp. 33-39.

Turk, D., France, R. & Rumpe, B. (2014). Limitations of agile software processes.

Vuorinen, J. (2011). Scrum-menetelmän käyttö Pirkanmaalaisissa ohjelmistoyrityksissä. Diplomityö. Tampereen teknillinen yliopisto.

Wells, D., (2001). Extreme Programming: A gentle introduction. www.extremeprogramming.org. Viitattu 5.3.2019.

Yau, A. & Murphy, C. (2013). Is a Rigorous Agile Methodology the Best Development Strategy for Small Scale Tech Startups?