Markus Roslund

ORGANISAATION SISÄISTEN TILANNETEKIJÖIDEN VAIKUTUKSET TIETOJÄRJESTELMIEN KEHITYSMENTELMIEN VALINTAAN

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
2015

TIIVISTELMÄ

Roslund, Markus
Organisaation sisäisten tilannetekijöiden vaikutukset tietojärjestelmien kehitysmenetelmien valintaan
Jyväskylä: Jyväskylän yliopisto, 2015, 38 s.
Tietojärjestelmätiede, kandidaatintutkielma
Ohjaaja: Halttunen, Veikko

Tietojärjestelmien kehitysprojektit kohtaavat edelleen budjettiin, projektin keston venymiseen sekä järjestelmän laatuun liittyviä ongelmia. Tietojärjestelmiä kehitetään erilaisilla menetelmillä erilaisissa tilannetekijöiden muodostamissa konteksteissa. Osa edellä mainituista ongelmista liittyy tietyn menetelmän tai menetelmän osien huonosta soveltuvuudesta tiettyyn kontekstiin. Tämä tutkielma keskittyy tarkastelemaan organisaation sisäisiä tilannetekijöitä, joita ovat esimerkiksi projektin koko, organisaation kulttuuri, projektin tyyppi ja kehitystyön ulkoistaminen. Tutkielman tiedonkeruumenetelmänä käytetään kirjallisuuskatsausta. Tässä tutkielmassa pyritään selvittämään tarkasti ja laaja-alaisesti, mitkä ovat vaikuttavimmat organisaation sisäiset tilannetekijät ja kuinka ne vaikuttavat valitun kehitysmenetelmän käyttöön. Lisäksi tutkielmassa pyritään selvittämän, uusimpaan tutkimustietoon perustuen, kuinka tiettyjä menetelmiä voidaan käyttää erilaisissa haasteellisissa tilanteissa. Tutkielmassa tehdään karkea kahtiajako suunnittelukeskeisiin sekä ketteriin menetelmiin, joita käytetään tilannetekijöiden tarkastelussa. Tutkielmassa päädytään toteamaan, että nykyisin suurta suosiota saaneita ketteriä menetelmiä voidaan käyttää lähes missä tahansa kontekstissa, mutta tällöin menetelmä täytyy usein räätälöidä tavalla tai toisella paremmin projektin kontekstiin sopivaksi. Kirjallisuuskatsauksen perusteella tullaan myös johtopäätökseen, jonka mukaan tulevaisuudessa olisi syytä kehittää uusia menetelmiä, jotka voivat muokkautua laaja-alaisesti erilaisiin muuttuviin tilannetekijöiden määrittämiin konteksteihin.

Asiasanat: tilannetekijä, kontingenssitekijä, konteksti, kehitysmenetelmä, menetelmän räätälöinti

ABSTRACT

Roslund, Markus
How Organizations' Internal Situational Factors Affect Information Systems' Development Methods
Jyväskylä: University of Jyväskylä, 2015, 38 p.
Information Systems, Bachelor's Thesis
Supervisor: Halttunen, Veikko

Information system development projects still face problems concerning budget overruns, deadline overruns and lack of quality of the system. Information systems are being developed using several different method in different contexts which are formed from several factors called situational factors. Some of the earlier mentioned problems are related to the fit between context and the method or fragments of the method. This thesis focuses on inspecting organizations' internal situational factors such as project size, organizational culture, project type and outsourcing. The information gathering for this thesis was carried out as literature review. This thesis aims to find out accurately and pervasively, what are the most significant internal situational factors of an organization and how they affect on the use of the chosen development method. Further on, this thesis aims to find out, based on the most recent research, how different methods can be successfully used in a context in which they are considered to fit poorly. This thesis divides development methods roughly to plan-based methods and agile methods and uses this separation to inspect situational factors. This thesis concludes that, nowadays, popular agile methods can be used practically in any context but agile practices must be tailored one way or another to better fit the context at hand. Further on, based on the literature review, this thesis concludes that, in future, there is a need for development methods which are adaptable to any given context which is formed from individual situational factors.

Keywords: situational factor, contingency factor, context, development method, method tailoring

SISÄLLYS

1 JOHDANTO

Tietojärjestelmiä on kehitetty jo vuosikymmeniä erilaisilla menetelmillä. Kehitysprosessit, työtavat sekä työn laatu ovat kehittyneet jatkuvasti näiden vuosikymmenien aikana. Menetelmien kehittymisestä huolimatta kehitykseen liittyvät perustavanlaatuiset ongelmat ovat kuitenkin varjostaneet, ja varjostavat edelleen, koko järjestelmäkehitysprosessia (El Emam, 2008). Tällaisia ongelmia ovat esimerkiksi projektille osoitetun budjetin ylitys, projektiin kulutetun aikarajan ylittäminen, toteutetun järjestelmän huono laatu tai jopa projektin täysimääräinen hylkääminen (esim. El Emam, 2008; Fitzgerald, Hartnett & Conboy, 2006; Ropponen & Lyytinen, 2000). Vaikka järjestelmäkehitykseen käytetyt menetelmät ja prosessit ovat kehittyneet ajan saatossa, niin osassa projekteista ei päästä haluttuihin lopputuloksiin. Yksi syy projektien epäonnistumiseen on projektin toteutukseen käytetyn menetelmän huono soveltuvuus projektin kontekstiin. Projektin kontekstin määrittävät yhdessä erilaiset tekijät, joita kutsutaan tässä tutkielmassa tilannetekijöiksi. Tutkimustulosten perusteella tilannetekijöiden merkitys projektin onnistumiselle on niin merkittävä, ettei niiden huomioimista voida sivuuttaa (esim. Barki & Suzanne Rivard, 2001; Heeager, 2013; Virtanen, Pekkola & Paivarinta, 2013).

Tilannetekijöiden päätyyppejä on tunnistettu ainakin 44 kappaletta (Clarke & O'Connor, 2012). Näitä tekijöitä ovat esimerkiksi organisaation koko, projektin koko, toimiala, projektin budjetti, kulttuuri ja niin edelleen (Clarke & O'Connor, 2012). Tilannetekijät voidaan jakaa karkeasti organisaation sisäisiin sekä ulkoisiin tilannetekijöihin (Kruchten, 2013). Lyhyesti, menetelmä on tapa, jolla tietojärjestelmää kehitetään. Tämä tutkimus keskittyy organisaation sisäisten tilannetekijöiden vaikutusten tarkastelemiseen projektin kehitysmenetelmän näkökulmasta. Tarkastelussa käytetään perinteisten suunnitelmakeskeisten kehitysmenetelmien sekä ketterien kehitysmenetelmien näkökulmia. Tämä tutkielma ei pyri kuvaamaan kaikkien mahdollisten yksittäisten menetelmien kohtaamia haasteita vaan se pysyttelee yleisemmällä tasolla jakamalla menetelmät kahtia edellä mainitulla tavalla. Tämän tutkielman päämääränä on kirjallisuuskatsauksen avulla tuoda esille erilaiset tilannetekijöihin liittyvät haasteet,

joita tietojärjestelmäkehitysprojektit tavallisesti kohtaavat käyttäessään jotakin menetelmää.

Tutkimusongelmana on selvittää, että mitä ongelmia tietyn menetelmän käyttö tietyssä tilannetekijöiden muodostamassa kontekstissa luo. Lisäksi tutkimuksen tavoitteena on selvittää, kuinka näitä ongelmia on pystytty selättämään. Tutkimuskysymykset ovat seuraavat:

1. *Minkälaisia haasteita suunnitelmakeskeisten ja ketterien menetelmien käyttö voi aiheuttaa eri konteksteissa?*
2. *Kuinka menetelmän sovittamisesta tiettyyn kontekstiin syntyviä haasteita voidaan ratkoa?*

Ensimmäiseen tutkimuskysymykseen vastataan kolmannessa luvussa ja toiseen tutkimuskysymykseen vastataan neljännessä luvussa.

Tämän tutkimuksen tiedonkeruumenetelmänä käytettiin kirjallisuuskatsausta. Kirjallisuuskatsaus pyrittiin rakentamaan aihealueeseen liittyvien merkittävien, eniten lähteenä käytettyjen, lähteiden sekä uusimpien, empiiristä tietoa tarjoavien lähteiden varaan. Tähän aihealueeseen liittyvää empiiristä aineistoa tarkasteltaessa on huomioitava, että järjestelmäkehityksen menetelmiä tutkiva ala etenee käytäntö edellä, tutkimus perässä (Ågerfalk, Fitzgerald & In, 2006). Tästä syystä monet uusimmat viitekehykset perustuvat organisaatioissa käytettäviin ”parhaisiin tapoihin”. Lähdeaineistoa haettiin pääasiassa Google Scholar –palvelun kautta. Hakusanoina käytettiin muun muassa seuraavia termejä: *situational factor*, *contingency factor*, *context factor*, *environmental factor, software development method*, *agile*, *plan-based*, *method engineering*, *method tailoring*, *method customization* sekä näiden termien erilaisia yhdistelmiä. Hakuprosessissa käytettiin paljon backward ja forward searchia. Lisäksi uusimman luotettavan tutkimustiedon saamiseksi tarkasteltiin vuosina 2013–2015 julkaistujen artikkelien otsikot, ja valittiin niistä aihealueen kannalta relevanteiksi koetut artikkelit lähempään tarkasteluun. Luotettavimpana alaa koskevina julkaisuina käytettiin Jyväskylän yliopiston tietojenkäsittelytieteiden raportointiohjeessa mainittuja kahdeksaa tietojärjestelmätieteen julkaisua sekä neljää ohjelmistotuotannon julkaisua (Liite 1).

Kirjallisuuskatsauksen perusteella on erittäin selvää, että tilannetekijät vaikuttavat käytetyn järjestelmäkehitysmenetelmän käyttöön, joka puolestaan vaikuttaa projektin onnistumiseen (Barki & Suzanne Rivard, 2001; Vijayasarathy & Butler, 2015). Mikään kehitysmenetelmä ei ole sellaisenaan yleispätevä, koska menetelmä on aina riippuvainen kontekstista, jossa menetelmää käytetään (esim. Barki & Suzanne Rivard, 2001; Heeager, 2013; Turk, France & Rumpe, 2005). Kirjallisuuskatsauksen pohjalta käy ilmi, että ketteriä menetelmiä voidaan käyttää niin sanotun ”agile sweet spotin” ulkopuolella esimerkiksi korkean turvallisuuden tai hyvin monimutkaisten järjestelmien kehityksessä globaalissa ympäristössä. Tällöin ketteriä menetelmiä ei usein kuitenkaan voida käyttää sellaisenaan, vaan niitä on muokattava kyseiseen kontekstiin sopivaksi (esim. Fitzgerald, Russo & O'Kane, 2003; Virtanen ym., 2013).

Tutkielman ensimmäisessä pääluvussa määritellään tämän tutkielman kannalta tärkeimmät käsitteet, jotka ovat tilannetekijät sekä kehitysmenetelmä. Toisessa pääluvussa esitellään kirjallisuuskatsauksen tuloksia ja vastataan siihen, mitä ongelmia tietyn menetelmän käyttö tietyssä kontekstissa voi luoda. Kolmannessa pääluvussa esitellään kirjallisuudessa esiin tulleita keinoja, joilla huonosti tietyn kontekstin kanssa yhteen soveltuvaa menetelmää voi muokata paremmin kontekstiin sopivaksi. Viidennessä kerrataan tutkielman sisältö ja esitetään jatkotutkimuskohteita yhteenvedon ja pohdinnan aikana.

2 TILANNETEKIJÄT JA TIETOJÄRJESTELMIEN KEHITYSMENETELMÄT

Tässä pääluvussa määritellään tämän tutkimuksen kannalta tärkeimmät käsitteet. Ensimmäisessä alaluvussa määritellään, mitä tilannetekijä-käsite tarkoittaa tietojärjestelmäkehityksen yhteydessä. Toisessa alaluvussa määritellään, mitä menetelmä tarkoittaa tietojärjestelmäkehityksen yhteydessä. Lisäksi samassa alaluvussa esitellään hieman tarkemmin, mitä tarkoittavat kahtena ääripäänä nähtävät suunnitelmakeskeiset sekä ketterät menetelmät. Sen jälkeen kolmannessa luvussa käsitellään tilannetekijöiden ja menetelmien suhdetta toisiinsa.

2.1 Tilannetekijät

Tilannetekijä on mikä tahansa tekijä, joka vaikuttaa tuotteen kehittämiseen tai sen tarjoamiin palveluihin (Weerd, Versendaal & Brinkkemper, 2006). Järjestelmäkehitysprojektiin ja sen kehitysmenetelmään vaikuttavat tilannetekijät voidaan jakaa kahteen osaan: organisaation sisäisiin sekä ulkoisiin tekijöihin. Organisaation ulkoiset tilannetekijät ovat koko organisaatioon vaikuttavia tekijöitä, kuten organisaation toimialan lainalaisuudet tai maailmanmarkkinatilanne. Organisaation sisäiset tilannetekijät ovat tekijöitä, jotka vaikuttavat organisaation sisäisellä tasolla. Organisaation sisäiset tilannetekijät voidaan jakaa organisaatiotason sekä projektitason sisäisiin tilannetekijöihin. Pienten organisaatioiden kohdalla projektin ulkoiset ja sisäiset tilannetekijät vaikuttavat kuitenkin samalla tasolla. (Kruchten, 2013.) Organisaation sekä yksittäisen projektin sisäiset sekä ulkoiset tekijät voivat vaikuttaa toisiinsa. Esimerkiksi maailmantalouden romahtaessa organisaation talous voi romahtaa ja sen myötä yksittäisen projektinkin talous romahtaa. Toisaalta esimerkiksi projektien työntekijöiden taitojen ja tiimien prosessien kehittyessä koko organisaation arvo voi muuttua asiakkaiden näkökulmasta houkuttelevammaksi, joka näkyy toimialalla kilpailun lisääntymisenä.

2.1.1 Organisaation sisäiset tilannetekijät

Järjestelmän kehitysmenetelmän käyttöön vaikuttavien tilannetekijöiden määrä on suuri (Clarke & O'Connor, 2012). Organisaation sisäisiksi tilannetekijöiksi voidaan luokitella esimerkiksi kehitettävän järjestelmän koko, järjestelmän monimutkaisuus, toimistotilat, käytettävissä oleva aika, työntekijöiden ammatilliset sekä sosiaaliset taidot, projektin maantieteellinen hajautus, projektin toimintojen ulkoistaminen, aikaero, asiakkaan saavutettavuus, vaatimusten volatiilisuus, loppukäyttäjän osallistaminen, järjestelmän turvallisuuden tärkeys, kulttuuri, kommunikaatiotavat, dokumentaation tarve ja niin edelleen. Tekijöitä on siis hyvin monia erilaisia ja niillä on monia erilaisia vaikutuksia kehitysprosessiin. Esimerkiksi globaalisti hajautetussa laajassa järjestelmäkehitysprojektissa ei voida kommunikoida niin paljon kasvokkain kuin esimerkiksi kymmenen hengen paikallisessa projektissa. Samassa esimerkkikontekstissa aikaero voi tuoda haasteita, koska ihmiset tekevät toisella puolella maailmaa töitä eri aikaan.

Vaikuttavien tilannetekijöiden laajan kirjon lisäksi lisähaasteita tuo myös järjestelmäkehitysprojektien dynaaminen luonne. Projektit ovat harvoin staattisia ja yksikin tapahtuma voi muuttaa kehitysprosessia (McLeod & Doolin, 2012). Tällaisia muutoksia tilannetekijöissä ovat esimerkiksi henkilöstömuutokset, vaatimusten muuttuminen sekä muuttunut käytettävä teknologia. Näihin erilaisiin muutoksiin on osattava jollain tavalla varautua, ja muutosten tapahtuessa niihin myös reagoida. Esimerkiksi järjestelmäkehitysprojektia ei voi laittaa tauolle projektipäällikön jäädessä yllättävälle pitkälle sairauslomalle. Kehitysprojektin kannalta onkin tärkeää jatkuvasti seurata näitä muutoksia ja muokata kehitysprosessia sen muuttuneeseen tilanteeseen paremmin sopivaksi (Xu & Ramesh, 2007).

2.1.2 Organisaation ulkoiset tilannetekijät

Organisaation ulkoisia tilannetekijöitä ovat tekijät, jotka vaikuttavat koko organisaation toimintaan. Ulkoiset tilannetekijät määrittävät kontekstin, jossa organisaatio toimii, eli organisaation toimintaympäristön. Organisaation ulkoisia tilannetekijöitä voidaan kutsua myös ympäristötekijöiksi (Bekkers, van de Weerd, Spruit & Brinkkemper, 2010). Tällaisia tekijöitä ovat esimerkiksi toimiala ja sen sääntely, maailmanmarkkinatilanne, tuotteen lokalisoinnin tarve, poliittinen ilmapiiri, teknologiset riippuvaisuudet, työntekijöiden liikkuvuussäännökset (Bekkers, van de Weerd, Brinkkemper & Mahieu, 2008; Bekkers ym., 2010) ja jopa luonnonilmiöt.

Organisaation ulkoisiin tilannetekijöihin voidaan harvoin vaikuttaa (Bekkers ym., 2010), ainakaan lyhyellä aikavälillä. Organisaatioissa voidaan harvoin päättää esimerkiksi lakimuutoksesta, jonka avulla halpa työvoima voitaisiin tuoda omaan maahan tekemään töitä lähtömaan minimipalkalla. Koska ulkoisiin tilannetekijöihin ei juurikaan voida itse vaikuttaa, ja niiden sanelemien sääntöjen mukaan on kuitenkin toimittava, joudutaan organisaatioissa muok-

kaamaan prosesseja ulkoisen ympäristön mukaan. Tämä tutkielma keskittyy kuitenkin tarkastelemaan organisaation sisäisiä tilannetekijöitä ja niiden vaikutuksia kehitysmenetelmän valinnassa.

2.2 Tietojärjestelmien kehitysmenetelmät

Tässä alaluvussa kuvataan tietojärjestelmän kehitysmenetelmän määrite lyhyesti muutamaa lähdettä käyttäen. Lisäksi tässä alaluvussa kuvataan erilaisten kehitysmenetelmäsuuntausten ääripäiden ominaispiirteitä, jotta tällä hetkellä käytössä olevien menetelmien luonteesta saadaan riittävä kuva tämän tutkimuksen tutkimustulosten hahmottamiseksi. Menetelmien määritelmät ovat suppeahkoja, eikä määritelmien ole tarkoituskaan olla yksityiskohtaisia eikä kokonaisvaltaisia, sillä liian tarkka menetelmän kuvaaminen ei palvelisi tämän tutkielma päämääriä.

2.2.1 Menetelmä yleisesti

Kehitysmenetelmä on tapa, jolla tietojärjestelmää kehitetään. Menetelmään voidaan luokitella kuuluvaksi kaikki toimenpiteet, jotka suoritetaan järjestelmän elinkaareen aikana aina suunnittelun, toteutuksen, implementoinnin ja ylläpidon aikana (Conboy, 2009). Menetelmään kuuluu kaikki projektin koordinoimiseen sekä toteutukseen liittyvät tavat sekä menetelmään liittyvät arvot ja tavoitteet (Conboy, 2009). Partanen (2014, 11) tiivistää menetelmän tarkoittavan ”ennalta määriteltyä joukkoa käsitteitä, tekniikoita, käytänteitä ja muita seikkoja, joita ohjelmistokehittäjät ja sidosryhmät voivat käyttää ohjelmiston kehittämisessä hyödyksi”.

Kehitysmenetelmien joukko on laaja. Yleisesti käytettyjä menetelmiä ovat muuan muassa vesiputousmalli, V-malli, Rational Unified Process (RUP), Rapid Application Development (RAD), Joint Application Development (JAD), Feature-Driven Development (FDD), Crystal Clear, Dynamic System Development Method (DSDM), Adaptive Software Development (ASD), Extreme Programming (XP), Lean Software Development (LD) ja scrum (Chow & Cao, 2008; Ruparelia, 2010). Kehitysmenetelmät ovat erilaisia ja ne voidaan jakaa eri kategorioihin monella eri tavalla. Kehitysmenetelmät voidaan jakaa esimerkiksi kahden ääripään kesken perinteisiin suunnitteluun perustuviin suunnitelmakeskeisiin menetelmiin ja iteratiivisiin ketteriin menetelmiin. Menetelmissä on paljon erilaisia käytäntöjä ja prinsiippejä. Toiset ovat enemmän suunnitteluun perustuvia ja toiset taas perustuvat enemmän tilanteisiin reagointiin. Menetelmät voivat lisäksi olla sekä vaiheittaisia että iteratiivisia. Tässä tutkimuksessa kuitenkin pysytellään abstraktimmalla tasolla käyttäen karkeaa kahtiajakoa suunnitelmakeskeisiin menetelmiin sekä ketteriin menetelmiin, koska tämä tutkimus pyrkii tutkimaan enemmän tilannetekijöiden vaikutuksia.

2.2.2 Perinteiset suunnitelmakeskeiset menetelmät

Perinteisien suunnitelmakeskeisten menetelmien yhteydessä puhutaan usein niin sanotusta *vesiputousmallista*. Vesiputousmallia luonnehditaan tavallisesti yksisuuntaiseksi, ylhäältäpäin johdetuksi, ei-iteratiiviseksi tapahtumien sarjaksi (Thummadi, Shiv, Berente & Lyytinen, 2011). Vesiputousmallia noudatteleva prosessi etenee vaiheittain. Vesiputousmallissa yksi vaihe tehdään valmiiksi ennen seuraavaan vaiheeseen siirtymistä. Nämä vaiheet ovat Munassarin ja Govardhan (2010) mukaan järjestelmän vaatimusmäärittely, ohjelmiston vaatimusmäärittely, arkkitehtuurisuunnittelu, yksityiskohtainen suunnittelu, koodaaminen, testaus ja ylläpito. Kehitysvaiheilla ei ole juurikaan päällekkäisyyttä. Edellisessä vaiheessa tuotetut tuotokset toimivat seuraavan vaiheen syötteinä. Jos kehitysprosessin myöhäisessä vaiheessa syntyy tarve muuttaa esimerkiksi vaatimusmäärittelyä, niin tällöin joudutaan muuttamaan isoa joukkoa muitakin tuotoksia. Muutoksien tekeminen taaksepäin usean eri vaiheen tuotokseen on kallista ja vie aikaa.

Vesiputousmallia on käytetty hyvin laaja-alaisesti vaikka sitä on kritisoitu paljon. Kritiikki johtuu muuan muassa suuresta dokumentaation määrästä, kankeasta ja kalliista prosessista, tarkan etukäteissuunnitelun mahdottomuudesta ja ohjelmistovirheiden näkymisestä vasta prosessin loppuvaiheessa, kun itse koodaus aloitetaan. Vesiputousmallissa on myös vahvuuksia. Vesiputousmalli on helppo ymmärtää ja ottaa käyttöön. Vesiputousmalli on laajalti käytetty ja mallin noudattama prosessi tunnetaan teoriassa erittäin hyvin. Vesiputousmallin vaiheisiin jaetussa prosessissa on selvät tuotokset sekä virstanpaalut, jolla prosessin etenemistä voidaan seurata ja mitata. Lisäksi vesiputousmallin nähdään toimivan hyvin kehittyneiden tuotteiden sekä heikkojen tiimien kanssa. (Munassar & Govardhan, 2010.)

2.2.3 Ketterät menetelmät

Ketterillä kehitysmenetelmillä tarkoitetaan yleisesti menetelmiä, jotka ovat syntyneet *Agile Menifeston* (Beck ym., 2001) periaatteiden pohjalta. Ketterien menetelmien keskeisimmät yhteiset arvot ovat (Beck ym., 2001):

- **Yksilöitä ja kanssakäymistä** enemmän kuin menetelmiä ja työkaluja
- **Toimivaa ohjelmistoa** enemmän kuin kattavaa dokumentaatiota
- **Asiakasyhteistyötä** enemmän kuin sopimusneuvotteluja
- **Vastaamista muutokseen** enemmän kuin pitäytymistä suunnitelmassa

Ketterien menetelmien on tarkoitus muuttaa kehitysprosessia joustavammaksi ja tukevammaksi muuttuvia tilanteita varten. Suosituimmat näistä menetelmistä ovat projektinhallintaan keskittyvä menetelmä *scrum* sekä enemmän kehityskäytäntöihin keskittyvä *Extreme Programming* (*XP*) (Pikkarainen, Haikara, Salo, Abrahamsson & Still, 2008).

Menetelmän sanominen ketteräksi on lähes merkityksetöntä, jos sitä ei aseteta mihinkään asiayhteyteen (Conboy, 2009). Conboyn määritelmän (2009) mukaan, tietojärjestelmien kehitysmenetelmien tapauksessa, ketteryys on "tietojärjestelmän kehitysmenetelmän jatkuva valmius luoda muutosta nopeasti tai luontaisesti, ottaa muutos avosylin vastaan etu- tai jälkikäteen, ja oppia muutoksesta samalla tuottaen arvoa (taloudellinen, laadukas, yksinkertainen) asiakkaalle hyödyntäen kehitysmenetelmän yhteisiä osia ja vuorovaikutussuhteita ympäristönsä kanssa". Organisaatio ei siis ole automaattisesti ketterä, jos se ottaa käyttöön ketteriä menetelmiä, vaan sen on oltava prosesseiltaan aidosti ketterä, jotta sitä voidaan kutsua ketteräksi (Kruchten, 2013). Abrahamssonin, Salon, Ronkaisen ja Warstan määritelmän (2002) mukaan kehitysmenetelmä on ketterä, jos kehitysprosessi on inkrementaalinen eli vähitellen kasvava, yhteistoiminnallinen, suoraviivainen ja mukautuva. Inkrementaalisuus tarkoittaa nopeiden syklien avulla tehtyjä jatkuvia pieniä julkaisuja. Yhteistoiminnallisuus painottaa asiakkaan kanssa jatkuvasti käytävää kommunikaatiota. Suoraviivaisuus tarkoittaa, että menetelmä on hyvin dokumentoitu, helposti opittava ja helposti muokattavissa. Mukautuvuudella he tarkoittavat muutosten sallimista viimeisillä mahdollisilla hetkillä.

Scrum keskittyy projektinhallinnan ketteriin keinoihin. Scrum on kevyt ja helppo ymmärtää, mutta vaikea hallita täydellisesti. Scrum on iteratiivinen ja vähitellen kasvavaa mallia noudattava järjestelmäkehityksen lähestymistapa, jonka peruspilareita ovat läpinäkyvyys, katselmus ja sopeutuminen. Järjestelmää kehitetään *sprinteissä*, jotka kestävät noin kuukauden tai vähemmän. Sprintin aikana on tarkoitus tuottaa valmiiksi jokin uusi osa järjestelmään alusta loppuun asti. Yhden sprintin aktiviteetteihin kuuluu sprintin suunnittelu, sprintin katselmointi ja sprintin retrospektiivi. Näiden aktiviteettien lisäksi scrumissa käytetään lyhyitä päivittäisiä päiväpalavereja, joiden aikana kukin työntekijä antaa nopean tilannekatsauksen työnsä edistymisestä. Scrumissa perinteinen suunnitteludokumentaatio on korvattu kevyimmillä *käyttäjäkertomuksilla*, *sprintin kehitysjonolla* ja *tuotteen kehitysjonolla*. Laajoja ja yksityiskohtaisia vesiputousmallin mukaisia suunnitteludokumentteja ei käytetä. (Sutherland & Schwaber, 2014.)

XP keskittyy enemmän kehityskäytänteisiin. Nämä käytänteet ovat Beckin (1999a) mukaan seuraavat:

- suunnitelupeli (planning game)
- pienet julkaisut (small releases)
- vertauskuvat (metaphor)
- yksinkertainen suunnittelu (simple design)
- testivetoinen kehitys (tests)
- refaktorointi (refactoring)
- pariohjelmointi (pair programming)
- jatkuva integrointi (continuous integration)
- koodin yhteinen omistajuus (collective ownership)
- koko ajan läsnä oleva asiakas (on-site customer)

- 40 tuntiset viikot (40-hour weeks)
- avoin työtila (open workspace)
- vain sääntöjä (just rules)

Käytänteiden lisäksi Extreme Programming sisältää projektin elinkaarimallin, joka koostuu Abrahamssonin, Salon, Ronkaisen ja Warstan (2002, 19) mukaan, Beckin (1999b) kuvailuun perustuen, kuudesta vaiheesta: tutkimusvaihe, suunnitteluvaihe, iteraatiovaiheet, tuotteistamisvaihe, ylläpitovaihe ja lopetusvaihe.

Tietojärjestelmien kehittämiseen voidaan käyttää, ja käytetäänkin enenemässä määrin, useiden menetelmien yhdistelmiä. Menetelmiä voidaan yhdistellä laajamittaisesti tai vaihtoehtoisesti ottaa käyttöön vain jotain yksittäisiä käytänteitä, esimerkiksi parikoodauksen. Menetelmiä yhdisteltäessä puhutaan *hybridimenetelmästä*. Menetelmien prosesseja voidaan myös muokata. Tällöin puhutaan menetelmän *räätälöinnistä*.

3 MENETELMÄN SOVELTUMINEN TILANNETEKIJÖIDEN MÄÄRITTÄMÄÄN KONTEKSTIIN

Erilaiset kehitysmenetelmät nähdään soveltuvan parhaiten tiettyihin tilanteisiin. Tässä luvussa kuvataan ketterien menetelmien sekä suunnitelmakeskeisten menetelmien soveltuvuutta erilaisiin konteksteihin, jotka muodostuvat kontekstin määrittävistä tilannetekijöistä. Luvussa pyritään tuomaan esille mahdollisimman kattavasti erilaisten menetelmäsuuntausten (ketterä, suunnitelmakeskeinen) heikkoudet ja vahvuudet erilaisissa tilanteissa. Ketteriä sekä suunnitelmakeskeisiä menetelmiä ja niiden yhdistelmiä on lukuisia. Tässä tutkielmassa kehitysmenetelmiä käsitellään enemmän abstraktilla tasolla kuin kunkin menetelmän yksityiskohtaisella tasolla, käyttäen tarkastelussa luvuissa 2.2.2 sekä 2.2.3 esiteltyjä määritelmiä suunnitelmakeskeisistä ja ketteristä menetelmistä. Luvussa esitetty tieto ei pyri toimimaan ohjeena vaan se pyrkii esittelemään kattavasti erilaiset tilannetekijöihin liittyvät seikat, jotka ovat kehitysmenetelmän soveltamisen kannalta olennaisia.

3.1 Projektin ja projektitiimin koko

Ketterät menetelmät nähdään soveltuvan parhaiten pieniin projekteihin, jotka toteutetaan pienessä organisaatiossa (Abrahamsson, Conboy & Wang, 2009; Kruchten, 2013). Projektin koko onkin yksi suurimmista järjestelmäkehitysprojektin epäonnistumista ruokkivista riskitekijöistä ketteriä menetelmiä käytettäessä (Kruchten, 2013), varsinkin hajautetuissa projekteissa (Hossain, Babar & Paik, 2009). Ketteriä menetelmiä käytettäessä työskennellään normaalisti avokonttorissa ja projektin sisäinen kommunikointi tapahtuu pääasiassa kasvotusten (Beck ym., 2001). Lisäksi, ketteriä menetelmiä käytettäessä kommunikoinnista ei synny kovin paljoa dokumentaatiota, koska ketterien menetelmien periaatteisiin kuuluva vapaamuotoinen kommunikointi ei tuota automaattisesti yhtä paljon dokumentaatiota, kuin suunnitelmakeskeiset menetelmät. Tästä voi syntyä haasteita suurissa projekteissa ja suurissa organisaatioissa, joissa projek-

tin kanssa tekemisissä on satoja ihmisiä organisaation sisältä sekä ulkopuolisista sidosryhmistä (Pikkarainen ym., 2008). Esimerkiksi ketteriä menetelmiä käyttäviin prosesseihin yleisesti kuuluvat päivittäiset päiväpalaverit voivat olla haasteellisia toteuttaa suurilla henkilömäärillä (McHugh ym., 2013). Yksinkertaisesti, kaikki työntekijät eivät voi aina kommunikoida kasvotusten, osallistua kaikkiin kokouksiin ja muistaa valtavan suurta määrää tietoa ilman apuvälineitä. Suunnitelmakeskeisiä menetelmiä käytettäessä prosessi tuottaa automaattisesti erilaisia artefakteja (esimerkiksi arkkitehtuurisuunnitelmat ja käyttötapauskaaviot), joita voidaan käyttää kommunikaation välineenä. Erilaisten järjestelmän kehitystä koskevien tietojen tarkistaminen voi olla useassa tapauksessa helpompaa dokumenteista, sillä sähköinen dokumentti ei ole aikaan tai paikkaan sidottu toisin kuin keskustelutilanne kasvokkain.

Vaikka suurikin määrä ihmisiä saadaan tilaan, jossa voidaan kommunikoida kasvotusten, niin projektin koon paisuessa ja pituuden venyessä ongelmia saattaa syntyä työntekijöiden vaihtuvuuden ja ihmisten muistiongelmien takia. Jos kommunikoitua tietoa ei ole dokumentoitu, se voi jäädä projektin jättäneen henkilön mukaan. Tämän lisäksi työntekijöiden vaihtuvuuden on havaittu aiheuttavan epäluottamusta, sillä luottamuksen muodostaminen vie aikaa (Moe, Šmite, Hanssen & Barney, 2014). Luottamuksen puute voi johtaa väärinymmärryksiin ja jopa tiedon pimittämiseen toisilta työntekijöiltä. Projektin venyessä ajallisesti pitkäksi, voivat projektissa koko ajan mukana olleetkin työntekijät unohtaa, mitä he ovat tehneet. (Xu & Ramesh, 2007.) Suunnitelmakeskeisiä menetelmiä käytettäessä edellä mainitut haasteet eivät ole niin merkittäviä, koska suuri osa järjestelmää koskevista tiedosta on dokumentoitu. Kun tieto on dokumentoitu, niin hiljaisen tiedon arvo vähenee. Edellä esiteltyjen seikkojen pohjalta voidaan todeta, että ketterien menetelmien periaatteiden mukaiset menetelmät eivät kaikissa tapauksissa lisää kommunikointia, läpinäkyvyyttä ja tiedon siirtymistä suurten projektien tapauksissa. Toiseksi voidaan todeta, että suunnitelmakeskeisiä menetelmiä käytettäessä projektin ja tiimin koolla ei ole yhtä paljon merkitystä, kuin ketteriä menetelmiä käytettäessä, suunnitelmakeskeisiä menetelmiä käytettäessä syntyy automaattisesti kattava dokumentaatio, jota voidaan hyödyntää erinäisissä asioissa.

3.2 Organisaation koko, hierarkia ja jäykkyys

Organisaation ominaisuudet vaikuttavat siihen, kuinka hyvin tietty kehitysmenetelmä soveltuu kyseiseen organisaatioon. Organisaation rakenne saattaa rajoittaa tiettyjen menetelmien tai käytäntöjen käyttöä kyseisessä organisaatiossa tai organisaation yksittäisessä projektissa. Tällaisia rakenteellisia seikkoja ovat muun muassa organisaation koko, hierarkia, jäykkyys ja organisaation kulttuuri.

Organisaation koko itsessään vaikuttaa organisaation kommunikointitapoihin. Suurissa organisaatioissa tiedon kulkeutuminen on usein haasteellista (esim. Austin & Devin, 2009; Bass, 2014; Xu & Ramesh, 2007). Organisaation

koon kasvaessa kaikkea tietoa ei voida kommunikoida enää kasvotusten ihmiseltä toiselle. Tiedon välittämiseen tarvitaan tästä syystä enemmän dokumentoitua tietoa (Xu & Ramesh, 2007). Ketteriä menetelmiä käytettäessä ”ylimääräisen” dokumentaation tuottaminen pyritään minimoimaan, koska siitä ei usein ole välitöntä hyötyä asiakkaalle (Fowler & Highsmith, 2001).

Erittäin pienten organisaatioiden tapauksessa edellä mainittuja kommunikointiongelmia ei esiinny samassa mittakaavassa kuin suurissa organisaatioissa, koska työntekijöitä on vähän, ja sen ansiosta tiedonjako on helpompaa. Erittäin pienissä organisaatioissa tietojärjestelmiä voidaan toteuttaa onnistuneesti sekä ketterillä (Kruchten, 2013) että suunnitelmakeskeisillä menetelmillä (Estler, Nordio, Furia, Meyer & Schneider, 2014). Sánchez-Gordónin ja O’Connorin (2015) tutkimuksessa erittäin pienet organisaatiot raportoivat, että asiakkaan tyytyväisyys on heidän tärkein prioriteettiinsa, koska organisaation taloudellinen selviytyminen on täysin kiinni asiakkaiden toimeksiannoista. Tärkein yksittäinen asia asiakkaiden tyytyväisyyden saavuttamisessa on toimiva ohjelma. Toimiva ohjelma syntyy onnistuneen vaatimusmäärittelyn pohjalta, joten vaatimusmäärittelyn onnistuminen on erittäin kriittinen tekijä projektin onnistumisen, ja koko organisaation olemassaolon kannalta. Samassa tutkimuksessa erittäin pienet organisaatiot kertoivat, että he joutuvat jättämään lähes kaiken dokumentaation pois, sillä dokumentaation tekemiseen ei ole tarpeeksi resursseja (aika, raha). Resurssipuutteen takia myös testaaminen on vähäistä. Vain kaikkein kriittisimmät, asiakkaan kanssa yhdessä päätetyt, tietojärjestelmän osat testataan. (Sánchez-Gordón & O’Connor, 2015.) Ihme, Pikkarainen, Teppola, Kääriäinen ja Biot (2014) vielä lisäksi havaitsivat, ettei erittäin pienillä organisaatioilla ole aikaa eikä rahaa dokumentoida uudelleenkäytettäviä komponentteja. Suunnitelmakeskeiset menetelmät perustuvat laajaan dokumentaation, jota ei pystytä resurssipuutteiden takia tuottamaan erittäin pienissä yrityksissä.

Edellä esiteltyjen tutkimustulosten valossa ketterien menetelmien voidaan katsoa soveltuvan, ainakin tietyiltä osin, luonnollisesti tietojärjestelmien kehitykseen erittäin pienissä organisaatioissa. Ketterien menetelmien soveltuvuudesta huolimatta, pienet organisaatiot eivät kuitenkaan käytä aktiivisesti mitään tiettyä ennalta määriteltyä kehitysmenetelmää, vaan he ovat kehittäneet omaan ympäristöönsä kulloinkin sopivat kevyet menetelmät (Sánchez-Gordón & O’Connor, 2015).

Organisaation koon lisäksi organisaation hierarkkisuus vaikuttaa käytettävään kehitysmenetelmään. Ketteriä menetelmiä käyttävät tiimit ovat usein itseään ohjaavia (Beck ym., 2001), joka saattaa tuottaa haasteita ylhäältäpäin johdetuissa organisaatioissa (Van Waardenburg & Van Vliet, 2013). Projektia koskeviin päätöksiin saatetaan joutua odottamaan hyväksyntää organisaation johtoportaasta, ennen kuin projektia voidaan taas jatkaa. Päätöksenteon siirtyessä myöhempään ajankohtaan, projektin aikataulu saattaa venyä ja näin luoda lisää kustannuksia.

Uusien menetelmien käyttöönottoa perinteisiä prosesseja käyttävässä organisaatiossa voi jarruttaa lisäksi organisaation sisäinen kulttuuri (Lindvall ym., 2004). Jos suuri osa organisaation työntekijöistä on tehnyt koko uransa ajan tie-

tojärjestelmäkehitystä suunnitelmakeskeisiä menetelmiä käyttäen, voi heidän olla vaikeaa omaksua uusia erilaisia kehitystapoja (esim. Hoda, Kruchten, Noble & Marshall, 2010; Laanti, Salo & Abrahamsson, 2011; Van Waardenburg & Van Vliet, 2013). Uusia kehitystapoja ei haluta ottaa käyttöön, koska uusien toimintatapojen hyötyjä ei nähdä (Van Waardenburg & Van Vliet, 2013). Uudet tavat voidaan kokea jopa haitallisiksi, koska omia tuttuja ajattelumalleja joudutaan muuttamaan ja uusia tapoja käytettäessä voidaan joutua osallistumaan enemmän projektin sisäiseen kanssakäymiseen (Van Waardenburg & Van Vliet, 2013). Ketterien menetelmien käyttö voi olla haasteellista organisaatioissa, joissa organisaation rakenne on hierarkkinen ja päätöksenteko on hyvin keskittynyttä, koska päätöksenteon keskittyminen johdolle on ristiriidassa ketterien menetelmien itseohjautuvien ja itsejohdettujen tiimien kanssa (Van Waardenburg & Van Vliet, 2013). Tältä osin suunnitelmakeskeiset menetelmät voidaan katsoa istuvan paremmin organisaatioon, jonka organisaatiorakenne on vertikaalinen, sillä suunnitelmakeskeiset menetelmät luonnehditaan useasti ylhäältäpäin johdetuiksi (Thummadi ym., 2011).

3.3 Organisaation kypsyys

Organisaation järjestelmäkehitysprosessien kypsyys parantaa projektin suorituskykyä (Di Tullio & Bahli, 2014) ja näin ollen parantaa projektin onnistumismahdollisuuksia. Organisaation prosessien kypsyyttä mitataan yleensä erilaisilla malleilla kuten *CMM* (*Capability Maturity Model*) –malleilla ja *ISO/IEC 15504* –standardilla (Fontana, Fontana, da Rosa Garbuio, Paula Andrea, Reinehr & Malucelli, 2014). Näillä malleilla ja standardeilla määriteltyjen organisaation prosessien kypsyystason määrittely voidaan nähdä soveltuvan paremmin perinteisiä menetelmiä käyttävien organisaatioiden prosessien kypsyyden arvioimiseen (Fontana ym., 2014), sillä ketterissä menetelmissä arvostetaan ihmisiä ja heidän keskinäistä vuorovaikutusta enemmän kuin prosesseja (Beck ym., 2001). Ketterien menetelmien prosessit eivät myöskään ole useimmiten kovin tarkasti kuvattuja, jotta niitä voitaisiin käyttää CMMI-DEV:n mukaiseen arvioimiseen (Fontana ym., 2014).

Ketterien menetelmien ja korkeimpien CMM-tasojen yhteensoveltuvuudesta on esitetty vastakkaisia näkökulmia. Fontana ym. (2014) uskovat, etteivät ketteriä menetelmiä käyttävät prosessit voi koskaan saavuttaa täyttä kypsyyttä siirtämättä kiintopistettään ihmisistä prosesseihin, sillä jos näin tapahtuu, prosessi ei ole enää ketterä. Toisaalta ketteriä menetelmiä on jo pystytty käyttämään onnistuneesti useissa korkeimman CMMI -kypsyystason organisaatioissa (Bass, 2014). Lisäksi, kehitysprosessien kypsyyttä ei ole kaikissa tapauksissa mielekästä tai edes mahdollista määritellä. Esimerkiksi useimmat erittäin pienet organisaatiot käyttävät täysiä omia muokattuja prosessejaan, koska heillä ei ole resursseja (aika, raha, henkilöstö) tai he eivät koe raskaista standardisoiduista prosesseista, kuten CMM:sta, olevan hyötyä (Sánchez-Gordón & O'Connor, 2015). Tämä ei kuitenkaan tarkoita sitä, etteivätkö erittäin pienet organisaatiot

olisi kiinnostuneita kehittämään prosessejaan (Sánchez-Gordón & O'Connor, 2015).

Olisi kapeakatseista sanoa, ettei ketteriä menetelmiä käyttävä organisaatio voi olla prosesseiltaan kypsä, jos se arvostaa ihmisiä enemmän kuin prosesseja. Prosessien kypsyys ketterien ja suunnitelmakeskeisten menetelmien kohdalla voidaankin luokitella erilaiseksi. Suunnitelmakeskeisissä menetelmissä kypsyydestä kertoo viimeiseen asti hiotut prosessit, kun taas ketteriä menetelmiä käyttävien organisaatioiden prosessien kypsyyttä kuvaavat paremmin ihmiskeskeiset määreet. Näitä määreitä tai periaatteita ovat Fontanan ym. (2014) mukaan projektin kanssa toimiminen kommunikoiden ja ollen motivoitunut, asiakkaista ja ohjelmiston laadusta välittäminen, vaatimusten muutoksen salliminen, tiedonjako, lähdekoodin ja testien hallinta ketteryyttä tukevilla työkaluilla ja menetelmillä, itseohjautuvuus kestävällä vauhdilla, ketterien käytäntöjen yhdenmukaistaminen ja jatkuva kehittäminen sekä helposti asiakkaille ja johdolle esiteltävien tulosten tuottaminen.

3.4 Projektin hajautus ja globaali ulkoistaminen

Nykyisin useita tietojärjestelmiä kehitetään maantieteellisesti hajautetusti ja ulkoistetusti. Projektin kehityksen ulkoistaminen johonkin toiseen organisaatioon lisää projektin epäonnistumisen riskiä (Jun, Qiuzhen & Qingguo, 2011). Tavanomainen tapaus on kehitysprojektin suorittavan osan, eli ohjelmoinnin, ulkoistaminen johonkin pienemmän palkkatason maahan, esimerkiksi Intiaan tai Kiinaan, jotta työvoimakustannuksissa voidaan säästää. Kauas ulkomaille ulkoistamisesta voi kuitenkin aiheutua useita erilaisia haasteita, jotka voivat jopa heikentää projektin laatua ja pidentää prosessia. Useimmat ongelmista liittyvät kommunikaatioon ja työn laatuun (Moe ym., 2014), mutta haasteita voi syntyä myös muista syistä.

Eri puolille maailmaa sijoitetut kehitysprojektin toimipisteet aiheuttavat jo sijaintinsa puolesta haasteita varsinkin ketteriä menetelmiä käytettäessä, sillä ketteriä menetelmiä käytettäessä jatkuvasti kasvokkain tapahtuvat tiimin keskeiset tapaamiset ovat keskeisessä asemassa tiedonvälityksessä. Ketterien menetelmien periaatteisiin kuuluva kasvotusten tapahtuva kanssakäynti ja päivittäiset lyhyet päiväpalaverit eivät myöskään ole jatkuvasti mahdollisia ajallisista sekä taloudellisista syistä, sillä matkustaminen vie aikaa sekä rahaa.

Järjestelmäkehitysprojektin maantieteellinen hajautuminen voi kommunikaatiopuutteen takia aiheuttaa tilanteen, jossa eri toimipisteissä ei tiedetä, mitä muualla tehdään. Tästä voi helposti syntyä päällekkäistä työtä tai järjestelmää voidaan kehittää eriäviin suuntiin. Molemmissa tapauksissa organisaatiolle syntyy lisäkustannuksia. (Virtanen ym., 2013.)

Aikaeron takia voi syntyä kommunikaatiohaasteita (Khan & Azeem, 2014). Ulkomaille ei voida usein olla koko ajan yhteydessä, koska toisella puolella maailmaa ei välttämättä olla töissä samaan aikaan. Kommunikointi ja varsinkin erilaisten neuvotteluiden ja kokousten pitäminen on käytännössä mahdotonta

normaalina toimistotyöaikana. Vaikka eri hajautettujen toimipisteiden työajat menisivätkin hieman päällekkäin, niin esimerkiksi kokoukset saattavat venyä pituudelta yli neljätuntisiksi (Hossain ym., 2009), jolloin normaali työaika joudutaan ylittämään. Ylityöt ovat puolestaan organisaatioille useimmiten kalliimpi vaihtoehto tehdä työtä. Hossain ym. (2009) mainitsevatkin aikaerosta johtuvan samanaikaisen kommunikointiajan puutteen olevan yksi olennaisimmista haasteista scrumin käytössä globaalin ulkoistetun projektin kontekstissa. Suunnittelukeskeisiä menetelmiä käytettäessä jatkuvasti järjestettävien kokousten merkitys voidaan nähdä pienemmäksi, sillä kommunikointi voi tapahtua osittain kehitysprosessin aikana syntyneen dokumentaation ja muiden artefaktien pohjalta (Turk, France & Rumpe, 2002). Kommunikointi ei siis ole kaikissa tapauksissa sidottuna aikaan ja paikkaan. Modernit kommunikaatioteknologian välineet auttavat tietenkin näissä ongelmissa. Niistä lisää edempänä tässä tutkielmassa.

Globaaleissa ympäristöissä organisaatio voi kohdata haasteita paikallisen lainsäädännön tai esimerkiksi viisumilupien kanssa. Työntekijöiden liikkuvuutta tai työaikoja voivat rajoittaa jotkin pykälät. Lisäksi laajoissa globaalisti ulkoistetuissa järjestelmäkehitysprojekteissa organisaatioiden monimutkaiset tekniset ja toimintaperiaatteelliset standardit voivat osoittautua haasteeksi (Bass, 2014). Nämä seikat on syytä selvittää aina projektikohtaisesti.

Globaalista ulkoistamisesta voi seurata tiimin sisäisiä luottamusongelmia. Estlerin ym. (2014) tutkimuksessa haastatteluun vastanneet työntekijät kokivat projektin henkilöstön kykyjen huononevan, mitä maantieteellisesti kauemmas ulkoistavan organisaation sijaintimaasta mennään. Jos kanssatyöntekijöiden taitoihin ei uskota, voi luottamus heikentyä. Luottamuksen heikentyminen voi puolestaan johtaa esimerkiksi väärinymmärryksiin ja tiedon pimittämiseen (Xu & Ramesh, 2007). Ketteriä menetelmiä käytettäessä hiljaisen tiedon rooli on suurempi kuin suunnitelmakeskeisissä menetelmissä vähäisen dokumentaation takia. Tämän haasteen osalta suunnitelmakeskeiset menetelmät voidaan nähdä paremmin soveltuvaksi globaaleihin ulkoistettaviin projekteihin.

Lomat eivät sijoitu aina samaan ajankohtaa kaikissa maissa. Lomien sijoittuminen projektin aikana eri maissa eri aikaan on järkevää huomioida jollain tavalla, jottei vastaan tule tilannetta, jossa työt seisovat pahimmillaan useamman yhtäaikaisen lomajakson ajan. Projektin etenemättömyydestä syntyy luonnollisesti lisäkuluja, jos työn etenemisen keskeytymistä ei ole otettu huomioon esimerkiksi toteutettavaa järjestelmää koskevaa sopimusta laadittaessa.

Projektin tyypistä riippuen, ulkomaille ulkoistaminen voi olla turvallisuusriski (Shao & David, 2007). Tällainen riski saattaa syntyä esimerkiksi, jos puolustusvoimien projekteja ulkoistettaisiin ulkomaille. Käytännössä mikään seikka ei voi taata tärkeiden järjestelmää koskevien tietojen päätymistä vääriin käsiin. Toisaalta näin voi tapahtua myös kotimaassa, mutta todennäköisyyttä voidaan pitää huomattavasti pienempänä esimerkiksi isänmaallisuuden sekä kehitysprosessin tarkan seurantamahdollisuuden takia.

3.5 Kulttuuriset eroavaisuudet

Kulttuuri vaikuttaa lähes kaikkeen kanssakäymiseen, työtapoihin ja käytänteisiin. Jokaisen organisaation sisällä on oma kulttuuri, joka muokkautuu ajan kuluessa (Laanti ym., 2011). Organisaation kulttuuri voi olla hyvinkin avointa uusille kokeiluille tai sitten organisaatiossa voi esiintyä jyrkkää muutosvastarintaa. Organisaation kulttuuri voi olla hyvin vertikaalinen (hierarkkinen) tai sitten horisontaalinen (tasavertaisempi).

Kulttuuri toimii usealla tasolla. Organisaation sisäisen kulttuurin lisäksi eri maantieteellisillä alueilla on omia käytäntöjään ja tapojaan. Tämän tyyliset kulttuurierot näkyvät helposti esimerkiksi globaaleissa ja ulkoistetuissa tietojärjestelmäkehitysprojekteissa. Pelkästään kielellisistä eroista voi seurata kriittisiä ongelmia projektin onnistumiselle (Khan & Azeem, 2014). Työntekijä voi jättää mielipiteensä tai näkemyksensä jopa kokonaan sanomatta, jos hän on epävarma omasta suullisesta ilmaisustaan vieraalla kielellä (Hossain ym., 2009). Lisäksi, kielellisten seikkojen, esimerkiksi aksentin tai murteen, takia ei aina ymmärretä täysin toista osapuolta. Tilannetta pahentaa vielä se, jos ei uskalleta kysyä tarkentavia kysymyksiä epäselvistä asioista. Uskalluksen puute kysyä epäselvistä asioista voi johtaa väärinymmärryksiin ja pahimmillaan huonolaatuiseen koodiin (Moe ym., 2014).

Aasialaisista kulttuureista tulevat työntekijät eivät useimmiten anna kielteistä vastausta eivätkä kritisoi mitään yleisissä tapaamisissa (Khan & Azeem, 2014). Tästä voi helposti syntyä väärinkäsityksiä eri kulttuuritaustaisten ihmisten välille, jos asiaa ei oteta huomioon esimerkiksi globaaleissa työympäristöissä. Jossain kulttuureissa, esimerkiksi Intiassa, voi ihmisten taustalla edelleen vaikuttaa kastijärjestelmä tai jokin muu yhteiskunnallinen hierarkkinen järjestelmä, joka voi rajoittaa kommunikaatiomahdollisuuksia työntekijöiden kesken (Khan & Azeem, 2014).

Eri kulttuureissa on erilainen työmoraali ja –etiikka (Khan & Azeem, 2014). Nämä erot voivat osoittautua haasteellisiksi, kun erilaisen työmoraalin omaavat henkilöt tekevät töitä yhdessä (Khan & Azeem, 2014). Esimerkiksi tavallisesti Aasiassa odotetaan vastausta viestiin nopeasti, Euroopassa hieman hitaammin, mutta Karibian maissa vastauksen saaminen saattaa kestää 3-4 päivää (Khan & Azeem, 2014). Eri kulttuureissa voi lisäksi olla eri käsityksiä siitä, mikä on laadukasta työnjälkeä ja mikä ei. Samanlainen käsitys laadusta on ensiarvoisen tärkeää, sillä esimerkiksi Moen ym. (2014) tutkimuksessa kaikki organisaatiot kertoivat globaalin ulkoistamisen lopettamisen suurimmaksi syyksi juuri tehdyn työn huonon laadun.

Työn huonon laadun lisäksi eri kulttuureissa voi olla eri käsityksiä projektiin sitoutumisesta. Esimerkiksi Estlerin ym. (2014) tutkimuksessa työntekijät jättivät ulkomaille ulkoistetussa projektissa projektin helpommin kuin sisäisissä projekteissa. Lähtemisen arveltiin johtuvat todennäköisesti liian yksitoikkoisesta työtehtävästä, sillä ulkoistetut tehtävät ovat usein triviaaleja ohjelmointitehtäviä. Moe ym. (2014) toteavat, että globaalisti ulkoistettujen projektien suurin

epäonnistumisen syy on ulkomailla sijaitsevan toimipisteen työntekijöiden suuri vaihtuvuus. Työntekijöistä ei saada lyhyessä ajassa tarpeeksi motivoituneita. Lisäksi uusien työntekijöillä ei ole aina tarpeeksi tietoa projektin aihealueesta ja toimialasta, joten heidän kouluttamiseensa kuluu paljon aikaa ja rahaa. (Moe ym., 2014.) Aasiassa työpaikan vaihto on myös helpompaa, sillä töitä on tarjolla enemmän kuin esimerkiksi Euroopassa (Estler ym., 2014).

Kulttuuriin liittyviä huomionarvoisia asioita on lukuisia edellä mainittujen lisäksi, mutta niiden kaikkien esitteleminen ei ole mahdollista tämän tutkimuksen osana. Lisäksi, on hyvä muistaa, ettei kulttuuria pystytä vieläkään tarpeeksi tarkasti edes määrittelemään. Koko kulttuuriin liittyvä ongelmakenttä liittyy ehkä eniten kommunikaatiotapojen eroihin eri kulttuurista tulevien ihmisten kesken. Mitään tiettyä menetelmää tärkeämpää onkin mahdollisesti toisen kulttuurin ymmärtäminen, yhteisten kommunikaatiopelisääntöjen sopiminen, riittävien sitouttavien sopimusten laatiminen sekä asioiden perille menemisen varmistaminen epäselvissä tilanteissa.

3.6 Uudelleenkäytettävät komponentit

Uudelleenkäytettäviä komponentteja tehdessä täytyy olla laajakatseisempi kuin keskittyä vain kehitteillä olevaan järjestelmään (Turk ym., 2005). Ketterät menetelmät nähdään soveltuvan heikosti uudelleenkäytettävien komponenttien toteutukseen, sillä ketterät menetelmät tuottavat lähtökohtaisesti ratkaisuja tiettyyn ongelmaan (Turk ym., 2005). Uudelleenkäytettäviä komponentteja halutaan tehdä vain, jos niillä nähdään olevan jatkossa taloudellisia hyötyjä tai jos ne parantavan kehitteillä olevan järjestelmän laatua. Uudelleenkäytettävän komponentin tulee olla myös erittäin luotettava, sillä virheen vaikutukset kertaantuvat jokaiseen järjestelmään, joka käyttää kyseistä komponenttia (Turk ym., 2002). Ketterät menetelmät tuottavat komponentteja, jotka selviävät tietyistä ongelmista, joita järjestelmä saattaa kohdata. Ei siis kaikista ongelmista, joita voi vaatia hyviltä uudelleenkäytettäviltä komponenteilta. Lisäksi, ketterien menetelmien periaatteisiin kuuluu ”turhan” dokumentaation minimointi. Dokumentoimaton uudelleenkäytettävä komponentti on käytännössä hyödytön, jos kukaan ei löydä sitä tai kukaan ei tiedä tarkalleen, mitä sillä on tarkoitus tehdä. (Turk ym., 2005.)

Suunnitelmakeskeisissä menetelmissä jokainen järjestelmään toteutettava osa suunnitellaan, joten suunnittelun aikana voidaan huomioida komponentin uudelleenkäytön mahdollisuus ja toteuttaa komponentti uudelleenkäyttöä ajatellen. Koska suunnitelmakeskeisiin menetelmiin kuuluu komponenttien dokumentointi, syntyy toteutettavasta komponentista aina samalla dokumentaatio, jolloin sen uudelleenkäytön voi nähdä mutkattomammaksi. Mikään asia ei sinällään estä ketteriä menetelmiä käyttäviä projekteja tuottamasta uudelleenkäytettäviä komponentteja, mutta näin toimittaessa prosessi voi menettää osan ketteryydestään, koska prosessissa joudutaan käyttämään aikaa suunnitteluun ja dokumentointiin (Turk ym., 2005).

3.7 Vaatimusten volatiilisuus

Tietojärjestelmän kehitykseen kuuluu olennaisena osana järjestelmältä vaadittavien vaatimusten kartoitus, määrittäminen sekä niiden hallinta. Suunnitelmakeskeisissä menetelmissä vaatimusmäärittely tapahtuu pääosin aivan projektin alkuvaiheessa. Tästä voi koitua ongelmia sillä, hyvin volatiilisissa ympäristöissä suunnitelmakeskeiset menetelmät voivat tuottaa ratkaisuja ongelmiin, jotka ovat jo muuttuneet (Austin & Devin, 2009). Järjestelmän vaatimusmääritykset pyritään tekemään suunnitelmakeskeisiä menetelmiä käytettäessä mahdollisimman tarkasti ja kattavasti, koska uusien vaatimusten määritteleminen sekä vaatimusten muuttaminen on usein raskas ja kallis operaatio, sillä muitakin kehitystyön lomassa syntyneitä artefakteja joudutaan muuttamaan vaatimusten muuttuessa tai niitä lisättäessä.

Yleisen käsityksen mukaan suuri osa järjestelmästä ja sen toiminnoista voidaan suunnitella etukäteen, jos järjestelmältä vaaditut ominaisuudet ja siltä vaadittava toiminnallisuus on hyvin tiedossa etukäteen (Barki & Suzanne Rivard, 2001). Tarkka ja kattava vaatimusten määritteleminen projektin alkuvaiheessa on kuitenkin useissa tapauksissa haasteellista useasta syystä. Ensinnäkin vaatimukset tulevat useimmiten asiakkaalta, mutta aina asiakaskaan ei tiedä, mitä kaikkea hän haluaa kehitettävältä tietojärjestelmältä. Sen lisäksi, että vaatimukset voivat muuttua ja niitä voi tulla lisää, asiakas ja vaatimusmääritystä tekevä asiantuntija eivät välttämättä täysin ymmärrä toisiaan, jolloin voi syntyä väärinkäsityksiä. Toiseksi, kehitettävällä tietojärjestelmällä ei välttämättä ole aina valmiiksi varsinaista asiakasta tai edes tiettyä loppukäyttäjää (Xu & Ramesh, 2007), jolloin oikeellinen vaatimusten määritys jää vaatimusmäärittelijän toimialan tuntemuksen ja ammattitaidon varaan. Tämän tyyppisiä kehitysprojekteja voivat olla esimerkiksi teknologiavetoinen innovatiivisen kehitysyksiön kokeilu luoda jotain täysin uutta.

Vaatimusten vaihtuvuutta voi pitää yhtenä keskeisimmistä tietojärjestelmän kehitysmenetelmää koskevista ongelmista, jos vaihtuvuuteen ei ole ennalta määriteltyjä toimintamalleja. Ketterät menetelmät pyrkivät pääsemään tästä ongelmasta eroon minimalistisella etukäteissuunnittelulla, jottei muutoksista koidu suuria kuluja raskaan muutosprosessin takia. Ketterien menetelmien perusperiaatteisiin kuuluu vaatimusten muutosten salliminen jopa kehitystyön loppuvaiheessa (Beck ym., 2001). Toisaalta, vaatimusten pysyessä lähes muuttumattomina, ketterät menetelmät voivat olla hieman tehottomia (Hoda ym., 2010). Tehottomuus puolestaan voi aiheuttaa vältettävissä olevia kustannuksia, jos asiakasorganisaation edustajaa ei tarvitsekaan kutsua paikalle esimerkiksi viikoittain.

3.8 Kommunikaatio asiakkaan kanssa

Suunnitelmakeskeisiä menetelmiä käytettäessä jatkuva kanssakäyminen asiakkaan kanssa ei ole aina välttämätöntä. Suunnitelmakeskeisiä menetelmiä käytettäessä on yleistä, että asiakkaan kanssa ollaan yhteydessä projektin alkuvaiheessa, kun tehdään kehitettävän tietojärjestelmän vaatimusmäärittelyä ja seuraavaksi vasta, kun toteutetusta järjestelmästä saadaan palautetta (Hoda, Noble & Marshall, 2011; Van Waardenburg & Van Vliet, 2013). Tähän on useita syitä. Ensinnäkin suunnitelmakeskeiset menetelmät perustuvat kattavaan suunnitteluun, jossa edellisen vaiheen tuloste on seuraavan vaiheen syöte. Kun vaatimusmäärittely on valmis, voidaan sen pohjalta (periaatteessa) vaiheittain toteuttaa koko järjestelmä valmiiksi asti. Toiseksi asiakasorganisaation yhteyshenkilö voi olla esimerkiksi henkilö, joka ei voi sitoutua pitkään järjestelmäkehitysprojektiin ja olla koko ajan kehitystiimin saatavilla. Edellä mainitusta seikasta johtuen voi syntyä tilanne, jossa epäselviä vaatimusmäärityksiä ei välttämättä edes saada tarkennettua, koska asiakasorganisaation edustaja ei ole tarvittaessa saatavilla (Hoda ym., 2010), ja järjestelmän kehittämisen jatkaminen joudutaan rakentamaan arvelujen varaan. Suunnitelmakeskeisiä menetelmiä käytettäessä kehitystä voidaan jatkaa asiakkaan hyväksymään suuntaan, koska siitä on tehty hyväksytty suunnitelma. Suunta saattaa toki olla väärä.

Ketteriä menetelmiä käytettäessä jatkuva kanssakäyminen asiakkaan ja kehitystiimin välillä on lähes välttämättömyys, ja Hoda ym. (2011) ovatkin tunnistaneet riittämättömän asiakaskontaktin olevan yksi suurimmista huolenaiheista ketteriä kehitysmenetelmiä käyttävissä projekteissa. Yhden iteraation aikana toteutetun järjestelmän osan katselmuksen yhteydessä saadun palautteen perusteella, järjestelmän vaatimuksia lisätään, tarkennetaan ja muokataan paremmiksi. Jos asiakasorganisaation edustajalta ei saada palautetta toteutusta järjestelmän osasta, voi projektin eteneminen hidastua tai järjestelmän kehitys voi lähteä väärään suuntaan asiakkaan näkökulmasta. Tästä seuraa helposti turhaa työtä ja sen takia taloudellisia menetyksiä. Yksi iso tekijä heikkoon asiakaskontaktiin on asiakasorganisaatiossa käytettävät erilaiset prosessit. Asiakasorganisaatiossa saatetaan olla tottuneita suunnitelmakeskeisiin menetelmiin eikä siten ymmärretä, että kehitystiimin kanssa on oltava kommunikaatioyhteydessä koko projektin keston ajan. (Van Waardenburg & Van Vliet, 2013.)

Barkin ja Suzanne Rivardin (2001) mukaan, päätöksentekijöiden täytyy olla enemmän mukana projektin toteutuksessa, jos toteutettavan järjestelmän lopullinen tarkoitus ei ole täysin selvillä. Lisäksi, jos projektiin liittyy iso riski, tarvitaan enemmän formaalia suunnittelua kuin pienen riskin projekteissa (esim. Austin & Devin, 2009; Barki & Suzanne Rivard, 2001). Näiden tutkimustulosten perusteella ketterät menetelmät soveltuvat siis heikommin korkean riskin sisältäviin projekteihin. Toisaalta, Jun ym. (2011) toteavat, että ulkoistaminen lisää riskiä, mutta ulkoistavan organisaation muodollisten suunnitteluja kontrollimekanismien käyttäminen on haasteellista, ja ulkoistettaessa olisikin

tärkeämpää käyttää epävirallisia kommunikointi- ja koordinaatiotapoja projektin hallintaan.

Edellä mainittujen seikkojen lisäksi on syytä muistaa, että tietojärjestelmän kehitysprosessi on dynaaminen, eikä asiakaskontaktien määrä ole välttämättä vakio (esim. tapaaminen kerran kahdessa viikossa). Mahdolliseen asiakaskontaktin tiheyden muutokseen on osattava varautua jollain tavalla.

3.9 Vahvasti säännellyt ja korkean turvallisuuden vaativat alat sekä niiden vaatima dokumentaatio

Ketterien tietojärjestelmäkehitysmenetelmien ja raskaasti säänneltyjä alojen nähdään yleisesti soveltuvan heikosti yhteen (esim. Fitzgerald, Stol, O'Sullivan & O'Brien, 2013; Heeager, 2013; McHugh ym., 2013), sillä säännellyille aloille tuotettujen tietojärjestelmien ja niissä käytettyjen prosessien täytyy olla tarkasti jäljitettävissä (Fitzgerald ym., 2013), jotta järjestelmän laatu ja erinäisten säädösten noudattaminen voidaan todentaa. Tuotteen jäljitettävyys ja laadun varmistus tapahtuu hyvin pitkälti hyödyntäen erilaista dokumentaatiota, jota ketteriä menetelmiä käyttävät prosessit tuottavat lähtökohtaisesti vähän. Erilaisista prosessin tuotoksista (artefakteista) voidaan tarkastaa esimerkiksi tiettyjen alaa koskevien säännösten täyttyminen (Hoda ym., 2010). Säädösten täyttymistä valvoo usein joku ulkopuolinen arvioija (Fitzgerald ym., 2013), joten säännösten noudattamisen tarkistaminen pelkästä koodista voi olla vaikeaa, koska tarkistaja ei välttämättä ymmärrä koodia lainkaan.

Tietyt alat ovat säänneltyjä tietojärjestelmien osalta yleisesti sen takia, koska järjestelmän pettäminen voi johtaa vakaviin menetyksiin. Näitä menetyksiä voivat olla esimerkiksi suuret taloudelliset menetykset tai pahimmassa tapauksessa virhe järjestelmässä voi johtaa ihmishenkien menetykseen. Tällaisia aloja ovat esimerkiksi ilmailuala, terveysala ja maanpuolustus. Aloja koskevien säännösten lisäksi kehitettävään järjestelmään voi vaikuttaa eri maissa käytössä olevat säännökset sekä erilaisten yhdistysten tai standardien mukaiset säännökset (Fitzgerald ym., 2013).

3.10 Projektin tyyppi, ylläpito ja julkaisutahti

Stankovic ym. (2013) eivät havainneet projektin tyypin vaikuttavan millään tavalla projektin onnistumiseen, jos projektissa käytettiin ketteriä menetelmiä. Toisaalta, aikaisemmin tässä tutkielmassa esiteltyjen argumenttien pohjalta, voidaan jo todeta, että käytettävä menetelmä ei ole kaikissa tapauksissa optimaalisin vaihtoehto jokaiseen projektiin. Tämän lisäksi menetelmän soveltuvuus voi muuttua projektin aikana, esimerkiksi siirryttäessä kehitetyn järjestelmän ylläpitovaiheeseen, sillä asiakkaan kanssa ei tarvitse olla jatkuvasti yh-

teydessä, eikä koko ajan tarvitse tuottaa iteratiivisesti jotakin uutta (Heeager & Rose, 2014). Huonosti soveltuvan menetelmän käyttö voi jopa luoda negatiivisia jännitteitä kehitystiimin sisällä, jos esimerkiksi koko ajan täytyy da ”turhaan” vaikka varsinaista tarvetta uudelle julkaisulle ei edes olisi (Hoda ym., 2010). Dokumentaation tarpeen nähdään lisääntyvän ylläpitovaiheessa, sillä se koetaan pakolliseksi järjestelmän julkaisun jälkeisen kehittymisen eheyden hallinnassa (Heeager & Rose, 2014). Ketteriä menetelmiä käytettäessä suuri osa kommunikaatiosta tapahtuu kasvotusten, mutta se ei ole aina tarpeellista tai edes tuottavuutta lisäävää ylläpitovaiheeseen siirtyneen järjestelmäkehitysprojektin kohdalla (Heeager & Rose, 2014). Edellä mainituista syistä suunnitelmakeskeisten menetelmien voidaan nähdä soveltuvan paremmin järjestelmäkehitykseen, jos projekti on siirtynyt ylläpitovaiheeseen. Toisaalta ylläpidon aikana voi tulla tarve korjata nopeasti jokin ominaisuus tai ongelma, ja ketterien menetelmien uusiin tilanteisiin nopeasti reagoiva ominaispiirre voi soveltua ylläpitoon kelvollisesti. Ylläpitovaiheessa ketteriä menetelmiä käyttävä tiimi voi kokea motivaatio-ongelmia, koska projektilla ei ole enää ”mitään” konkreettista tavoitetta (esim. projektin julkaisu), sillä ylläpitoprojekti ei ole koskaan valmis (Heeager & Rose, 2014).

Suunnitelmakeskeiset menetelmät nähdään soveltuvan laajojen ja monimutkaisten järjestelmien kehittämiseen paremmin kuin ketterien menetelmien, koska etukäteen tehtävä kattava suunnitteluprosessi ottaa huomioon tietojärjestelmän ohjelmistoarkkitehtuurin. Ketteriä menetelmiä käytettäessä tietojärjestelmän laajamittainen ja kokonaisvaltainen hahmottaminen voi olla vaikeaa, koska ketterät menetelmät pyrkivät välttämään liiallista suunnittelua ja ylimääräistä dokumentaatiota. Ketterät menetelmät pyrkivät ratkaisemaan eteen tulevat ongelmat sitä mukaan, kun niitä ilmenee. Ohjelmistoarkkitehtuuri on kuitenkin niin laaja, monimutkainen ja osistaan riippuvainen kokonaisuus, ettei kaikkia ongelmia voida helposti ratkaista ”lennosta”. Lisäksi, joidenkin tietojärjestelmän arkkitehtuurillisten osien muuttaminen voi olla lähes mahdotonta, koska ne toimivat niin keskeisessä osassa koko järjestelmää tai yhtä jopa isompaa järjestelmien joukkoa (Turk ym., 2002). Muutokset johonkin tiettyyn arkkitehtuurin osaan voivat osoittautua liian kalliiksi toteuttaa tai niiden vaikutukset voivat olla liian laaja-alaisia, joten niitä ei yksinkertaisesti ole kannattavaa toteuttaa (Turk ym., 2002). Tämän lisäksi organisaatioilla saattaa olla käytössä sellaisia vanhoja *legacy-järjestelmiä* (legacy systems), joihin ei uskalleta koskea, koska ei ole tarkalleen tiedossa, kuinka ne toimivat. Laajojen ja monimutkaisten järjestelmien kohdalla arkkitehtuurisuunnittelu on täten tarpeellista. Suunnitelmakeskeisten menetelmien voidaan siis nähdä sellaisenaan sopivan paremmin monimutkaisten, arkkitehtuurisuunnittelua vaativien, tietojärjestelmien kehitykseen.

4 MENETELMIEN KÄYTTÖ HAASTEELLISISSA KONTEKSTEISSA

Edellisessä pääluvussa esiteltiin kirjallisuudesta eniten esiin nousseita tietojärjestelmän kehitysmenetelmiä koskevia tilannetekijöitä, joiden takia tietyn menetelmän käyttö ei ole aina täysin ongelmatonta kaikissa konteksteissa, mutta jotakin menetelmää on kuitenkin käytettävä. Jos projektiin vaikuttavat tilannetekijät näyttävät aiheuttavan haasteita käytettäväksi suunnitellun menetelmän käytölle, on menetelmää usein muokattava, koska kehitettävästä järjestelmästä halutaan aina tehdä mahdollisimman laadukas toteutus. Tässä pääluvussa esitellään kirjallisuudesta löydettyjä ratkaisukeinoja tilannetekijöiden aiheuttamiin haasteisiin. Suuri osa lähdeaineistosta on uutta. Lähdetutkimusten aineistot ovat usein suppeita tapaustutkimuksia, joten kaikkien tutkimustulosten yleistämistä ei voi pitää vielä kovin luetettavana. Vaikka suuri osa lähdeaineiston tutkimuksista käsittelee vain pientä tutkimusaineistoa, voidaan todeta, että erilaisia menetelmiä pystytään käyttämään onnistuneesti lähes missä tahansa ympäristössä, mutta menetelmän käyttäminen tietyissä ympäristöissä saattaa vaatia menetelmän räätälöintiä.

4.1 Organisaation ja projektin koko sekä kulttuuri

Ison hajautetun tiimin hallinta voi olla haasteellista ainakin ketteriä menetelmiä käytettäessä esimerkiksi tiedonkulun kannalta. Hossainin ym. (2009) mukaan scrumia käytettäessä yleisesti käytetty strategia kyseisen ongelman ratkaisemiseksi, on jakaa isot tiimit pienempiin tiimeihin, jotka vuorovaikuttavat keskenään. Tehokkaan tiedonjaon turvaamiseksi voidaan käyttää esimerkiksi niin sanottua ”*scrum of scrums*” –mallia, jossa jokaisen pienen tiimin *scrummasterit* pitävät päivittäin omille tiimeilleen tiimin sisäisen koordinoivan tapaamisen. Oman tiimin tapaamisen lisäksi scrummasterit osallistuvat tiimienvälisiin koordinoiviin tapaamisiin, joita kutsutaan siis nimellä scrum of scrums. (Bass, 2014.)

Monet isot ja perinteiset organisaatiot noudattavat vertikaalista, ylhäältä johdettua, mallia. Ketterien menetelmien käytöstä voi koitua haasteita, jos organisaation rakenne on vertikaalinen. Tätä ongelmaa pystytään lieventämään muun muassa muuttamalla johdon ja sidosryhmien asenteita ketteriä menetelmiä kohtaan. Asenteita pystytään muuttamaan tekemällä ketterät menetelmät ja niiden tuomat edut tutuiksi johdolle ja sidosryhmille. Kun ketteriä menetelmiä käyttävät järjestelmäkehitysprosessit tunnetaan paremmin, on esimerkiksi helpompi kerätä ja priorisoida järjestelmän vaatimuksia sekä saada palautetta tehdystä työstä. Liiketoimintapuolen tiedot ovat usein jakautuneet hyvin laajalle. Liiketoimintapuolen tiedonkulku voidaan kuitenkin kerätä yhteen ja kanavoida tiimille vain *tuoteomistajan* (product owner) kautta, jolloin tiedonkulku sopii esimerkiksi scrumin kanssa paremmin yhteen. Tiedon kanavointi vain yhdelle henkilölle voi kuitenkin johtaa tilanteeseen, jossa liiketoimintapuolta on vaikeampaa osallistaa projektiin. Edellä mainittuun ongelmaan voidaan kuitenkin varautua parantamalla kommunikointia, jolla saadaan liiketoimintapuolen työntekijät ymmärtämään, että miksi joku asia ei toimi ja miksi heidän on tehtävä muutoksia vaatimusmäärittelyyn. Tähän voidaan käyttää esimerkiksi vapaamuotoisempia tapaamisia sekä liiketoimintapuolen osallistamista aktiivisesti järjestelmää demonstroiviin aktiviteetteihin. (Van Waardenburg & Van Vliet, 2013.)

4.2 Maantieteellinen hajautus ja ulkoistaminen

Kuten tässä tutkielmassa todetaan aikaisemmin, tietojärjestelmäkehitysprojektin maantieteellinen hajautus ja ulkoistaminen voivat aiheuttaa erinäisiä haasteita kehitysprosessille.

Kulttuurierojen ymmärrystä, yhteistä näkemystä sekä ymmärrystä toisen toimipaikan prosesseista pyritään parantamaan muun muassa erilaisien vierailujen ja työntekijävaihtojen avulla. Esimerkiksi ulkoistava ja ulkoistamisen kohteena oleva organisaatio voivat lähettää yhden työntekijän töihin toiseen organisaation, jotta toisen organisaation toimintatapoihin voidaan tutustua tarkemmin. Vierailujen ja vaihtojen lisäksi koko projekti voidaan käynnistää samassa toimipisteessä, ja ketteriä menetelmiä käytettäessä tehdä muutama iteraatio yhdessä ennen varsinaisen ulkoistamisosuuden aloittamista. (Hossain ym., 2009.) Eri kulttuureissa on eroavat käsitykset siitä, kuinka nopeasti johonkin viestiin täytyy vastata (Khan & Azeem, 2014). Tähän ongelmaan voidaan varautua esimerkiksi sopimuksella, joka velvoittaa vastaamaan viestiin jonkin aikaikkunan sisällä viestin saapumisesta.

Nykyaikaiset kommunikaatioteknologian välineet ratkaisevat tai ainakin pienentävät joitain haasteita, joita liittyy hajauttamiseen. Esimerkiksi videopuhelujen, videopuhelukonferenssien ja erilaisten pikaviestimien avulla voidaan korvata osa kasvokkain tapahtuvasta kommunikaatiosta.

4.3 Aikaero

Hajautukseen liittyviin aikaero-ongelmiin on pyritty löytämään ratkaisuja esimerkiksi työaikojen synkronoinnilla. Toisessa toimipisteessä työn voidaan aloittaa aikaisemmin ja toisessa myöhemmin, jotta yhteistä aikaa, esimerkiksi kokouksiin, on riittävästi. Lisää joustoa eriäviin työaikoihin on saatu muun muassa tekemällä etätöitä iltaisin sekä tekemällä tarvittaessa pidempiä päiviä (Hossain ym., 2009.).

Rajoitetun yhteisen ajan takia kokouksien kestoa voidaan rajoittaa lyhyemmäksi. Rajoitetun yhteisen ajan takia kokoukset ovat tehokkaampia ja niistä pidetään paremmin kiinni. Yömyöhään venyviin kokouksiin voi osallistua esimerkiksi vain tärkeimmät avainhenkilöt, jotka kommunikoivat tarvittavat asiat seuraavana päivänä omalle tiimilleen. Näin kaikkien ei tarvitse olla paikalla normaalien työaikojen ulkopuolelle sijoittuvissa kokouksissa. (Hossain ym., 2009.)

Dokumentaation lisääminen on yksi keino tukea kommunikaatiota tilanteissa, joissa kaikilla henkilöillä ei ole mahdollisuutta osallistua samaan aikaan kokoukseen. Kokouksen pääkohdat voidaan kirjoittaa esimerkiksi wikiin tai kehitysjonossa (backlog), johon on globaali vapaa pääsy, oleviin käyttäjätarinoihin voidaan liittää käyttötapauskaavioita, jotta väärinymmärryksiltä vältyttäisiin. (Hossain ym., 2009.) Lisäksi yksittäisten tiimien tuottamien tuotteen kehitysjonojen (product backlog) yhdistämisen on havaittu auttavan yhteistyötä vaativien työtehtävien suunnittelussa ja järjestelyssä (Van Waardenburg & Van Vliet, 2013).

4.4 Projektin tyyppi

Ketteriä menetelmiä koskevia, suunnittelun vähäisyydestä johtuvia, arkkitehtuuriongelmia on ratkaistu useilla eri tavoilla. Yksi näistä tavoista on niin sanottu ”kävelevä luuranko” (*walking skeleton*) (Cockburn, 2008). Kävelevä luuranko tarkoittaa sitä, että järjestelmän arkkitehtuurista tehdään tärkeimmät kohdat ja liitetään ne toisiinsa. Näin varmistetaan arkkitehtuurille merkittävien komponenttien keskinäisen kanssakäymisen toimiminen. Itse järjestelmän komponentit ja niiden toiminnallisuus toteutetaan myöhemmin iteraatio kerrallaan.

Tiukasti säännellyillä aloilla jostain on pystyttävä varmentamaan, että järjestelmä noudattaa tiettyjä säännöksiä. Tämä tapahtuu usein tarkistamalla asian jostain dokumentista. Ketteriä menetelmiä käyttävät prosessit kuitenkin pyrkivät minimoimaan ”ylimääräisen” dokumentaation. Tätä ristiriitaa on pyritty ratkaisemaan useilla eri tavoilla. Haasteista on päästy yli esimerkiksi etukäteen osoittamalla, että kullekin standardissa vaaditulle kohdalle löytyy käytetystä menetelmästä, joko sellaisenaan tai räätälöimällä, jokin aktiviteetti, jolla järjestelmä voidaan toteuttaa standardia noudattaen (Jonsson, Larsson & Punnekkat, 2012). Laadunvarmistus voidaan sovittaa esimerkiksi scrumin kanssa yhteen

niin, että erilaisten kehitysjonojen kohtia validoidaan jatkuvasti sekä kehittäjien, että asiakkaiden toimesta. Riskienhallinnan ongelmia voidaan lieventää laittamalla vaatimuksia tärkeysjärjestykseen, ja toteuttamalla suurimman riskin sisältävät komponentit ensimmäisenä. Järjestelmän jäljitettävyyteen on puolestaan kehitetty tehokkaita automaattisia työkaluja, joiden avulla tiettyjen ehtojen noudattamisen toteutuminen on helppo tarkistaa. Keinoja on siis monia. Haasteet liittyvät usein johonkin tiettyyn standardiin, jota täytyy noudattaa (esim. Jonsson ym., 2012) ja näin ollen mitään yksiselitteistä toimintatapaa kaikkien säänneltyjä toimialoja koskevien ongelmien ratkaisemiseen on haasteellista eritellä. Tärkeintä onkin huomioida, että myös ketteriä menetelmiä voidaan käyttää säännellyillä toimialoilla onnistuneesti, mutta oikeanlaisten työkalujen sekä menetelmän räätälöinti ovat avainasemassa onnistuneessa kehitysprosessissa (Fitzgerald ym., 2013).

Kehitysprojektin muuttuessa ylläpitovaiheeseen, projektin luonne muuttuu merkittävästi. Tavallisesti ketterien menetelmien on nähty soveltuvan huonommin ylläpitotehtäviin. Ketteriä menetelmiä kuitenkin pystytään käyttämään onnistuneesti myös ylläpitotehtävissä (Heeager & Rose, 2014), jopa globaalisti hajautetuissa ympäristöissä (Kilpala & Kärkkäinen, 2015). Tällöin vaaditaan kuitenkin menetelmän räätälöintiä tilanteeseen sopivaksi (Heeager & Rose, 2014). Heeager & Rose (2014) ovat koonneet sekä teoreettisen että empiirisen tutkimuksen pohjalta yhdeksän heuristiikan listan, jonka avulla ketteriä menetelmiä voidaan käyttää onnistuneesti järjestelmäkehitysprojektin ylläpitovaiheessa. Lista on seuraava:

1. Käytä sprinttejä ylläpitotyön organisoimiseen; tasapainota asiakkaiden portfolioiden tarpeet (esimerkiksi kriittiset ja kiireelliset tapaukset käsitellään ensin)
2. Salli odottamattomat ja kiireelliset korjauspyynnöt asiakkailta
3. Organisoi tiedon jakautuminen tiimissä, kun työntekijät toimivat irrallisten tehtävien parissa (vältä tilanteita, joissa vain yksi henkilö on ainut asiantuntija tietyn järjestelmän kanssa)
4. Organisoi työntekijöille useita vastuuasiakkaita (vain yhden sijasta)
5. Tasapainota dokumentaation ja kasvokkain käytävän kanssakäymisen suhde sopivaksi
6. Tee (mahdollisimman vähän) jäsenneltyjä käyttötapauksia kommunikaation avuksi
7. Vahvista (koodin) yhteisomistajuutta ja toisilta oppimista testaamalla toisten koodia
8. Löydä uusia tiimiä motivoivia tekijöitä korvaamaan aikaisempi luonnollisesti syntynyt motivaatio sprintin tai projektin loppuun saattamisesta
9. Pidä suunnittelukokoukset mahdollisimman lyhyinä ja tehokkaina

4.5 Kommunikointi asiakkaan kanssa ja asiakkaan osallistaminen

Ketterien menetelmien käyttö vaatii enemmän asiakkaan osallistamista, kuin perinteiset suunnitelmakeskeiset menetelmät. Jos ketteriä menetelmiä käytetään ympäristössä, jossa käytetään pääsääntöisesti perinteisiä strukturoituja prosesseja, voi syntyä erinäisiä ongelmia. Asiakkaaseen ei pystytä mahdollisesti olemaan niin paljon yhteydessä koko projektin ajan kuin olisi tarve esimerkiksi vaatimusmäärittelyn ja palautteen saamisen kannalta. Tätä ongelmaa on pyritty lieventämään asiakkaan sitouttamisella esimerkiksi allekirjoitetulla sopimuksella tai jakamalla vastuualueita asiakasorganisaatiossa useamman työntekijän kesken (Hoda ym., 2010). Näin kukaan asiakasorganisaation edustaja ei joudu sitoutumaan jatkuvasti yhteen projektiin pitkäksi aikaa. Jos projektin tyyppi on puolestaan sellainen, että sillä ei ole varsinaista asiakasta (esim. teknologiavetoinen innovaatiokokeilu), voidaan kuvitteellinen asiakaskunta rekrytoida ja testauttaa jotakin järjestelmän uutta toiminnallisuutta heidän avullaan (Hoda ym., 2010).

Järjestelmäkehitystä koskeviin sopimuksiin liittyy haasteita, joita pyritään ratkaisemaan pääsääntöisesti erilaisten joustavuutta edeltävien ratkaisujen avulla. Järjestelmäkehitystä koskevaa sopimusta voi olla tekemässä esimerkiksi myyntiosaston työntekijä, joka ei välttämättä osaa ottaa huomioon joustoa vaativia seikkoja aikataulun ja taloudellisten asioiden kannalta. Sopimus yritetään tehdä pienimpään mahdolliseen hintaan, joten projektin ensimmäisestä päivästä lähtien projektiin luodaan paineita. Tiukkoja sopimuksia koskevia ongelmia selätetään muun muassa lisäämällä ajallista ja rahallista puskuria jonkin tietyn toimenpiteen tekemiseen kuluvaan aika-arvioon. Eräs toinen tapa on sopimusten jakaminen osiin. Sopimus voidaan tehdä esimerkiksi yksi ominaisuus kerrallaan. Tämä mahdollistaa sen, että sopimusta on helpompi lähestyä, koska projektin keskeyttäminen ei epäonnistuessaan tuota niin paljon kustannuksia. (Hoda ym., 2010.)

5 YHTEENVETO JA POHDINTA

Tietojärjestelmien kehittämisprojekteja vaivaa edelleen ongelmat, jotka koskevat muun muassa budjetin ja määräajan ylitystä, järjestelmän huonoa laatua tai jopa koko projektin peruuttamista. Järjestelmän kehitykseen liittyy olennaisena osana kehitykseen käytettävä menetelmä. Tietyt menetelmät tai menetelmän osat soveltuvat paremmin erilaisiin projektin konteksteihin. Kontekstin määrittävät yksittäiset tilannetekijät, jotka voivat muuttua projektin aikana. Tämä tutkielma keskittyi tarkastelemaan organisaation sisäisiä tilannetekijöitä, joita ovat muun muassa organisaation ja projektin koko, järjestelmän tyyppi, kulttuuri, järjestelmää koskevat säännökset ja niin edelleen. Tarkastelussa käytettiin kahta näkökulmaa, jotka ovat suunnitelmakeskeiset menetelmät sekä ketterät menetelmät.

Tämän tutkielman ensimmäinen tutkimuskysymys oli: *Minkälaisia haasteita suunnitelmakeskeisten ja ketterien menetelmien käyttö voi aiheuttaa eri konteksteissa?* Ensimmäiseen tutkimuskysymykseen vastattiin laajasti toisessa pääluvussa. Kirjallisuuskatsauksen perusteella haasteita on todella iso joukko, joka koostuu eri teemojen alle luokiteltavista ongelmista. Tärkeimmät näistä ongelma-alueista ovat perustavanlaatuiset ongelmat, jotka voidaan liittää eri menetelmäsuuntauksiin. Suunnitelmakeskeiset menetelmät eivät esimerkiksi toimi hyvin, jos järjestelmän vaatimukset muuttuvat kehitysprosessin aikana, joka on aivan tavallista. Muutosprosessi on vaatimusmäärittelyn jälkeen hyvin kallis operaatio. Ketterät menetelmät kohtaavat puolestaan perustavanlaatuisia ongelmia esimerkiksi isojen ja kompleksisten järjestelmien tapauksissa, kun laajaa arkkitehtuurisuunnittelua tarvitaan järjestelmän kytkeytyvyyden varmistamiseksi. Muut ongelmat liittyvät hyvin pitkälti pohjimmiltaan erilaisiin organisaation sisäisiin sekä organisaatioiden välisiin kommunikaatio-ongelmiin. Keskeinen havainto on, että jonkin tietyn menetelmän paras hyöty saadaan esille vain tietyissä kehityskonteksteissa. Paras tapa lähestyä käytettävän menetelmän valintaa on räätälöidä menetelmä tilanteeseen sopivaksi vaikka se ei aina olisikaan helppoa.

Mikään menetelmä ei sellaisenaan toimi kaikissa tilanteissa ja konteksteissa. Menetelmiä pystytään kuitenkin onnistuneesti muokkaamaan ja yhdistele-

mään kulloiseenkin kontekstiin sopivaksi, ja esimerkiksi (räätälöidyillä) ketterillä menetelmillä voidaan onnistuneesti kehittää lähes minkä tahansa tyyppisiä tietojärjestelmiä. Menetelmien räätälöinnillä on kuitenkin aina hintansa. Esimerkiksi dokumentaation lisääminen ketteriä menetelmiä käyttävään prosessiin vähentää menetelmän ketteryyttä, eikä sitä voida kutsua enää aidosti ketteräksi. Ketteryys ei kuitenkaan pitäisi olla aina itseisarvo, sillä ”liiasta” ketteryydestä voi olla myös haittaa tietyissä tilanteissa, jos näihin tilanteisiin ei ole osattu varautua.

Tutkielman toinen tutkimuskysymys oli: *Kuinka menetelmän sovittamisesta tiettyyn kontekstiin syntyviä haasteita voidaan ratkoa?* Toiseen tutkimuskysymykseen vastataan tarkemmin kolmannessa pääluvussa. Pääsanomana voidaan kuitenkin todeta, että käytännössä kaikkiin eri menetelmille haasteellisiin tilanteisiin on löydetty jokin ratkaisu. Kaikkia ratkaisukeinoja ei kuitenkaan vielä voida yleistää, sillä useat tutkimukset perustuvat vain projekteissa työskentelevien henkilöiden omiin arvioihin, eikä todelliseen objektiiviseen tarkkailuun. Lisäksi otettaessa muualla toimineita käytäntöjä käyttöön on syytä muistaa, että käytäntö on toiminut jossain tietyssä kontekstissa. Tutkimuksissa pystytään harvoin eristämään ja täten tarkasti mittaamaan minkään tietyn yksittäisen tekijän vaikutuksia projektin kulkuun. Edellä mainittua seikkaa voidaankin pitää äärimmäisen vaikeana asiana, sillä järjestelmäkehitys on hyvin monimutkainen ja monikerroksinen sosiopoliittinen prosessi. Lisäksi, tietyn tekijän variointi on vaikeaa, sillä yksikään projekti ei käytännössä ikinä voi olla samanlainen.

Tämän tutkielman puutteina voidaan pitää tiedonhankintamenetelmää, sillä se ei ollut systemaattisin mahdollinen, ja varmasti jotain relevanttia tietoa on jäänyt käsittelemättä. Toisena puutteena voidaan pitää, ettei tutkielmassa käsitellä ollenkaan tilannekohtaista menetelmän koostamista (situational method engineering, SME) vaan tämä tutkimus perustuu enemmän kontingenssinäkökulmaan. SME on menetelmä, jossa tilanteeseen sopiva menetelmä koostetaan useista, valmiiksi määritellyistä, menetelmän osista (Brinkkemper, Saeki & Harmsen, 1999). Kontingenssinäkökulma perustuu puolestaan enemmän olettamukseen, että kehitysprosessia varten pystytään valitsemaan menetelmä, joka soveltuu parhaiten kyseiseen kontekstiin. Molemmissa lähestymistavoissa on kuitenkin omat ongelmansa. SME:ia kuvataan usein raskaaksi, kalliiksi, monimutkaiseksi ja vaikeasti toteutettavaksi tavaksi (Henderson-Sellers & Ralyté, 2010), jota ei ole edes juurikaan käytetty käytännössä (Kuhrmann, Méndez Fernández & Tiessler, 2014). Kontingenssilähestymistapa nähdään usein utopistisena, sillä kehittäjien pitäisi tuntea todella hyvin useita eri menetelmiä, mutta tosiasiassa kehittäjät tuntevat vain yhden metodin, ja sitäkään ei tunneta riittävän hyvin (Fitzgerald ym., 2003). Lisäksi, Conboyn (2009) mukaan, juuri oikean metodin valinta on kehittäjille äärimmäisen vaikeaa.

Ketterien menetelmien tuomat edut ovat huomattavat, mutta kuten tässäkin tutkimuksessa on esitetty, haasteita niiden käyttämisessä on edelleen. Jostain syystä ketterien menetelmien oppikirjaversioita yritetään mieluummin yhä uudelleen räätälöidä kontekstiin sopivaksi kuin kehittää täysin uusia ketteriä menetelmiä (Conboy, 2009). Tulevaisuuden tutkimuksessa pitäisi keskittyä

enemmän kahteen asiaan. Ensinnäkin tiettyjen tilannetekijöiden varioimiseen ja varioimisen seurauksiin pitäisi kehittää jokin luotettava tutkimusmenetelmä, jotta voidaan selvittää tietyn tilannetekijän ominaisuuksien synnyttämät seuraukset. Jotta tilannetekijöitä voidaan luotettavasti mitata, täytyy alalla olla yhteinen käsitteistön määritelmä, nimittäin esimerkiksi projektin onnistuminen tai prosessien kypsyys voivat tarkoittaa hyvin erilaisia asioita. Toiseksi pitäisi keskittyä luomaan uusia, kontekstin mukaan skaalautuvia menetelmiä, jotka pystyvät nopeasti reagoimaan myös projektin aikaisiin, tilannetekijöissä tapahtuviin, muutoksiin.

LÄHTEET

Abrahamsson, P., Conboy, K. & Wang, X. (2009). "Lots done, more to do": The current state of agile systems development research. *European Journal of Information Systems, 18*(4), 281-284.

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2002). *Agile software development methods: Review and analysis (VTT publications)* VTT Publications.

Ågerfalk, J., Fitzgerald, B. & In, O. P. (2006). Flexible and distributed software processes: Old petunias in new bowls. Teoksessa D. Crawford. (toim.), *Communications of the ACM,* (26-34). New York, NY, USA: Citeseer.

Austin, R. D. & Devin, L. (2009). Research commentary-weighing the benefits and costs of flexibility in making software: Toward a contingency theory of the determinants of development process design. *Information Systems Research, 20*(3), 462-477.

Barki, H. & Suzanne Rivard, J. T. (2001). An integrative contingency model of software project risk management. *Journal of Management Information Systems, 17*(4), 37-69.

Bass, J. M. (2014). How product owner teams scale agile methods to large distributed enterprises. *Empirical Software Engineering,* 1-33.

Beck, K. (1999a). Embracing change with extreme programming. *Computer, 32*(10), 70-77.

Beck, K. (1999b). *Extreme programming explained: Embrace change* Addison-Wesley.

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

Bekkers, W., van de Weerd, I., Brinkkemper, S. & Mahieu, A. (2008). The influence of situational factors in software product management: An empirical study. *Software Product Management, 2008. IWSPM'08. Second International Workshop On,* (41-48). Barcelona, Spain: IEEE.

Bekkers, W., van de Weerd, I., Spruit, M. & Brinkkemper, S. (2010). A framework for process improvement in software product management. *Systems, software and services process improvement* (s. 1-12) Springer.

Brinkkemper, S., Saeki, M. & Harmsen, F. (1999). Meta-modelling based assembly techniques for situational method engineering. *Information Systems, 24*(3), 209-228.

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

Clarke, P. & O'Connor, R. V. (2012). The situational factors that affect the software development process: Towards a comprehensive reference framework. *Information and Software Technology, 54*(5), 433-447.

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

Di Tullio, D. & Bahli, B. (2014). The impact of software process maturity on software project performance: The contingent role of software development risk. *Systèmes D'Information & Management, 18*(3), 85-116.

El Emam, K. (2008). A replicated survey of IT software project failures. *Software, IEEE, 25*(5), 84-90.

Estler, H., Nordio, M., Furia, C. A., Meyer, B. & Schneider, J. (2014). Agile vs. structured distributed software development: A case study. *Empirical Software Engineering, 19*(5), 1197-1224.

Fitzgerald, B., Hartnett, G. & Conboy, K. (2006). Customising agile methods to software practices at intel shannon. *European Journal of Information Systems, 15*(2), 200-213.

Fitzgerald, B., Russo, N. L. & O'Kane, T. (2003). Software development method tailoring at motorola. *Communications of the ACM, 46*(4), 64-70.

Fitzgerald, B., Stol, K., O'Sullivan, R. & O'Brien, D. (2013). Scaling agile methods to regulated environments: An industry case study. Teoksessa D. Notkin, B. Cheng & K. Pohl. (toim.), *Proceedings of the 2013 International Conference on Software Engineering,* (863-872). Piscataway, NJ, USA: IEEE Press.

Fontana, R. M., Fontana, I. M., da Rosa Garbuio, Paula Andrea, Reinehr, S. & Malucelli, A. (2014). Processes versus people: How should agile software development maturity be defined? *Journal of Systems and Software, 97*, 140-155.

Fowler, M. & Highsmith, J. (2001). The agile manifesto. *Software Development, 9*(8), 28-35.

Heeager, L. T. (2013). The agile and the disciplined software approaches: Combinable or just compatible? *Information systems development* (s. 35-49) Springer.

Heeager, L. T. & Rose, J. (2014). Optimising agile development practices for the maintenance operation: Nine heuristics. *Empirical Software Engineering,* 1-23.

Henderson-Sellers, B. & Ralyté, J. (2010). Situational method engineering: State-of-the-art review. *J.UCS, 16*(3), 424-478.

Hoda, R., Kruchten, P., Noble, J. & Marshall, S. (2010). Agility in context. *ACM Sigplan Notices,* (74-88). New York, NY, USA: ACM.

Hoda, R., Noble, J. & Marshall, S. (2011). The impact of inadequate customer collaboration on self-organizing agile teams. *Information and Software Technology, 53*(5), 521-534.

Hossain, E., Babar, M. A. & Paik, H. (2009). Using scrum in global software development: A systematic literature review. *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference On,* (175-184). Limerick, Ireland: IEEE.

Ihme, T., Pikkarainen, M., Teppola, S., Kääriäinen, J. & Biot, O. (2014). Challenges and industry practices for managing software variability in small and medium sized enterprises. *Empirical Software Engineering, 19*(4), 1144-1168.

Jonsson, H., Larsson, S. & Punnekkat, S. (2012). Agile practices in regulated railway software development. *Software Reliability Engineering Workshops*

(ISSREW), 2012 IEEE 23rd International Symposium On, (355-360). Dallas, Texas, USA: IEEE.

Jun, L., Qiuzhen, W. & Qingguo, M. (2011). The effects of project uncertainty and risk management on IS development project performance: A vendor perspective. *International Journal of Project Management, 29*(7), 923-933.

Khan, S. U. & Azeem, M. I. (2014). Intercultural challenges in offshore software development outsourcing relationships: An exploratory study using a systematic literature review. *Software, IET, 8*(4), 161-173.

Kilpala, M. & Kärkkäinen, T. (2015). Distributed scrum when turning into maintenance: A single case study. *Second International Conference on Computer Science and Information Technology (COSIT 2015),* Geneva, Switzerland (55-67).

Kruchten, P. (2013). Contextualizing agile software development. *Journal of Software: Evolution and Process, 25*(4), 351-361.

Kuhrmann, M., Méndez Fernández, D. & Tiessler, M. (2014). A mapping study on the feasibility of method engineering. *Journal of Software: Evolution and Process, 26*(12), 1053-1073.

Laanti, M., Salo, O. & Abrahamsson, P. (2011). Agile methods rapidly replacing traditional methods at nokia: A survey of opinions on agile transformation. *Information and Software Technology, 53*(3), 276-290.

Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., . . . Kähkönen, T. (2004). Agile software development in large organizations. *Computer, 37*(12), 26-34.

McHugh, M., McCaffery, F., Fitzgerald, B., Stol, K., Casey, V. & Coady, G. (2013). Balancing agility and discipline in a medical device software organisation. *Software process improvement and capability determination* (s. 199-210) Springer.

McLeod, L. & Doolin, B. (2012). Information systems development as situated socio-technical change: A process approach. *European Journal of Information Systems, 21*(2), 176-191.

Moe, N. B., Šmite, D., Hanssen, G. K. & Barney, H. (2014). From offshore outsourcing to insourcing and partnerships: Four failed outsourcing attempts. *Empirical Software Engineering, 19*(5), 1225-1258.

Munassar, N. M. A. & Govardhan, A. (2010). A comparison between five models of software engineering. *International Journal of Computer Science, IJCSI, 7*(5), 95-101.

Partanen, M. (2014). *Ketterän menetelmän räätälöinti.* Tietojärjestelmätieteen pro gradu -tutkielma. Jyväskylän yliopisto.

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P. & Still, J. (2008). The impact of agile practices on communication in software development. *Empirical Software Engineering, 13*(3), 303-337.

Ropponen, J. & Lyytinen, K. (2000). Components of software development risk: How to address them? A project manager survey. *Software Engineering, IEEE Transactions On, 26*(2), 98-112.

Ruparelia, N. B. (2010). Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes, 35*(3), 8-13.

Sánchez-Gordón, M. & O'Connor, R. V. (2015). Understanding the gap between software process practices and actual practice in very small companies. *Software Quality Journal,* 1-22.

Shao, B. & David, J. S. (2007). The impact of offshore outsourcing on IT workers in developed countries. *Communications of the ACM, 50*(2), 89-94.

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

Sutherland, J. & Schwaber, K. (2014,). The scrum guide. Haettu 31.8.2015 osoitteesta http://www.scrumguides.org/scrum-guide.html

Thummadi, B. V., Shiv, O., Berente, N. & Lyytinen, K. (2011). Enacted software development routines based on waterfall and agile software methods: Socio-technical event sequence study. *Service-oriented perspectives in design science research* (s. 207-222) Springer.

Turk, D., France, R. & Rumpe, B. (2002). Limitations of agile software processes. *Third International Conference on Extreme Programming and Flexible Processes in Software Engineering, XP2002,* Alghero, Italy (43-46).

Turk, D., France, R. & Rumpe, B. (2005). Assumptions underlying agile software development processes. *Journal of Database Management, 16*(4), 62-87.

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

Vijayasarathy, L. & Butler, C. (2015). Choice of software development methodologies-do project, team and organizational characteristics matter? *Software, IEEE*

Virtanen, P., Pekkola, S. & Paivarinta, T. (2013). Why SPI initiative failed: Contextual factors and changing software development environment. *System Sciences (HICSS), 2013 46th Hawaii International Conference On,* (4606-4615). Wailea, HI, USA: IEEE.

Weerd, I. v., Versendaal, J. & Brinkkemper, S. (2006). A product software knowledge infrastructure for situational capability maturation: Vision and case studies in product management. *Technical Report UU-CS, 2006*, 1-16.

Xu, P. & Ramesh, B. (2007). Software process tailoring: An empirical investigation. *Journal of Management Information Systems, 24*(2), 293-328.

LIITE 1 LUETTELO JULKAISUISTA

- European Journal of Information Systems (EJIS)
- Information Systems Journal (ISJ)
- Information Systems Research (ISR)
- Journal of the Association for Information Systems (JAIS)
- Journal of Information Technology (JIT)
- Journal of Management Information Systems (JMIS)
- Journal of Strategic Information Systems (JSIS)
- MIS Quarterly (MISQ)
- IEEE Transactions on Software Engineering(TSE)
- Communications of the ACM (CACM)
- Empirical Software Engineering (ESEJ)
- IEEE Software