Ella-Maria Mikkola

KETTERÄT MENETELMÄT GLOBAALISSA OHJELMISTOKEHITYKSESSÄ

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2017

TIIVISTELMÄ

Mikkola, Ella-Maria
Ketterät menetelmät globaalissa ohjelmistokehityksessä
Jyväskylä: Jyväskylän yliopisto, 2017, 30 s.
Tietojärjestelmätiede, kandidaatin tutkielma
Ohjaaja(t): Seppänen, Ville

Globaali ohjelmistokehitys on lisääntynyt 2000-luvulla huomattavasti kaiken muunkin tuotannon globalisoituessa. Globaalilla ohjelmistokehityksellä tavoitellaan esimerkiksi kustannusten laskua ja mahdollisuutta päästä käsiksi isompaan määrään resursseja. Kehitystyön hajautuminen maantieteellisesti tuo kuitenkin mukanaan myös erilaisia haasteita liittyen kommunikointiin, koordinointiin sekä kontrollointiin. Toinen ohjelmistokehityksen nouseva trendi on ketterät menetelmät. Ne ovat saaneet alkunsa 1990-luvulla ja korostavat kommunikointia, yhteistyötä, tuotekeskeisyyttä ja muutokseen vastaamista. Ketteriä menetelmiä käytetään myös globaalisti hajautuneissa ohjelmistoprojekteissa ja tässä tutkimuksessa etsitäänkin hyötyjä, joita ketterät menetelmät tarjoavat globaalille ohjelmistokehitykselle. Tutkimus on toteutettu systemaattisena kirjallisuuskatsauksena ja se vastaa kahteen tutkimuskysymykseen: 1) Miten ketteriä menetelmiä sovelletaan globaalissa ohjelmistokehityksessä? ja 2) Miten nämä sovellusmenetelmät tukevat ketterien menetelmien käyttöä globaaleissa ohjelmistoprojekteissa? Tutkimuksesta selviää, että ketterät menetelmät tarjoavat ratkaisuita kaikkiin kolmeen globaalin ohjelmistokehityksen ongelmaan ja että ketteriä menetelmiä on hyödyllistä käyttää globaalisti hajautuneissa projekteissa.

Asiasanat: ketterät menetelmät, globaali ohjelmistokehitys, hajautunut kehittäminen

ABSTRACT

Mikkola, Ella-Maria
Agile Methods in Global Software Development
Jyväskylä: University of Jyväskylä, 2017, 30 p.
Information Systems, Bachelor's Thesis
Supervisor(s): Seppänen, Ville

Global software development has increased during 21th century while all other industries have also globalized. With global software development companies try to reach lower costs and access to bigger resource pool. Geographically distributed development work brings also different kind of challenges regarding communication, coordination and control. Another emerging trend in software development industry is agile methods. They have originated in 1990 decade and the basic idea of agile methods is to emphasize communication, collaboration, importance of the product and responding to the change. Agile methods are used also in globally distributed software projects and the meaning of this study is to find out the benefits of usage of agile methods in global software development. This study is a systematic literature review and it answers to two research questions: 1) How are agile methods applied in global software development and 2) How do these applied solutions support global software development? The study reveals that agile methods offer solutions to all three challenges of global software development and also that it is beneficial to use agile methods in globally distributed projects.

Keywords: agile methods, global software development, distributed agile

TAULUKOT

SISÄLLYS

1 JOHDANTO

Kaikkien markkinoiden sekä tuotannon globalisoituminen on johtanut myös ohjelmistotuotannon globalisoitumiseen (Herbsleb, 2007). Globalisaation kiihtyminen ja siitä seurannut ohjelmistokehityksen hajaantuminen maailmanlaajuisesti on luonut tarpeen ymmärtää kehittämisen ja johtamisen menettelytapoja, jotta globaalisti tapahtuva ohjelmistokehitys olisi onnistunutta. Ajan kuluessa metodit ovat kehittyneet ja jalostuneet ja uusia menetelmiä on syntynyt. (Damian & Moitra, 2006, 18). Ohjelmistokehitys onkin kasvavissa määrin useassa paikassa samanaikaisesti tapahtuvaa, monikulttuurista ja globaalisti hajautunutta (Herbsleb & Moitra, 2001, 17).

Globaalisti hajautunutta ohjelmistokehitystä kutsutaan englanniksi termeillä global software development tai global software engineering. Usein puhutaan myös termistä distributed software development. Tässä tutkielmassa käytetään kuitenkin käsitettä globaali ohjelmistokehitys.

Toinen jo pitkään kasvanut trendi ohjelmistokehityksen alalla on ketterät menetelmät (agile methods). Ketterät menetelmät ovat joukko kehitysmetodeja, joille on yhteistä muutokseen vastaaminen, vaiheittaisuus, kommunikoinnin merkityksen korostaminen sekä tuote edellä toimiminen. Ketterien menetelmien perusarvoja ja periaatteita on määritelty Ketterän kehittämisen julistuksessa (Beck ym., 2001).

Yhä useammin ketteriä menetelmiä käytetään myös globaalissa ohjelmistokehityksessä. Globaali ohjelmistokehitys tuo mukanaan paljon hyötyjä, kuten mahdollisuuden päästä käsiksi isoon määrään resursseja, näin saada innovatiivista työvoimaa sekä mahdollisuuden päästä lähelle asiakasta. Globalisoituminen tuo mukanaan myös haasteita liittyen kommunikointiin, koordinointiin sekä kontrollointiin. Usein ketterien menetelmien käytöllä halutaankin vastata näihin haasteisiin. Ketterän menetelmän käytön voidaan ajatella helpottavan esimerkiksi kommunikaatiovaikeuksia, sillä ketterissä menetelmissä tehokasta kommunikointia korostetaan erityisen paljon.

Tässä tutkielmassa tutkitaankin, miten ketterät menetelmät helpottavat globaalin ohjelmistokehityksen kolmea perusongelmaa: kommunikaatiota, koordinointia sekä kontrollointia. Tämän lisäksi tutkitaan, miten tämä helpotus käytännön tasolla tapahtuu. Tutkimuskysymykset ovat siis seuraavat:

1. Miten ketteriä menetelmiä sovelletaan globaalissa ohjelmistokehityksessä?

   ja

2. Miten nämä sovellusmenetelmät tukevat ketterien menetelmien käyttöä globaaleissa ohjelmistoprojekteissa?

Tässä tutkimuksessa selviää, että ketterät menetelmät tarjoavat ratkaisuita niin kommunikointi-, koordinointi- kuin kontrollointiongelmiin. Sähköisten viestimien kuten chat-palveluiden, sähköpostin sekä tele- ja videokonferenssityökalujen käytöllä ketterien menetelmien vaatima kommunikointi onnistuu ja globaalin ohjelmistokehityksen kommunikointiongelmat helpottuvat. Koordinointiongelmiin tärkeimpänä ratkaisuna on projektinhallintatyökalut, joiden avulla työtehtäviä voidaan jakaa tehokkaasti. Kontrollointia taas helpottaa itse ketterien menetelmien tarjoamat selkeät prosessit ja käytänteet, jotka varmistavat, että kaikki projektin jäsenet tietävät, mitä projektissa on tekeillä. Kontrollointikäytänteet saavat myös kaikki tiimiläiset tuntemaan itsensä tärkeiksi ja samanarvoisiksi.

Alla olevassa alaluvussa esitellään tarkemmin tämän tutkimuksen tutkimusmenetelmä, systemaattinen kirjallisuuskatsaus. Lisäksi alaluku kattaa tutkimuksen lähdekirjallisuuden hakemisprosessin sekä seulonnan, jolla varmistetaan, että lähdemateriaali on mahdollisimman laadukasta sekä relevanttia tutkimuksen sisällön kannalta.

Toinen luku käsittelee ketteriä menetelmiä. Luvussa esitellään ketteryyden käsite sekä kuvaillaan, miten ja millaisissa olosuhteissa ketterät menetelmät ovat syntyneet sekä avataan niiden ominaispiirteitä. Lisäksi kuvaillaan ketteriä ohjelmistoprojekteja sekä esitellään lyhyesti tunnetuimpia ketteriä menetelmiä.

Kolmannessa luvussa käsitellään globaalia ohjelmistokehitystä sekä sen käsitettä. Luku kuvaa seikat, jotka ovat ajaneet ohjelmistokehityksen globalisoitumista. Lisäksi luvussa kuvaillaan globalisoitumisesta aiheutuvia hyötyjä ja ongelmia. Tarkemmin syvennytään globaalin ohjelmistokehityksen kolmeen perusongelmaan kommunikointiin, koordinointiin sekä kontrollointiin, sillä seuraavassa luvussa ketterien menetelmien ominaisuuksilla etsitään ratkaisuita näihin kolmeen ongelmaan.

Neljäs luku on siis itse tutkimusluku, jossa tutkimus on toteutettu hyvin kvantitatiivisin periaattein sekä systemaattisesti. Luvussa kuvaillaan, miten ketterät menetelmät helpottavat edellä mainittuja globaalin ohjelmistokehityksen ongelmia. Lisäksi tutkitaan, mitä nämä helpotuskeinot ovat käytännössä. Jokaisen alaluvun lopussa luodaan synteesi aiheesta.

Viidennessä ja viimeisessä luvussa tehdään yhteenveto koko tutkimuksesta sekä sen tuloksista. Yhteenvedossa esitetään myös mahdollisia jatkotutkimustarpeita.

1.1 Tutkimusmenetelmä

Tässä tutkielmassa hyödynnetään Salmisen (2011) systemaattisen kirjallisuuskatsauksen mallia. Malli on valittu siksi, koska tutkielman tulokseksi halutaan saada synteesi, jossa on käytetty lähdemateriaalina aikaisempia tutkimuksia. Systemaattisuus ja sen huolellinen toteuttaminen takaavat sen, että tulokset voidaan yleistää ja aiheesta voidaan luoda kokonaiskuva. Mallin vaiheet on esitetty alla (Salminen, 2011 mukaillen Fink, 2005):

1. Tutkimuskysymysten asettaminen
2. Kirjallisuuden ja tietokantojen valinta
3. Hakutermien valinta: sanoja ja fraaseja. Ajatuksena löytää materiaalia, joka vastaa tutkimuskysymyksiä.
4. Hakutulosten seulonta esimerkiksi kielen ja aikajänteen mukaan
5. Metodologinen seulonta: artikkelien ja tutkimusten tieteellisen laadun arviointi → valikoidaan mukaan laadukkain materiaali
6. Itse katsauksen tekeminen. Standardoitu keräilymenetelmä vaaditaan.
7. Tulosten syntetisointi: tämänhetkisen tiedon raportointi, tutkimustarpeen osoittaminen, löydösten selittäminen, tutkimuksen laadun kuvaus (luotettavuuden ja tarkkuuden varmistaminen) → tulosten laadullinen synteesi

Lähdekirjallisuus on haettu ja seulottu seuraavin periaattein. Hakutietokannaksi on valittu vain Google Scholar, sillä se kattaa myös muut yleiset tietokannat (kuten IEEE ja ACM) ja tarjoaa viittausten määrän. Hakuterminä toimi ensisijaisesti ”agile methods in global software development”, ja tämän avulla löydettiin 40 aiheeseen sopivaa artikkelia tai kirjaa. Kuitenkin hakutuloksia läpikäydessä havaittiin, että termi ”distributed agile” löytyy usean aiheeseen soveltuvan lähdemateriaalin otsikosta tai tiivistelmäosiosta, joten uusi haku suoritettiin tällä hakutermillä ja sen avulla löydettiin vielä 61 artikkelia tai kirjaa lisää. Tässä vaiheessa lähdemateriaalin soveltuvuutta aiheeseen mitattiin vain otsikko- ja tiivistelmätasolla, joten mukaan pääsi materiaalia melko löyhin perustein.

Tämän jälkeen, lähdemateriaalilukeman ollessa 101:ssa, suoritettiin sisällöllinen ja laadullinen seulonta. Tutkimuksen tavoitteena on selvittää, miten ketterillä menetelmillä pystytään vastaamaan globaalin ohjelmistokehityksen haasteisiin, joten lähdemateriaali seulottiin läpi sen perusteella, vastasiko se globaalin ohjelmistokehityksen kommunikointi-, koordinointi- ja kontrollointivaikeuksiin ketterien menetelmien tarjoamalla avulla. Aineisto luokiteltiin sen perusteella, vastaako se näistä yhteen, kahteen vai kaikkiin ongelmiin.

Seuraavaksi seulottiin lähdemateriaalin laatu. Seulonnassa käytettiin seuraavia laatukriteereitä: 1) Tieteellisyys ja tutkimuksen rakenne, 2) Tutkimusmenetelmän kuvaus sekä 3) Objektiivisuus.

Valittuun lähdeaineistoon haluttiin ainoastaan tieteellisiä tutkimuksia, joten ensimmäinen kriteeri sulkee pois esimerkiksi aikakauslehtimäiset haastattelut, joissa haastatellaan alan ammattilaisia tietystä teemasta. Valitun kirjallisuuden tieteellinen pohja pitää perustua laadukkaaseen lähdeaineistoon sekä ra-

kenteen on oltava hyvä, jotta tutkimusten löydöksiin voidaan luottaa. Rakenteella tarkoitetaan siis sitä, että materiaali noudattaa tieteellisen tutkimuksen, kuten tapaustutkimuksen tai kirjallisuuskatsauksen rakennetta – yllä mainittu haastattelu ei tätä toteuta.

Valituissa tutkimuksissa pitää olla kuvattuna, miten tutkimus on toteutettu. Usein tälle on omistettu oma luku, jossa kuvataan tutkimusmetodi sekä esimerkiksi lähdemateriaalin seulonta tai haastattelujen toteutus. Joissakin tapauksissa tutkimusmenetelmän kuvaus löytyy johdannon yhteydestä. Tällä varmistetaan tutkimuksen laadukkuus ja se, että tutkimuksen tulokset voidaan tarvittaessa jäljittää lähdeaineistoon.

Viimeisenä tarkistettiin tutkimusten objektiivisuus. Useimmissa tapauksissa liian subjektiivinen lähdemateriaali kuitenkin karsiutui jo toisen kriteerin kohdalla. Liialla subjektiivisuudella tarkoitetaan tässä tapauksessa sitä, että tutkimuksen toteuttaja on esimerkiksi tuonut omia kokemuksiaan mukaan tekstiin tai tutkimustulokset eivät ole yleistettävissä. Tästä esimerkkinä tutkimukset, joissa globaalin ohjelmistokehityksen ongelmiin tarjotaan ratkaisuna yksittäistä kaupallista tuotetta kuten jotakin kommunikointityökalua.

Tyypillisesti systemaattisen kirjallisuuskatsauksen lähdekirjallisuutta voidaan seuloa myös julkaisuvuoden ja viittausmäärän mukaan. Tässä tutkimuksessa näin ei ole kuitenkaan järkevää tehdä, sillä uusia tutkimuksia on melko vähän ja niiden viittausmäärät ovat luonnollisesti pienempiä kuin vanhempien tutkimuksien. Mikäli materiaalia karsittaisiin julkaisuvuoden perusteella, vanhimmat ja eniten viitatuimmat (toisin sanottuna suosituimmat ja arvostetuimmat) julkaisut jäisivät pois synteesistä. Mikäli taas kirjallisuutta seulottaisiin viittausmäärän mukaan, jäisivät uusimmat tutkimukset pois, missä oletettavasti käsiteltäisiin esimerkiksi uusien teknologioiden tarjoamia ratkaisuita globaalin ohjelmistokehityksen ongelmiin. Vanhempien tutkimusten tarkastelussa on kiinnitetty kuitenkin erityistä huomiota sisältöön, ja sisällön paikkansapitävyyttä nykypäivään on arvioitu tarkasti.

Loppujen lopuksi kirjallisuuskatsaukseen päätyi 19 kappaletta tapaustutkimuksia, joiden pohjalta synteesiä alettiin muodostamaan. Tapaustutkimus valittiin tähän tutkimukseen soveltuvana lähdemateriaalityyppinä siksi, että niiden perusteella on helppo muodostaa kvantitatiivisia tuloksia, esimerkiksi ”yhdessätoista tutkimuksessa yhdeksässätoista käytettiin sähköpostia kommunikointivälineenä”. Yksi syy lähdeaineiston karsimiselle oli myös se, että muuten tutkielma olisi venynyt liian pitkäksi.

2 KETTERÄT KEHITYSMENETELMÄT

Tässä luvussa kuvaillaan, miten ja millaisissa olosuhteissa ketterät kehitysmenetelmät ovat syntyneet. Luku kuvaa myös ketterien menetelmien ominaispiirteitä ja sitä, millaisiin projekteihin ne soveltuvat. Lopuksi esitellään lyhyesti muutama suosituin menetelmä.

Highsmith (2002) määrittelee ketteryyden käsitteen siten, että ketteryys on kykyä luoda muutosta ja vastata siihen, jotta onnistuttaisiin olemaan voittoisia sekasortoisessa liiketoimintaympäristössä.

2.1 Historia ja lähtökohdat

Lindvall ym. (2002, 197–198) esittävät, että ketterät menetelmät ovat syntyneet Basilin ja Turnerin (1975) Iterative Enhancement -tekniikan pohjalta vastaamaan liiketoiminnan kysyntään liittyen muutoksen hallintaan. Highsmithin ja Cockburnin (2001a, 4) mukaan ketterien menetelmien juuret johtavat 1990-luvulle, jolloin projektitiimit käyttivät onnistuneesti Scrumin, Dynamic System Development -metodin sekä Adaptive Software Development -metodin ensimmäisiä versioita.

Ketterät menetelmät ovat syntyneet ajatuksesta, että kehitysprojekteja toteutetaan epävarmassa ympäristössä, jossa vaatimusten muutokset ovat ailahtelevia ja tuotteita pitää toimittaa nopeasti. Tällaisilla projekteilla saattaa olla selkeä tavoite, mutta yksittäiset vaatimukset ovat epävakaita ja ne kehittyvät matkan varrella asiakkaan ja projektitiimin mukana. Tämän kaltaiset projektit eivät taivu täsmällisiin Plan-Driven Development -menetelmiin, kuten esimerkiksi vesiputousmalliin. (Highsmith & Cockburn, 2001a, 4). Nämä suunnitelmalähtöiset metodit, joissa käytetään paljon aikaa vaatimusten dokumentointiin sekä arkkitehtuurin suunnitteluun, koetaan turhauttavina ja toisinaan myös mahdottomina toteuttaa (Highsmith, 2002).

Abrahamssonin, Warstan, Siposen & Ronkaisen (2003, 244) mukaan ketterät menetelmät ovat syntyneet, kun liiketoimintayhteisöillä on ollut tarve kevy-

empään ratkaisuun, joka tarjoaa aiempaa nopeampia ja ketterämpiä ohjelmiston kehitysprosesseja.

Ketterät menetelmät ovat siis syntyneet vastaamaan selkeään tarpeeseen. On havaittu, että perinteisillä metodeilla ei pystytä vastaamaan muutokseen halutulla tavalla, joten on kehitetty uusia metodeita, jotka ovat kevyempiä ja mahdollistavat kehittämisen epävarmassa ympäristössä. Ketterät menetelmät ovat vastanneet myös tarpeeseen toimittaa tuotteita nopeammin.

2.2 Ominaispiirteet

Highsmith ja Cockburn (2001a, 4) esittävät, että ketterät menetelmät ovat yhdistelmä empiirisiä ja ennalta määriteltyjä prosesseja. Empiiriset prosessit syntyvät toiminnan edetessä epävarmoissa olosuhteissa. Tällaisia kokemusperäisiä prosesseja ovat esimerkiksi lyhyet iteraatiot, jatkuva testaaminen, itseohjautuvat tiimit, jatkuva yhteistyö ja jatkuva uudelleensuunnittelu perustuen sen hetkiseen tilanteeseen. Ennalta määritellyt prosessit liittyvät esimerkiksi tiettyihin kehitysmenetelmiin, joihin kuuluu tarkkoja algoritmeja, tarkasti määriteltyjä työtehtäviä ja täsmällistä tulosten mittaamista ja arviointia.

Samoja piirteitä löytyy myös Lindvallin ym. (2002, 201) tutkimuksesta, jossa ketteriä menetelmiä kuvataan iteratiivisiksi, vähittäisiksi, itseohjautuviksi sekä kasvaviksi. Itseohjautuvuudella ei suinkaan tarkoiteta sitä, ettei ketteriä tiimejä johdettaisi ollenkaan, vaan sillä tarkoitetaan sitä, että tiimit pystyvät järjestäytymään uudelleen ja uudelleen muodostaen erilaisia kokoonpanoja vastaamaan haasteisiin, joita tiimi kohtaa (Highsmith & Cockburn, 2001b, 132).

Toisen Highsmithin ja Cockburnin (2001b, 131) artikkelin mukaan vallitseva idea ketterässä kehittämisessä on se, että tiimit kykenevät olemaan tehokkaampia muutokseen vastaamisessa, kun tiimit pystyvät vähentämään kustannuksia tehokkaan informaationvaihdon avulla sekä lyhentämään aikaa, joka kuluu päätöksenteosta toteuttamiseen. Tehokkaan informaationvaihdon takaa se, että tiimin jäsenet on sijoitettu toisiaan lähelle, kasvokkain tapahtuvaa kommunikaatiota suositaan ja tiimin sopuisuutta, yhteisöllisyyttä ja moraalia korostetaan. Päätöksenteon ja toiminnan välistä aikaa voidaan lyhentää sillä, että osaavat ammattilaiset otetaan osaksi tiimiä sekä sillä, että tiimi työskentelee inkrementaalisesti.

Näiden lisäksi ketterien menetelmien piirteitä on kuvailtu Ketterän ohjelmistokehityksen julistuksessa (Beck ym., 2001), jossa esitetään seuraavat arvot, joita halutaan korostaa:

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

Yllä esitettyjen arvojen lisäksi samassa julistuksessa on esitetty tarkempia periaatteita, joita ketterä ohjelmistokehitys noudattaa. Seuraavissa kappaleissa

on käytetty lähteenä Beckin ym. (2001) Ketterän ohjelmistokehityksen julistusta, mikäli toisin ei mainita.

Yksi tärkeimmistä ketterien menetelmien periaatteista on asiakastyytyväisyyden takaaminen jo aikaisessa vaiheessa toimitetun toimivan ohjelmiston avulla. Projektin edetessä tähän ohjelmistoon lisätään iteraatio kerrallaan uusia ominaisuuksia, jotka iteraation päätteeksi ovat toimivia ja näin luovat arvoa asiakkaalle jokaisessa kehitysvaiheessa. Toimiva ohjelmisto uusin ominaisuuksin toimitetaan säännöllisesti, esimerkiksi kahdesta viikosta kahteen kuukauteen kestävän aikajakson välein. Kehitysiteraation lyhyyttä halutaan kuitenkin korostaa, joten toivottavaa on, että se pidetään mahdollisimman tiiviinä.

Toinen merkittävä seikka ketterissä menetelmissä on, että ne hyväksyvät vaatimusten muutokset projektin myöhäisissäkin vaiheissa – vaatimusten muuttuminen on jopa toivottua, sillä vaatimukset kehittyvät projektin edetessä asiakkaan ja projektitiimin kanssa (Highsmith & Cockburn, 2001a, 4). Ketterässä ohjelmistokehityksessä on tavoitteena, että muutos osataan kääntää asiakkaan kilpailueduksi.

Kolmas mainittava periaate liittyy kommunikoinnin korostamiseen. On toivottavaa, että asiakas ja kehittäjät ovat päivittäin tekemisissä toistensa kanssa koko projektin ajan. Myös projektitiimin sisäistä viestintää korostetaan, ja tehokkain tapa vaihtaa informaatiota onkin kasvokkain tapahtuva kommunikointi. Myös yksilöiden motivaatiota arvostetaan, ja heille pitää taata toimiva työskentely-ympäristö ja tuki, sekä heidän kykyihinsä pitää uskoa. Kun projektissa on motivoituneita ja kyvykkäitä yksilöitä, yksi periaatteista, itseohjautuvuus onnistuu.

Neljäs periaate on ajatus siitä, että projekti viedään läpi tuote edellä. Toimiva ohjelmisto on tärkein edistymisen mittaus ja kehitystyötä tehdään ajatuksella, että käytettävän teknologian ja ohjelmiston suunnittelun on oltava erinomaista. Yllä mainittu itseohjautuvuus nähdään sen edellytyksenä, että ohjelmiston arkkitehtuuri, vaatimukset ja suunnittelu onnistuvat. Myös yksinkertaisuutta korostetaan, ja turhaa työtä tulee välttää.

2.3 Ketterät ohjelmistokehitysprojektit

Kuten yllä on todettu, ketterät menetelmät ovat syntyneet tietynlaisissa olosuhteissa, ja tästä johtuen ne soveltuvat tietynlaisiin projekteihin. Koska ketterillä menetelmillä halutaan vastata muutokseen, on oletettavaa, että menetelmiä käytetään projekteissa, joissa muutoksen mahdollisuus on suuri. Vakaissa ja hyvin ennustettavissa olevissa projekteissa kyseisten menetelmien käyttö ei ole välttämättä tarpeellista tai järkevää, ja suunnittelulähtöiset menetelmät, kuten vesiputousmalli, voivat soveltua paremmin tämänkaltaisiin projekteihin. Tutkimuksissa on kuitenkin löydetty tietynlaisia projektin piirteitä ja ominaisuuksia, jotka ovat usein esillä puhuttaessa ketteristä ohjelmistokehitysprojekteista. Tässä alaluvussa kuvaillaan näitä ominaisuuksia.

Yksi tärkeimmistä tekijöistä, kun mitataan ketterien menetelmien soveltuvuutta projektiin, on projektitiimin koko. Noin kahdentoista hengen tiimeistä

on paljon kokemusta, jonkin verran myös tiimeistä, joissa on noin 25 jäsentä. Tiimin koon kasvaessa yli kahteenkymmeneen jäseneen koordinoinnista voi kuitenkin syntyä ongelmia. Yksi esimerkki, jossa ketterä menetelmä on onnistuttu skaalauttamaan huomattavan suurelle projektille, on noin 800 hengen kokoinen kehitysorganisaatio, joka hyödyntää ”Scrum of Scrums”-menetelmää, jossa organisaatio on jakautunut pienempiin kehitysyksiköihin. (Lindvall ym., 2002, 201–202.).

Toinen merkitsevä tekijä on projektin jäsenet ja heidän taitonsa. On todettu, että kehitystiimissä tulisi olla noin 25-33 prosenttia asiantuntevia ja kokeneita jäseniä. Tällaisilla jäsenillä tarkoitetaan henkilöitä, joilla on käytettävästä teknologiasta aiempaa kokemusta, kokemusta vastaavanlaisten ohjelmistojen kehittämisestä sekä hyvät kommunikointi- ja ihmistuntemustaidot. On tärkeää muistaa, että kokemus on nimenomaan teknologiasta eikä ketterien menetelmien tuntemisesta. (Lindvall ym., 2002, 202.).

Highsmith ja Cockburn (2001b, 131–132) esittävät, että ketterä ohjelmistokehitys keskittyy kyvykkäisiin yksilöihin. Heidän mukaansa ketterät projektitiimit pitävät yksilöiden pätevyyttä kriittisenä menestystekijänä projektien onnistumisen kannalta. Yksilöiden taidoista ei ole kuitenkaan hyötyä, jos tiimin kommunikointi ei toimi, joten yksilön kyvykkyyksien kehittämisen lisäksi ketterissä projekteissa korostetaan myös yhteistyön määrän kasvattamista. Ketterät prosessit ovat ennen kaikkea suunniteltu hyödyntämään niin yksilön kuin koko kehitystiiminkin ainutlaatuisia vahvuuksia.

Lindvall ym. (2002, 203) ovat havainneet, että ketterät menetelmän tarvitsevat vähemmän perehdytystä kuin niin sanotut perinteiset menetelmät. Yksi tehokkain tapa oppia ketterän kehityksen kulttuuria on tutkia aiempien ketterien projektien menestystekijöitä, joista tärkeimpinä ovat kulttuuri, ihmiset ja kommunikointi. Ketteryyden sanotaan olevan organisaatiokulttuuriin liittyvä asia, ja jos kulttuuri ei ole vastaanottavainen, ketteryys ei onnistu. Ketterään kulttuuriin liittyy vahvasti vapaus itseohjautuvuuteen sekä tuki neuvotteluun. Kuten ylläkin on mainittu, osaavien ihmisten rooli ketterässä ohjelmistokehityksessä on suuri. Ajatellaan, että vähempi määrä ihmisiä riittää, mikäli he ovat kyvykkäitä yksilöitä. Jotta organisaatio voi toimia ketterästi, täytyy sen haluta hyödyntää nopeaa kasvokkain tapahtuvaa kommunikaatiota.

Riskien ja ongelmien tunnistaminen aikaisessa vaiheessa on myös yksi tärkeä seikka ketterissä projekteissa. On tutkittu, että päivittäiset tapaamiset, joiden luonne halutaan pitää hyvin avoimena, rohkaisevat tiimin jäseniä raportoimaan ongelmista aikaisessa vaiheessa. Tällöin riskeihin pystytään puuttumaan aikaisessa vaiheessa ja seuraukset voidaan minimoida. (Lindvall ym., 2002, 204.).

Dokumentaatio on yksi keskustelluimmista asioista ketterien menetelmien projekteissa. Yksi näkemys on, että dokumentointi on heikko tapa kommunikoida, mutta toisinaan se on pakollista, jotta kriittistä tietoa voidaan käyttää myös myöhemmin hyödyksi. Ketterien menetelmien ideana on se, että kommunikoinnin tulisi olla mahdollisimman tehokasta, ja dokumentointi nähdään usein viimeisenä vaihtoehtona. (Lindvall ym., 2002, 204.).

On siis tutkittu, että ketterät menetelmän eivät sovellu kaikenlaisiin kehitysprojekteihin, vaan ketteriltä projekteilta on löydetty yhteneväisiä piirteitä.

Tällaisia ovat projektitiimin koko sekä tiimin yksilöiden kyvykkyys ja kyvykkäiden yksilöiden määrä tiimissä. Lisäksi on tutkittu, että ketterät menetelmät tarvitsevat vähemmän perehdytystä, kuin tavanomaiset metodit, mutta on havaittu, että organisaation täytyy olla vastaanottavainen, jotta se voi onnistuneesti ottaa käyttöön ketterän menetelmän. Riskien tunnistaminen aikaisessa vaiheessa on myös yksi ketterien projektien ominaisuus kuten myös dokumentaation määrän pitäminen minimissä.

2.4 Tunnettuja ketteriä menetelmiä

Highsmith ja Cockburn (2001a, 6) esittävät, että keskeisimpiin ketteriin menetelmiin kuuluu Lean Development, Adaptive Software Development, Scrum, eXtreme Programming, Crystal–metodi, Feature Driven Development ja Dynamic System Development –metodi. Abrahamsson ym. (2003) mainitsevat edellä mainittujen lisäksi Agile Modeling –metodin, Internet-speed Development –metodin sekä Pragmatic Programming –metodin. He lisäksi esittävät, että Crystal–metodi kuuluu Crystal–perheeseen, johon kuuluu useampia eri metodeita.

Extreme Programming eli XP on kerännyt näistä edellä mainituista metodeista eniten huomiota. Siihen kuuluu neljä arvoa: yhteisöllisyys, yksinkertaisuus, palautteenanto ja rohkeus. (Highsmith & Cockburn, 2001a, 7.). Extreme Programming –metodin tunnusomaisia piirteitä ovat lyhyet iteraatiot, palaute, läheinen asiakasyhteistyö, välitön kommunikointi ja koordinointi, jatkuva refaktorointi, jatkuva testaus, yhteinen koodinomistajuus sekä pariohjelmointi (Abrahamsson ym., 2003).

Scrum on myös yksi suosittu ketterä menetelmä. Se koostuu määritellyistä aloitus- ja lopetusvaiheista sekä niiden väliin jäävistä sprinteistä, jotka ovat epävakaita, muuttuvia ja epälineaarisia. Sprintti on iteratiivinen, yleensä viikosta neljään viikkoon kestävä ajanjakso, jonka aikana toimitettavaan tuotteeseen lisätään uusia toiminnallisuuksia. Scrumille tyypillistä on muuttuva toimitettava tuote, joustava aikataulu, pienet tiimikoot, säännölliset katselmukset, yhteistyö ja oliokeskeisyys. (Schawaber, 1997, 10-17.).

3 GLOBAALI OHJELMISTOKEHITYS

Tässä luvussa määritellään globaalin ohjelmistokehityksen käsite sekä kuvaillaan seikkoja, jotka ovat johtaneet ohjelmistokehityksen globalisoitumiseen. Lopuksi esitellään globaalin ohjelmistokehityksen aiheuttamia hyötyjä ja haasteita.

Herbsleb ja Moitra (2001, 17) kuvaavat, että ohjelmistokehitys on kasvavissa määrin useassa paikassa samanaikaisesti tapahtuvaa, monikulttuurista ja globaalisti hajautunutta. Tämä siis yksinkertaisuudessaan kuvaa globaalin ohjelmistokehityksen käsitteen.

3.1 Globalisoitumisen tekijät

Herbsleb ja Moitra (2001, 16–17) esittävät, että ohjelmistokehitys on alkanut globalisoitua, koska organisaatiot haluavat tavoitella alhaisempia kustannuksia sekä saavuttaa isomman määrän osaamista etätiimien sekä ulkoistamisen avulla. He esittävät, että seuraavat tekijät ovat kiihdyttäneet tätä trendiä:

1. Tarve hyödyntää globaalia resurssijoukkoa, jotta harvinaisia resursseja, missä ikinä ne sijaitsevatkaan, voidaan hyödyntää onnistuneesti ja kilpailukykyisesti
2. Markkinoiden läheisyydessä toimimisen hyödyt, sisältäen asiakkaan ja paikallisten olojen tietämyksen
3. Virtuaalisten tiimien nopea muodostaminen, jotta voidaan hyödyntää markkinamahdollisuuksia
4. Tuotantoajan lyhentäminen hyödyntämällä aikavyöhykkeitä ja ympärivuorokautista kehitystyötä
5. Tarve joustavuuteen, jota voidaan hyödyntää mahdollisissa yritysfuusioissa ja –ostoissa

Damian ja Moitra (2006, 18) väittävät, että globaalista ohjelmistokehityksestä on tullut jo välttämättömyys liiketoiminnalle esimerkiksi rahallisista syistä, resurssien niukkuudesta johtuen ja tarpeesta sijoittaa kehittäjiä asiakasta lähelle.

Ebert ja De Neve (2001, 69) ovat taas sitä mieltä, että globaali ohjelmistokehitys ei ole tavoite itsessään, vaan se on tulos tietoisesta liiketoiminta-orientoituneesta vaihtokaupasta, jota ajaa ekonomiset syyt sekä mahdollisuus päästä käsiksi parhaisiin tutkimus- ja kehitysresursseihin ympäri maailman. Herbsleb (2007) on sitä mieltä, että yksi tekijä, joka aiheuttaa ohjelmistokehityksen globalisoitumista on kaikkien markkinoiden sekä tuotannon globalisoituminen – tämä kasvattaa paineita tuottaa ohjelmistoja myös maailmanlaajuisesti hajautetuneesti.

3.2 Globalisoitumisen ongelmat ja hyödyt

Alla olevassa taulukossa (TAULUKKO 1) esitetään tiivistetysti globaalin ohjelmistokehityksen hyödyt ja ongelmat. Taulukon jälkeen kuvaillaan merkittävämpiä ongelmia tarkemmin sekä kuvataan muutamia hyötyjä, jotka eivät taulukossa esiinny.

Kommunikoinnilla tarkoitetaan niin kasvokkain kun tietoteknisillä viestimilläkin tapahtuvaa informaation vaihtoa. Koordinointi kattaa esimerkiksi työtehtävien jakamisen ja kontrollilla tarkoitetaan muun muassa projektinhallinnallisia toimia ja johtamista ja sitä, että toiset saadaan toimimaan projektin kannalta parhaalla mahdollisella tavalla.

TAULUKKO 1 Globaalin ohjelmistokehityksen hyödyt ja haitat (Ågerfalk & Fitzgerald, 2006, 28).

	Ajallinen etäisyys	**Maantieteellinen etäisyys**	**Sosiokulttuurinen etäisyys**
Kommunikointi	+ Parantunut kommunikoinnin dokumentointi - Vähentyneet mahdollisuuden synkronoituun kommunikointiin	+ Läheisempi sijainti markkinoihin + Etäyhteys kyvykkääseen työvoimaan - Kasvokkain tapahtuvien tapaamisten vaikeus	+ Innovointi ja parhaiden käytänteiden jakaminen - Kulttuuriset väärinymmärrykset
Koordinointi	+ Koordinointitarpeet voi minimoida - Tyypillisesti koordinointikulut nousevat	+ Joustavampi koordinaatiosuunnittelu - Vähentynyt informaali kontakti voi johtaa kriittisten tehtävien huomioimattomuuteen	+ Parempi oppiminen ja isompi kirjo osaamista - Ristiriitaiset työtavat voivat hankaloittaa tehokasta koordinointia
Kontrollointi	+ Aikavyöhykkeiden tarjoama etu, 24/7 työskentely - Projektinhallinta voi aiheuttaa viivästymistä	+ Kommunikointikanavat mahdollistavat kommunikoinnin dokumentoinnin - Hankala välittää visioita ja strategiaa	+ Ennakointi luontaista joissakin kulttuureissa - Eri käsitykset vallankäytöstä voi horjuttaa moraalia - Johtajien sopeuduttava paikalliseen sääntelyyn

Ågerfalk ja Fitzgerald (2006, 28) toteavat, että suurimmat ongelmat aiheutuvat maantieteellisestä etäisyydestä. Se johtaa aikavyöhykkeiden vaikutuksesta ajalliseen etäisyyteen sekä eri kulttuurien vaikutuksesta sosiokulttuuriseen etäisyyteen. Kun kehittäjät eivät ole sijoitettuna fyysisesti samaan paikkaan, joutuvat he turvautumaan asynkronisiin kommunikaatiovälineisiin kuten sähköpostiin. Aikavyöhykkeiden aiheuttama työskentelyaikojen ero taas voi johtaa siihen, että ihmiset eivät ole toistensa saatavilla silloin, kun tarve siihen olisi. Erilaiset sosiokulttuuriset taustat taas aiheuttavat taas sen, että ihmisillä on erilaiset työ- ja kommunikointitavat sekä äidinkieli on usein myös eri, mitkä saattavat johtaa turhautumiseen ja väärinkäsityksiin.

Herbsleb (2007) esittää, että tietoisuuden puute on yksi merkittävimmistä ongelmien aiheuttajista. Kun kehittäjät eivät voi kommunikoida reaaliaikaisesti esimerkiksi kohdatuista ongelmista, saattavat jotkin asiat jäädä huomioonottamatta tai syntyä väärinkäsityksiä. Suurin tietoisuuden puutteen aiheuttama ongelma kuitenkin liittyy muutokseen vastaamiseen – kohdatuista muutoksista ei voida kommunikoida reaaliaikeisesti ja tämä syö tehokkuutta.

Toinen Herbslebin (2007) nostama ongelma liittyy ristiriitaisuuteen. Hajautetuilla kehitysyksiköillä saattaa olla erilaiset kehitystyökalut, prosessit ja organisaatiokulttuuri. Nämä saattavat johtaa erilaisiin ongelmiin kuten erilaisten prosessien aiheuttamaan hämmennykseen ja väärinkäsityksiin työtavoista ja sen hetkisestä tilanteesta ja edistymisestä. Tulee kuitenkin ottaa huomioon, että kyseinen artikkeli on kymmenen vuoden takaa, ja toivottavaa olisi, että tämänkaltaiset, jopa yksinkertaisesti ratkaistavissa olevat ongelmat on osattu eliminoida.

Herbsleb ja Moitra (2001, 17) nostavat esiin erään strategisen ongelman, jota muissa artikkeleissa ei ole mainittu. Kehitystyön hajauttaminen ja globalisointi voi aiheuttaa usein vastustusta organisaatiossa. Korkein johto ja keskijohto saattavat usein olla eri mieltä hajauttamisen tuottamista hyödyistä ja yksilöt saattavat kokea, että heidän työpaikkansa on uhattuna ja monet eivät halua matkustaa useasti. Mikäli kehittäjiä sijoitetaan esimerkiksi asiakkaan luo saattaa se vaatia heiltä ulkomaille muuttoa jopa useaksi vuodeksi.

Eri tutkimuksissa on löydetty myös edellä mainittujen hyötyjen lisäksi muitakin positiivista puolia, joita globaali ohjelmistokehitys mahdollistaa. Innovointi on edellä mainittujen lisäksi yksi esimerkki, jota globaali ohjelmistokehitys tarjoaa. Pääsy eri kulttuureista tuleviin resursseihin varmistaa, että tuotteita kehitetään innovatiivisesti ja myös projekteja johdetaan sekä prosesseja kehitetään innovatiivisella tavalla. (Ebert & De Neve, 2001, 68). Conchúir ym. (2009, 129) esittävät, että yksi merkittävimmistä hyödyistä on se, että globaalit tiimit voidaan jakaa pienempiin osiin, jotka voivat samanaikaisesti tehdä kehitystyötä, esimerkiksi työskennellä ohjelmiston eri ominaisuuksien parissa. Tämä mahdollistaa sen, että kehityssyklit ovat lyhyempiä eli toiminta on tehokkaampaa kuin silloin, kun kaikki työskentelisivät saman ominaisuuden parissa.

4 KETTERIEN MENETELMIEN TARJOAMAT RATKAISUT GLOBAALIN OHJELMISTOKEHITYKSEN ONGELMIIN

Kuten johdannossa esitettiin, kirjallisuuskatsaukseen päätyi loppujenlopuksi 19 tapaustutkimusta, joiden pohjalta tutkimus ja synteesi luodaan. Tapaustutkimukset koskevat joko yhtä tai useampaa yksittäistä tapausta, ja useimmiten tapauksia on tutkittu haastattelujen avulla. Tässä tutkielmassa useampaa tapausta koskevat tutkimukset on laskettu yhdeksi tapaukseksi eli niitä ei eritellä erillisiksi tapauksiksi tuloksissa. Näin on uskallettu tehdä esimerkiksi siksi, että tapaukset ovat koskeneet samaa ketterää kehitysmenetelmää (Paasivaara, Durasiewicz & Lassenius, 2009) tai tapauksilla on yhteneväisiä sovellusmenetelmiä (Ramesh, Cao, Mohan & Xu, 2006).

Tapaustutkimuksista on etsitty yhteneviä ketterien menetelmien sovellusmenetelmiä. Tässä tutkimuksessa sovellusmenetelmällä tarkoitetaan esimerkiksi jotakin työkalua, kuten kommunikointivälinettä, joka tukee ketterän menetelmän käyttöä tai mahdollistaa ketterän menetelmän käytön globaalissa ohjelmistoprojektissa, mutta ei ole alkuperäisesti osa ketterää prosessia.

Tutkimuksista löytyneet sovellusmenetelmät on jaettu kolmeen luokkaan liittyen kolmeen eri globaalin ohjelmistokehityksen ongelmaan, joita ovat kommunikointi-, koordinointi- ja kontrollointiongelmat. Tärkeää on huomioida se, että useat sovellusmenetelmät koskevat monesti kahta tai kaikkia kolmea ongelmaa samanaikaisesti, joten tämä luo hieman päällekkäisyyttä ja yhdessä tutkimuksessa löytynyt yksi sovellusmenetelmä voi esiintyä useassa ryhmässä. Myös ketterien menetelmien tuomat hyödyt globaalin ohjelmistokehityksen ongelmiin voivat koskea useampaa ongelmaa samanaikaisesti. Näistä esimerkkinä esimerkiksi verkossa toimiva kanban-taulu, joka toimii samalla niin kommunikointivälineenä kuin myös tehtävien jakamisessa (koordinointi) sekä myöskin projektin tulosten seurannassa (kontrollointi) (Razzak & Ahmed, 2014).

Tässä luvussa siis esitellään tutkimuksista tehdyt löydökset sekä luodaan synteesi niiden pohjalta. Jokaista globaalin ohjelmistokehityksen ongelmaa kohden on oma alaluku, joka esittää kyseisen teeman sovellusmenetelmät sekä niiden hyödyt kehitysprojekteille. Joissakin tutkimuksissa hyötyjä on kuvailtu

kattavalla tavalla, mutta osa tutkimuksista ei käsitä hyötyjä ollenkaan ja tätä varten on tärkeää luoda synteesi ja johtopäätöksiä sovellusmenetelmien pohjalta nojaten muihin tutkimuksiin. Alaluvuissa siis vastataan ensin ensimmäiseen tutkimuskysymykseen liittyen ketterien menetelmien soveltamiseen ja tämän jälkeen tarkastellaan toista tutkimuskysymystä eli sovellusmenetelmien tarjoamia hyötyjä ja tukea projekteille.

4.1 Kommunikointi

Kahdeksantoista tutkimusta yhdeksästätoista käsittelee kommunikointia. Tämä oletettavasti johtuu siitä, että ketterät menetelmät haluavat korostaa kommunikointia. Hajautetuissa tiimeissä kasvokkain tapahtuva kommunikointi on ongelmallista, joten sen tilalle on pitänyt keksiä muita vaihtoehtoisia tapoja. Tämä alaluku käsittää nämä kommunikointikeinot. Alla oleva taulukko (TAULUKKO 2) esittää tutkimuksista löydetyt kommunikointimenetelmät ja sen, kuinka monesta tutkimuksesta kyseinen menetelmä löytyi. Tämän jälkeen esitellään, mihin tarkoitukseen kyseistä menetelmää käytetään ja millaisia hyötyjä sen käytöstä on ollut. Luvun lopussa luodaan synteesi ja johtopäätökset kommunikoinnista.

TAULUKKO 2 Kommunikoinnin sovellusmenetelmät lähdemateriaalissa.

Kommunikointimenetelmä	**Esiintymät lähdemateriaalissa (kpl)**
Chat	12
Sähköposti	11
Projektinhallintajärjestelmä	10
Vierailu	10
Videokonferenssi	9
Telekonferenssi	9
Puhelu	7
Näytön jakaminen	5
Versiokontrolli	4
Wiki	4
Kontaktihenkilö	4
Keskustelufoorumi	1
Tekstiviesti	1

4.1.1 Kommunikoinnin sovellusmenetelmät

Chat eli reaaliajassa tapahtuva pikaviestiminen nousi suosituimmaksi kommunikaatiovälineeksi. Tähän kategoriaan on yhdistetty tutkimuksista esille nousseet chat-palvelut, Instant Message –palvelut sekä niin kahden henkilön välinen kuin ryhmächatkeskustelukin. Yhteistä näille kaikille on se, että keskustelu on reaaliaikaista ja esimerkiksi kysymyksiin saadaan heti vastaus. Monessa tutkimuksessa nousikin esiin, että chat-palveluita käytettiin nopeiden kysymysten tai pienien asioiden selvittämiseen (Korkala & Maurer, 2014; Layman, Williams,

Damian & Bures, 2006; Modi & Abbott & Counsell, 2013; Niinimäki, 2011; Paasivaara ym., 2009; Razzak & Ahmed, 2014; Ramesh, Mohan & Cao, 2012). Chat-palveluita käytettiin myös asioista informoimiseen (Niinimäki, 2011) tai tarkistamaan, onko henkilö paikalla puhelua varten (Korkala & Maurer, 2014). Yhdessä tutkimuksessa chat-palveluita (sekä sähköpostia) käytettiin myös palaverien pitämiseen, sillä tele- ja videokonferenssit oli koettu liian haastaviksi kielimuurin ja ajan järjestämisen takia (Hole & Moe, 2008).

Toiseksi suosituin kommunikointiväline oli sähköposti. Sähköpostia käytettiin muun muassa projektin statusraporttien jakamiseen (Korkala & Abrahamsson, 2007), ongelmien selvittämiseen (Layman ym., 2006; Lee & Yong, 2010) ja palaverien muistiinpanojen jakamiseen (Paasivaara, Lassenius & Heikkilä, 2012).

Projektinhallintajärjestelmät sijoittuvat myös vahvasti koordinointi- sekä kontrollointikategoriaan mutta ne toimivat myös viestimisvälineenä, sillä niiden kautta voi kommunikoida esimerkiksi tehtävien jakamisesta tai projektin edistymisestä. Projektinhallintajärjestelmät kattavat tutkimuksista löydetyt Scrum backlog –järjestelmät, kanban-taulut, eXtreme Programming –työkalut sekä muut elektroniset järjestelmät tai tietovarastot, joihin kaikilla tiimin jäsenillä on pääsy sekä joita käytetään projektinhallinnallisiin tehtäviin. Tämän kaltaisia järjestelmiä käytettiinkin eniten tehtävien jakamisesta kommunikointiin (Bannerman, Hossain & Jeffery, 2012; Blomkvist, Persson & Åberg, 2015; Layman ym., 2006; Modi ym., 2013; Paasivaara, Durasiewicz & Lassenius, 2008; Paasivaara ym., 2009; Persson, Mathiassen & Aaen, 2012; Pries-Heje & Pries-Heje, 2011; Razzak & Ahmed, 2014; Ramesh ym., 2006). Myös Wikiä käytettiin samoihin tarkoituksiin (Lee & Yong, 2010; Razzak & Ahmed, 2014).

Eri toimipisteiden välisiä vierailuita hyödynnettiin myös useassa tapauksessa. Vierailut kuuluvat enemmän koordinointi- sekä kontrollointiosioihin, mutta vierailut koettiin tehokkaaksi tavaksi kommunikoida monimutkaisista asioista (Paasivaara ym., 2009; Razzak & Ahmed, 2014; Ramesh ym., 2012). Vierailujen avulla haluttiin myös luoda eri toimipisteiden ihmisten välille yhteisöllisyyttä sekä tutustuttaa heitä toisiinsa (Bannerman ym., 2012; Modi ym., 2013; Pries-Heje & Pries-Heje, 2011).

Video- ja telekonferensseja käytettiin palavereiden pitämiseen (Bannerman ym., 2012; Korkala & Maurer, 2014; Lee & Yong, 2010; Niinimäki, 2011; Noordeloos, Manteli & Van Vliet, 2012; Paasivaara ym., 2008; Paasivaara ym., 2009; Paasivaara ym., 2012; Pries-Heje & Pries-Heje, 2011; Ramesh ym., 2012; Ramesh ym., 2006). Palavereissa suunniteltiin tulevaa, arvioitiin edistymistä ja ratkaistiin ongelmia. Joissakin tapauksissa video- ja telekonferenssien yhteydessä hyödynnettiin myös näytön jakamista (Pries-Heje & Pries-Heje, 2011; Razzak & Ahmed, 2014).

Palavereita ja muuta tiedon jakamista hoidettiin myös kännykkäpuhelujen avulla (Korkala & Maurer, 2014; Modi ym., 2013; Ramesh ym., 2012; Ramesh ym., 2006). Myös puhelun yhteydessä käytettiin hyödyksi näytön jakamista (Korkala & Maurer, 2014).

Versiokontrollijärjestelmään jaettiin tehty koodi, jossa se oli kaikkien nähtävillä ja saatavilla (Hole & Moe, 2008; Lee & Yong, 2010; Paasivaara ym., 2008; Razzak & Ahmed, 2014). Myös yhdessä tapauksessa käytettiin tekstiviestejä

pienien asioiden hoitamiseen (Ramesh ym., 2012) sekä yhdessä keskustelufoorumia teknisistä asioista keskusteluun (Razzak & Ahmed, 2014). Joissakin tapauksissa toimipisteiltä oli valittu yksi henkilö kommunikoinnin hoitamiseen, koska se koettiin tehokkaimpana tapana toimia (Hole & Moe, 2008; Moe ym., 2014; Paasivaara ym., 2012; Ramesh ym., 2012). Tämä ei kuitenkaan tue millään lailla ketterien menetelmien periaatteita.

4.1.2 Kommunikoinnin sovellusmenetelmien hyödyt

Sähköpostin käytön hyödyiksi nähtiin se, että se tarjosi ”mustaa valkoisella” –ratkaisun, josta voi aina tarvittaessa myöhemmin tarkistaa asioita (Korkala & Maurer, 2014). Sähköpostista koettiin myös hyötyä ongelmien ratkaisemisessa (Lee & Yong, 2010). Sähköpostin käyttö koettiin yhdessä tapauksessa helpompana, kuin tele- tai videokonferenssit, koska sähköpostin avulla päästiin eroon kielimuurin ongelmista (Hole & Moe, 2008). Sähköpostilla saatiin jaettua myös helposti tietoa kaikille tiimiläisille kerralla (Lee & Yong, 2010; Niinimäki, 2011).

Chat-palveluiden hyödyksi nähtiin se, että se luo läpinäkyvyyttä tiimiläisten välille sekä paljastaa ongelmat ajoissa, jotta niihin voidaan löytää myös ratkaisu ajoissa. Chatin käytön eduksi nimettiin myös se, että se luo yhteyttä ja suhteita. (Paasivaara ym., 2009.).

Vierailuista koettiin olevan yleisesti paljon hyötyä. Ne paransivat kommunikaatiota sekä suhteita tiimiläisten välillä (Razzak & Ahmed, 2014). Vierailujen avulla voitiin myös selvittää asioita, jotka olisi ollut vaikeaa selvittää sähköpostin tai chatin avulla (Paasivaara ym., 2009).

Projektinhallintatyökalut nähtiin tehokkaimpana tapana jakaa tietoa työtehtävistä, deadlineista ja työtunneista, koska ne olivat helposti kaikkien saatavilla (Razzak & Ahmed, 2014).

Päivittäisten palaverien, jotka toteutettiin tele- tai videokonferenssina, hyödyksi koettiin se, että ne paransivat kommunikaatiota ja rohkaisivat tiimiläisiä avoimeen ilmapiiriin, jossa ongelmista uskallettiin puhua ääneen ja tarpeeksi ajoissa (Noordeloos ym., 2012; Paasivaara ym., 2009).

Yksi merkittävä asia, joka kuitenkin nähdään enemmän koordinointi- ja kontrollointikeinona, on itse ketterän prosessin (usein Scrumin) tarjoamat hyödyt. Ketterä menetelmä koettiin hyödyllisenä, koska se tuki avointa ja säännöllistä kommunikointia sekä strukturoi kommunikointia (Paasivaara ym., 2009). Kommunikoinnin koettiin myös lisääntyneen ketterän menetelmän käytön myötä (Paasivaara ym., 2008).

4.1.3 Synteesi

Useassa tapauksessa säännöllisen kommunikoinnin koettiin tuovan luottamusta hajautettujen tiimien välille (Noordeloos ym., 2012; Pries-Heje & Pries-Heje, 2011; Ramesh ym., 2012; Ramesh ym., 2006; Razzak & Ahmed, 2014), mikä on hajautetuissa projekteissa erittäin tärkeää. Kun kaksi tiimiä on maantieteellisesi kaukana toisistaan, heidän välilleen ei välttämättä synny samanlaista luottamussuhdetta kuin samassa paikassa työskenteleville, koska he eivät pääse kommunikoimaan kasvokkain.

Kuten huomata saattaa, eri tapauksista ilmeni monia erilaisia kommunikointivälineitä, joita myös käytettiin eri tarkoituksiin. Olisikin hyvä, että tiimillä olisi käytettävissään tarpeeksi laaja joukko erilaisia kommunikointivälineitä, joista he itse voisivat valita itselleen ja tilanteeseen sopivan työkalun. Tiimiläisten kommunikointia ei tulisi rajoittaa millään tavalla, sillä ketterät menetelmät korostavat runsasta ja epävirallista (informaalia) kommunikointia. Tästä syystä yhden kontaktihenkilön valitseminen ei ole järkevää. Laajalla kommunikointityökalujen kirjolla sekä vapaalla kommunikoinnilla varmistettaisiin, että tiimin jäsenet kommunikoisivat keskenään tarpeeksi ja esimerkiksi ongelmat tulisivat näin ratkaistuiksi ajoissa.

Kommunikointivälineitä käytetään eri tarkoituksiin ja niillä on erilaisia hyviä puolia. Sähköpostia kannattaisi käyttää koko tiimiin kohdistuvaan tiedotukseen sekä asiakkaalle kommunikointiin, sillä se voidaan nähdä virallisempana väylänä kuten esimerkiksi chat. Chat-palveluita kannattaisi taas käyttää pienempien asioiden selvittämiseen, sillä niiden avulla asiaan saa heti selvyyden. Tele- ja videokonferenssityökaluja kannattaisi hyödyntää palaverien pitämiseen, tutustumiseen sekä suhteiden luomiseen, sillä niiden avulla on mahdollista keskustella lähes kuin kasvokkain.

Työtehtävistä kommunikoimiseen kannattaisi käyttää jotakin projektinhallintatyökalua kuten esimerkiksi jotakin Scrum backlog –järjestelmää, joka mahdollistaa kaikille tiimin jäsenille pääsyn tarkastelemaan työtehtäviä, edistymistä sekä deadlineja. Tämä on myös tehokasta, sillä kaikki tärkeä projektinhallinnallinen informaatio on saatavilla samasta paikasta, eikä tietoja tarvitse esimerkiksi etsiä sähköpostikansioista.

Myös vierailuja kannattaisi hyödyntää, vaikka ne ovatkin huomattavasti kalliimpia kuin muut kommunikointimenetelmät. Niiden hyödyksi koetaan luottamuksen ja suhteiden kasvattaminen sekä toisiin tutustuminen, ja näitä ei välttämättä voi hoitaa elektronisten viestintävälineiden avulla – ainakaan samalla tasolla.

Johtopäätöksenä voidaan sanoa, että ketterän menetelmä käyttö tukee kommunikointia ja globaalin ohjelmistoprojektin on hyödyllistä käyttää jotakin ketterää menetelmää, sillä ne tarjoavat mahdollisuuden säännölliseen, useimmiten päivittäiseen kommunikointiin, joka helpottaa muun muassa ongelmien ratkaisemisessa. Projektitiimillä tulisi olla käytettävissään laaja joukko kommunikointivälineitä, jotta kommunikoinnista tehtäisiin mahdollisimman vaivatonta ja helppoa. Täytyy kuitenkin muistaa, että informaation jakaminen ei tapahdu itsestään, vaan ryhmän jäsenten täytyy olla siihen motivoituneita. Ketterät menetelmät korostavatkin sitä, että tiimissä tulisi olla motivoituneita yksilöitä.

4.2 Koordinointi

Vaikka ketterät menetelmät suosivatkin itseohjautuvuutta, on työtehtävien jakamisella silti tärkeä merkitys ketterissä projekteissa. Työtehtävät on usein jaettu esimerkiksi käyttäjätarinoiden (user story) avulla pienemmiksi kokonaisuuksiksi. Jotta voidaan toimia tehokkaasti, täytyy tiimin tietää, kuka työskentelee

minkäkin työtehtävän parissa. On myös tehokkuuden kannalta tärkeää seurata, kuinka paljon vaivaa kuhunkin tehtävään käytetään sekä kuka kyseistä työtä tekee. Koordinointi on myös tärkeää siksi, ettei turhaa työtä tehtäisi. Myös tätä, turhan työn välttämistä, ketterät menetelmät korostavat. Koordinointia hoidettiin tapaustutkimuksissa projektinhallintatyökaluilla, palavereilla, sähköpostilla sekä vierailuilla.

4.2.1 Koordinoinnin sovellusmenetelmät

Ehdottomasti suosituin koordinointikanava oli erilaiset projektinhallintajärjestelmät. Niitä käytettiin kymmenessä tapauksessa yhdeksästätoista (TAULUKKO 3). Lisäksi yhdessä tapauksessa käytettiin Wikiä Scrumin backlogin dokumentointiin. Tutkimuksissa mainittiin yleisesti Scrum backlog –työkalut sekä kaupallisia tuotteita kuten JIRA, GitHub, Redmine, Confluence sekä eXtreme Programming –työkalu XPlanner.

Kyseisiä työkaluja käytettiin ensisijaisesti käyttäjätarinoiden ja työtehtävien jakoon (Bannerman ym., 2012; Blomkvist ym., 2015; Layman ym., 2006; Lee & Yong, 2010; Modi ym., 2013; Paasivaara ym., 2008; Paasivaara ym., 2009; Persson ym., 2012; Pries-Heje & Pries-Heje, 2011; Ramesh ym., 2006; Razzak & Ahmed, 2014).

Myös tele- ja videokonferenssipalavereiden avulla jaettiin sekä priorisoitiin työtehtäviä (Modi ym., 2013; Noordeloos ym., 2012) sekä suunniteltiin tulevaa (Layman ym., 2006; Paasivaara ym., 2008; Paasivaara ym., 2009).

Sähköpostin avulla jaettiin palaverien muistiinpanot, joista selvisi palaverissa sovitut asiat (Paasivaaraym., 2012). Myös yhdessä tapauksessa työtehtävät jaettiin sähköpostin tai chatin avulla (Hole & Moe, 2008).

Vierailujen aikana suunniteltiin tulevaa (Korkala & Abrahamsson, 2007; Paasivaara ym., 2008; Paasivaara ym., 2009; Ramesh ym., 2006). Vierailua hyödynnettiin myös projektin käynnistämisessä (Pries-Heje & Pries-Heje, 2011).

TAULUKKO 3 Koordinoinnin sovellusmenetelmät lähdemateriaalissa.

Koordinointimenetelmä	**Esiintymät lähdemateriaalissa (kpl)**
Projektinhallintatyökalu	10
Palaveri (tele- tai videokonferenssi)	5
Vierailu	5
Sähköposti	2
Chat	1

4.2.2 Koordinoinnin sovellusmenetelmien hyödyt

Projektinhallintatyökalujen hyödyksi nähtiin se, että ne olivat aina kaikkien saatavilla (Blomkvist ym., 2015; Paasivaara ym., 2009; Razzak & Ahmed, 2014). Niiden hyödyksi nähtiin myös se, että kaikki tiesivät, mitä on tekeillä, kuka tekee mitäkin, kehen ottaa yhteyttä tiettyyn työtehtävään liittyen ja mikä on projektin aikataulu (Pries-Heje & Pries-Heje, 2011). Myös vierailujen todettiin parantavan koordinointia (Modi ym., 2013).

Ketterien menetelmien tarjoamien prosessien hyödyksi nähtiin se, että työtehtäviä jaettiin tehokkaasti – aina kun yksi tehtävä oli saatu tehdyksi, voi siirtyä heti uuteen. Tämä myös aiheutti sen, että hajautettu tiimi koki saavansa vastuuta ja arvostusta. (Noordeloos ym., 2012.). Ketterien menetelmien käytön koettiin myös helpottavan tietoisuutta työtehtävistä (Bannerman ym., 2012). Ketterien menetelmien käytön nähtiin selkeyttävän työnjakoa, projektin odotuksia, mitä muut ovat tekemässä ja myös sitä, mitä itse pitää tehdä (Pries-Heje & Pries-Heje, 2011).

4.2.3 Synteesi

Projektinhallintajärjestelmän käyttö olisi ehdottoman hyödyllistä globaalille ja ketterälle ohjelmistoprojektille. Sen käytöllä on erilaisia hyötyjä ja se tukee tehokkuutta sekä sitä, ettei turhaa työtä tehdä. Kyseinen järjestelmä tukee myös yhtä ketterän menetelmän piirrettä, itseohjautuvuutta, sillä se mahdollistaa sen, että kun yksi työtehtävä on tehty, voi siirtyä heti seuraavaan.

Toinen tärkeä hyöty, minkä projektinhallintajärjestelmä tarjoaa globaaleille projekteille on se, että sen avulla kaikki tiimiläiset tietävät aina, mitä on meneillään ja minkä ominaisuuden parissa työskennellään parhaillaan. Kun työntekijät ovat hajautuneet maantieteellisesti eivätkä istu samassa toimistossa, he eivät voi tarkistaa sanallisesti työkaveriltaan, mitä on tekeillä. Järjestelmä tarjoaa kaikille yhtäläisen mahdollisuuden seurata projektin etenemistä ja työtehtävien tekoa.

Ketterä menetelmä mahdollistaa globaaleissa projekteissa sen, että kaikki tietävät aina, mitä on tapahtumassa. Se tarjoaa myös tehokkuutta, mahdollisuuden itseohjautuvuuteen sekä selkeyttää työnjakoa. Johtopäätöksenä voidaan siis sanoa, että ketterän menetelmän käyttö globaaleissa kehitysprojekteissa tukee koordinaatiota ja helpottaa koordinaatio-ongelmia. Edellytyksenä tälle on kuitenkin projektinhallintajärjestelmän käyttö.

4.3 Kontrollointi

On ehdottoman tärkeää, että projektitiimissä on motivoituneita yksilöitä, jotta työtehtävät saadaan hoidettua ja jotta projekti onnistuu. Ihmisiä täytyy kontrolloida, jotta he tekevät oikeita ja hyödyllisiä asioita projektin kannalta ja tässä avainasemassa on johtaminen.

Jotta kontrollointi olisi onnistunutta, täytyy kahden edellä mainitun asian, kommunikoinnin ja koordinoinnin, olla onnistunutta. Jotta ihmiset saadaan tekemään asioita, joita projekti vaatii, täytyy esimerkiksi työtehtävien jakamisen olla tehokasta (Noordeloos ym., 2012).

Kontrollointia käsiteltiin lähdemateriaalissa huomattavasti vähemmän kuin kommunikointia ja koordinointia. Tämä saattaa johtua siitä, että kontrollointi saatetaan käsittää esimerkiksi osaksi kommunikointia: kun kommunikointi on onnistunutta esimerkiksi työtehtävien jaon osalta, ei erillisiä kontrollointitoimia välttämättä tarvita.

4.3.1 Kontrolloinnin sovellusmenetelmät ja niiden hyödyt

Tärkeimpänä kontrolloinnin sovellusmenetelmänä nähtiin itse ketterä kehitysmenetelmä. Se tarjosi selkeitä prosesseja ja käytänteitä, jotka motivoivat ihmisiä toimimaan halutulla tavalla. Se sai myös tiimiläiset olemaan ylpeitä tiimiydestä ja antoi energiaa tuotekeskeisyydestä ja nopeasta toimitustahdista. (Pries-Heje & Pries-Heje, 2011.).

Ketterän menetelmän käyttö osallisti kaikki tiimiläiset tärkeisiin tehtäviin kuten vaatimusmäärittelyyn, mikä loi yhteisöllisyyttä (Noordeloos ym., 2012). Ketterä menetelmä mahdollisti myös itseohjautuvuuden ja antoi näin luottamuksen tunnetta kehittäjiä kohtaan (Ramesh ym., 2012). Ketteryys tarjosi myös avointa näkyvyyttä koko projektiin sekä paransi kehittäjien välistä luottamusta (Paasivaara ym., 2009). Motivaation nähtiin myös lisääntyneen, koska tieto ja ihmiset olivat nopeasti saatavilla ja kaikki tiimiläiset nähtiin samanarvoisina ja kaikilla oli yhtäläinen mahdollisuus vaikuttaa työn lopputulokseen (Paasivaara ym., 2008).

4.3.2 Synteesi

Ketterien menetelmien käyttö globaalissa ohjelmistokehityksessä nähtiin lähdemateriaalissa avainehtona onnistuneelle kontrolloinnille. Ketterien menetelmien tuomat selkeät prosessit ja käytänteet motivoivat tiimiläisiä, ja motivoituneet yksilöt tekevät varmasti parempaa työtä kuin epämotivoituneet. Projektien työntekijät tunsivat itsensä myös tärkeiksi ja heihin luotettiin.

Koska menetelmä on kaikille projektissa tuttu, se tarjoaa yhteiset pelisäännöt ja näin eliminoi esimerkiksi kulttuurista johtuvia erilaisia käsityksiä liittyen johtamiseen. Itseohjautuvuus ja käytänteiden tarjoama tieto siitä, mitä projektissa on meneillään taas ehkäisee aikaeroista aiheutuvaa viivästymistä. Johtopäätöksenä voidaankin sanoa, että ketterien menetelmien tarjoamat prosessit ja käytänteet tukevat globaalin ohjelmistokehityksen kontrollointiongelmia.

5 YHTEENVETO

Tässä tutkielmassa tutkittiin ketterien menetelmien soveltamista globaaleissa ohjelmistoprojekteissa systemaattisen kirjallisuuskatsauksen avulla. Tutkielmassa etsittiin vastausta kahteen tutkimuskysymykseen, jotka ovat seuraavanlaiset:

3. Miten ketteriä menetelmiä sovelletaan globaalissa ohjelmistokehityksessä?

   ja

4. Miten nämä sovellusmenetelmät tukevat ketterien menetelmien käyttöä globaaleissa ohjelmistoprojekteissa?

Tutkimus noudatti Salmisen (2011, mukaillen Fink, 2005) systemaattisen kirjallisuuskatsauksen mallia. Tutkimuksen lähdemateriaali etsittiin ja seulottiin huolella, jotta synteesi olisi mahdollista tehdä sekä tulokset olisi luotettava yleistää. Tutkimusmenetelmä sekä lähdemateriaalin valinta esiteltiin tarkemmin ensimmäisessä luvussa eli johdannossa.

Toinen luku käsitteli ketteriä menetelmiä, ketteryyden määritelmää sekä ketterien menetelmien ominaisuuksia. Lisäksi esiteltiin, miten ketterät menetelmät ovat syntyneet ja kuvailtiin lyhyesti muutamaa yleisintä ketterää menetelmää. Ketterät menetelmät ovat syntyneet, kun kehitysprojekteja on pitänyt toteuttaa epävarmassa ympäristössä, jossa vaatimukset muuttuvat ja tuotteita pitää toimittaa nopeasti (Highsmith & Cockburn, 2001a, 4). Yhden näkökulman mukaan Basilin ja Turnerin (1975) Iterative Enhancement –tekniikka on ollut lähtökohtana ketterille menetelmille (Lindvall ym., 2002, 197–198). Ketterien menetelmien ominaisuudet kiteytyvät Ketterän ohjelmistokehityksen julistukseen (Beck ym., 2001), jossa korostetaan seuraavia asioita: 1) Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja, 2) Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota, 3) Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja ja 4) Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa.

Kolmannessa luvussa määriteltiin globaalin ohjelmistokehityksen käsite sekä seikat, jotka ovat ajaneet ohjelmistokehityksen hajautumiseen. Lisäksi kuvailtiin globaalin ohjelmistokehityksen hyötyjä ja ongelmia. Globaali ohjelmistokehitys on kasvavissa määrin useassa paikassa samanaikaisesti tapahtuvaa, monikulttuurista ja globaalisti hajautunutta (Herbsleb & Moitra, 2001, 17). Ohjelmistokehityksen globalisoitumista on ajanut muun muassa alhaisempien kustannuksien tavoittelu sekä mahdollisuus saavuttaa isompi määrä osaamista etätiimien sekä ulkoistamisen avulla (Herbsleb & Moitra, 2001). Globaalin ohjelmistokehityksen ongelmat on esitetty kattavasti luvussa löytyvästä taulukosta ja ne liittyvät kommunikointiin, koordinointiin sekä kontrollointiin.

Neljäs luku on itse kirjallisuuskatsausluku, jossa analysoitiin kirjallisuuskatsauksen materiaali ja etsittiin vastauksia tutkimuskysymyksiin. Tutkimuksesta selvisi, että ketterät menetelmät tarjoavat ratkaisuja globaalin ohjelmistokehityksen ongelmiin. Erilaisia ratkaisuja löytyi niin kommunikointi-, koordinointi-, kuin kontrollointiongelmiinkin. Kommunikointiongelmia tuki laaja joukko erilaisia kommunikointivälineitä ja –työkaluja, jotka tehokkaasti käytettyinä helpottivat globaalin ohjelmistokehityksen kommunikointiongelmia. Johtopäätöksenä voidaan sanoa, että mikäli tiimillä on käytettävissään erilaisia kommunikointityökaluja eri tarpeisiin, tarjoaa ketterä menetelmä ratkaisuja globaalin ohjelmistokehityksen kommunikaatio-ongelmiin.

Ketteristä menetelmistä löytyi myös tukea koordinointiin. Lähdemateriaalista selvisi vahvasti, että projektinhallintatyökalut helpottivat työtehtävien jakamista. Johtopäätökseksi saatiin, että mikäli käytettävissä on jokin projektinhallintatyökalu, ketterät menetelmät helpottavat myös globaalin ohjelmistokehityksen aiheuttamia koordinointiongelmia.

Kontrollointiongelmiin löytyi ratkaisu ketterien menetelmien tarjoamista selkeistä prosesseista ja käytänteistä. Kun kaikki tietävät nämä prosessit ja käytänteet, on johtaminen ja myös itseohjautuvuus helppoa. Ketterien menetelmien prosessit voivat helpottaa esimerkiksi kulttuurista johtuvia johtamiseen liittyviä erimielisyyksiä. Johtopäätöksenä voidaan sanoa siis, että ketterien menetelmien käyttö tukee myös globaalin ohjelmistokehityksen kontrollointiongelmia.

Rajoittavana tekijänä tässä tutkimuksessa tulee huomioida se, että lähdemateriaaliin hyväksyttiin hyvin erilaisia, esimerkiksi eri kokoisia tai eri alalla toimivia projekteja. Kriteeriksi riitti, että projektissa on käytössä jokin ketterä menetelmä ja että menetelmän prosesseja noudatetaan. Yksi jatkotutkimusaihe voisikin olla se, löytyykö globaaleista ja ketteristä projekteista eroavaisuuksia riippuen tiimin koosta. Olisi mielenkiintoista tutkia, miten ketterä menetelmä toimii todella suurikokoisessa hajautuneessa projektissa.

Koska ketterät menetelmät korostavat asiakasyhteistyötä, voisi toiseksi jatkotutkimusaiheeksi mainita esimerkiksi sen, miten hajautuneissa projekteissa huomioidaan asiakkaan kanssa tehtävä yhteistyö. Lisäksi olisi hyödyllistä vertailla hajautuneiden sekä samassa paikassa sijaitsevien tiimien tehokkuuden eroja – hajautuneet tiimit kuitenkin käyttävät merkittävän osan työajastaan esimerkiksi kommunikointivälineillä kommunikointiin.

LÄHTEET

Abrahamsson, P., Warsta, J., Siponen, M. T., & Ronkainen, J. (2003, May). New directions on agile methods: a comparative analysis. *Proceedings of the 25th International Conference on Software Engineering*, 244-254. IEEE.

Conchúir, E. Ó., Ågerfalk, P. J., Olsson, H. H., & Fitzgerald, B. (2009). Global software development: where are the benefits?. *Communications of the ACM*, *52*(8), 127-131.

Basil, V. R., & Turner, A. J. (1975). Iterative enhancement: A practical technique for software development. *IEEE Transactions on Software Engineering*, (4), 390-396.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Kern, J. (2001). Manifesto for agile software development.

Damian, D., & Moitra, D. (2006). Guest editors' introduction: Global software development: How far have we come?. *IEEE software*, *23*(5), 17-19.

Ebert, C., & De Neve, P. (2001). Surviving global software development. *IEEE software*, *18*(2), 62-69.

Fink, A. (2005). *Conducting research literature reviews: from the Internet to paper*. Sage.

Herbsleb, J. D. (2007, May). Global software engineering: The future of socio-technical coordination. In *2007 Future of Software Engineering* (pp. 188-198). IEEE Computer Society.

Herbsleb, J. D., & Moitra, D. (2001). Global software development. *IEEE software*, *18*(2), 16-20.

Highsmith, J. A. (2002). *Agile software development ecosystems* (Vol. 13). Addison-Wesley Professional.

Highsmith, J., & Cockburn, A. (2001a). Agile software development: The business of innovation. *Computer*, *34*(9), 120-127.

Highsmith, J., & Cockburn, A. (2001b). Agile software development: The people factor. *Computer*, *34*(11), 131-133.

Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., ... & Zelkowitz, M. (2002, August). Empirical findings in agile methods. In *Conference on Extreme Programming and Agile Methods* (pp. 197-207). Springer Berlin Heidelberg.

Salminen, A. (2011). Mikä kirjallisuuskatsaus. Johdatus kirjallisuuskatsauksen tyyppeihin ja hallintotieteellisiin sovelluksiin. *Vaasan yliopiston julkaisuja. Opetusjulkaisuja*, *62*.

Schwaber, K. (1997). Scrum development process. In *Business Object Design and Implementation* (pp. 117-134). Springer London.

Ågerfalk, J., & Fitzgerald, B. (2006). Flexible and distributed software processes: old petunias in new bowls. In *Communications of the ACM*.

KIRJALLISUUSKATSAUKSEN LÄHTEET

Bannerman, P. L., Hossain, E., & Jeffery, R. (2012, January). Scrum practice mitigation of global software development coordination challenges: A distinctive advantage?. In *System Science (HICSS), 2012 45th Hawaii International Conference o*n (pp. 5309-5318). IEEE.

Blomkvist, J. K., Persson, J., & Åberg, J. (2015, April). Communication through boundary objects in distributed agile teams. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (pp. 1875-1884). ACM.

Hole, S., & Moe, N. B. (2008, September). A case study of coordination in distributed agile software development. In *European Conference on Software process improvement* (pp. 189-200). Springer Berlin Heidelberg.

Korkala, M., & Abrahamsson, P. (2007, August). Communication in distributed agile development: A case study. In *Software Engineering and Advanced Applications, 2007. 33rd EUROMICRO Conference on* (pp. 203-210). IEEE.

Korkala, M., & Maurer, F. (2014). Waste identification as the means for improving communication in globally distributed agile software development. *Journal of Systems and Software*, 95, 122-140.

Layman, L., Williams, L., Damian, D., & Bures, H. (2006). Essential communication practices for Extreme Programming in a global software development team. *Information and software technology*, *48*(9), 781-794.

Lee, S., & Yong, H. S. (2010). Distributed agile: project management in a global environment. *Empirical Software Engineering*, 15(2), 204-217.

Modi, S., Abbott, P., & Counsell, S. (2013, August). Negotiating common ground in distributed agile development: A case study perspective. In *Global Software Engineering (ICGSE), 2013 IEEE 8th International Conference on* (pp. 80-89). IEEE.

Moe, N. B., Šmite, D., Šāblis, A., Börjesson, A. L., & Andréasson, P. (2014, September). Networking in a large-scale distributed agile project. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (p. 12). ACM.

Niinimäki, T. (2011, August). Face-to-face, email and instant messaging in distributed agile software development project. In *Global Software Engineering Workshop (ICGSEW), 2011 Sixth IEEE International Conference on* (pp. 78-84). IEEE.

Noordeloos, R., Manteli, C., & Van Vliet, H. (2012, August). From RUP to Scrum in global software development: A case study. In *Global Software Engineering (ICGSE), 2012 IEEE Seventh International Conference on* (pp. 31-40). IEEE.

Paasivaara, M., Durasiewicz, S., & Lassenius, C. (2008, August). Distributed agile development: Using Scrum in a large project. In *Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on* (pp. 87-95). IEEE.

Paasivaara, M., Durasiewicz, S., & Lassenius, C. (2009, July). Using scrum in distributed agile development: A multiple case study. In *Global Software*

Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on (pp. 195-204). IEEE.

Paasivaara, M., Lassenius, C., & Heikkilä, V. T. (2012, September). Inter-team coordination in large-scale globally distributed scrum: Do Scrum-of-Scrums really work?. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement* (pp. 235-238). ACM.

Persson, J. S., Mathiassen, L., & Aaen, I. (2012). Agile distributed software development: enacting control through media and context. *Information Systems Journal, 22*(6), 411-433.

Pries-Heje, L., & Pries-Heje, J. (2011, August). Why Scrum works: A case study from an agile distributed project in Denmark and India. In *Agile Conference (AGILE)*, 2011 (pp. 20-28). IEEE.

Razzak, M. A., & Ahmed, R. (2014, September). Knowledge sharing in distributed agile projects: Techniques, strategies and challenges. In *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on* (pp. 1431-1440). IEEE.

Ramesh, B., Cao, L., Mohan, K., & Xu, P. (2006). Can distributed software development be agile?. *Communications of the ACM*, *49*(10), 41-46.

Ramesh, B., Mohan, K., & Cao, L. (2012). Ambidexterity in agile distributed development: an empirical investigation. *Information Systems Research*, *23*(2), 323-339.