hyväksymispäivä arvosana

arvostelija

Johtaminen perinteisissä ja ketterissä ohjelmistoprojekteissa

Eero-Veikko Laine

Helsinki 4.5.2014

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta – Fakultet – Faculty	Laitos – Institution – Department
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä – Författare – Author

Eero-Veikko Laine

Työn nimi – Arbetets titel – Title

Johtaminen perinteisissä ja ketterissä ohjelmistoprojekteissa

Oppiaine – Läroämne – Subject

Tietojenkäsittelytiede

Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä – Sidoantal – Number of pages
	4.5.2014	21 sivua

Tiivistelmä – Referat – Abstract

Ohjelmistoprojektien jatkuva epäonnistuminen on kiinnittänyt huomiota projektien johtamiseen ja projektinhallintaan. Perinteisessä projektijohtamisessa vahvan projektipäällikön rooli ja huolellisen ennakkosuunnittelun rooli korostuu, kun taas uudemmissa ketterissä (agile) menetelmissä painotetaan projektitiimin jäsenten yhteistä päätöksentekoa, jatkuvaan muutokseen mukautumista ja väliaikaisia johtajarooleja.

ACM Computing Classification System (CCS):
K.6.1 [Project and People Management]

Avainsanat – Nyckelord – Keywords

johtaminen, projektijohtaminen, projektinhallinta, ketterät menetelmät

Säilytyspaikka – Förvaringställe – Where deposited

Muita tietoja – Övriga uppgifter – Additional information

Sisältö

1 Johdanto

Käsittelen tutkielmassani ohjelmistoprojektien johtamista ja tähän liittyvää projektinhallintaa sekä perinteisessä että ketterässä (agile) viitekehyksessä. Ketterät menetelmät ovat tarjonneet vaihtoehdon perinteiselle projektijohtamiselle, erityisesti hajoittaessaan johtamisvastuuta koko projektitiimille.

Johdantoluvun jälkeen aloitan toisessa luvussa käsittelemällä ohjelmistoprojektien luonnetta yleisesti. Kolmannessa luvussa tarkastelen projektinhallinnan perusteita ohjelmistokehityksessä. Neljännessä luvussa keskityn perinteisen, ketterää edeltävän ohjelmistokehityksen johtamismalleihin. Viidennessä luvussa käsittelen ketterää kehitystä ja johtamista ketterissä ohjelmistoprojekteissa. Kuudes luku on yhteenveto koko tutkielmasta.

IT-alalla työ on usein organisoitu pienten ryhmien toteuttamiksi projekteiksi. Projektien tuottavuutta on pitkään pidetty huonona: ohjelmistokehitys on monimutkaista ja vaikeaa, ja asiakkaiden ja teknologian muuttuvat vaatimukset muuttavat jatkuvasti projektin luonnetta. Ohjelmistokehittäjät kohtaavat usein ongelmia, joita he eivät ole voineet ennakoida ja joihin heillä ole valmista vastausta. Tämä asettaa ohjelmistoprojektin johtamiselle erityisiä haasteita, joihin eri projektinhallintatekniikat ovat yrittäneet vastata.

Perinteisesti projektiryhmää on johtanut erillinen projektipäällikkö, joka koordinoi ohjelmistokehittäjien toimintaa ja viestii ryhmän, asiakkaiden ja ylempien esimiesten välillä. Projektipäälliköihin kohdistuu suuria odotuksia: heiltä edellytetään muun muassa hyviä organisointi- ja neuvottelutaitoja, paineensietokykyä ja teknistä osaamista. Projektijohdolle asetettujen odotusten ja projektien todellisen onnistumisasteen välillä oli kuitenkin vielä 1990-luvulla suuri kuilu, ja 2000-luvulle tultaessa ohjelmistoyritysten olikin katsottu epäonnistuneen kyvykkäiden projektipäälliköiden kouluttamisessa.

1990-luvulla syntyneet ketterät menetelmät ottivat vastuun jakamisen ja tiimin jäsenten sitouttamisen yhdeksi tärkeimmistä painopisteistään: esimerkiksi Scrum-projektitiimeissä valta on jaettu tarkasti määriteltyjen roolien kesken. Johta-

juus säilyttää merkityksensä myös vastuuta jakavissa tiimeissä, ja jaetussa johtajuudessa johtajan rooli määrittyy tilannekohtaisesti asiantuntijuuden mukaan.

Vaikka erilaiset johtamismallit korostuvat perinteisessä ja ketterässä projektinhallinnassa, johtamisen voi ohjelmistokehityksen yhteydessä määritellä ryhmän tavoitteiden saavuttamiseen tähtääväksi ihmisiin vaikuttamiseksi. Eri menetelmissä ja suuntauksissa johtamiseen yhdistetään yleensä strategisen näkemyksen luominen, kommunikaatio ja konfliktien ratkaiseminen. Tekniset kyvyt voivat korostua yksilösuorituksissa, mutta projektitiimin johtamisessa vuorovaikutustaidot ovat lopulta ratkaisevia.

2 Ohjelmistoprojektien luonne

Ohjelmistoprojektien kokoluokka vaihtelee [JUR99], mutta tässä tutkielmassa keskitytään suhteellisen pienten tiimien toteuttamiin projekteihin. Projekteja määrittävät yleensä alla luetellut ominaisuudet [JUR99].

- Projekteilla on ennalta määritelty tavoite.
- Projektit ovat väliaikaisia rakenteita.
- Projekteilla on ennalta määritelty alku ja loppu, ja ne täytyy saattaa loppuun tietyn ajan sisällä.
- Projektit ovat kertaluontoisia rakenteita, eikä sama projekti toistu.
- Projekteilla on ennalta määritelty budjetti, tai ainakin projekteilla on budjetillisia rajoitteita.
- Projektien parissa työskentelevät niitä varten koostetut tiimit, joiden jäsenet saattavat toimia tiimin parissa osa- tai kokoaikaisesti, projektin tarpeista riippuen.

IT-alalla projektit ovat olleet jo pitkään korostuneessa osassa [JUR99]. Käytännössä ohjelmistoja kehitetään pääasiassa projekteina [JUR99]. Ohjelmistoprojektien tuottavuutta on pitkään pidetty huonona: projekteja leimaa aikataulujen pettäminen, epäonnistunut taloushallinto ja asiakkaiden tyytymättömyys lopputulokseen [FAR06, JUR99]. Ratkaisuja näihin ongelmiin on etsitty paremmasta johtamisesta [FAR06,

JUR99].

Ohjelmistoprojekteissa on piirteitä, jotka tekevät niistä erityisen haastavia prosesseja [FAR06, JUR99]. Ohjelmistojen kehittäminen itsessään on monimutkaista [FAR06, JUR99, NER07], ja jatkuvasti muuttuva teknologia monimutkaistaa sitä entisestään [FAR06, JUR99]. Ohjelmistokehityksessä kohdataan paljon nk. pahoja ongelmia (wicked problems), jotka ovat yksilöllisiä ja vaikeita pukea sanoiksi [NER07]. Lisäksi asiakkaiden vaatimukset tai edes tiimin jäsenet eivät välttämättä pysy samoina koko projektin ajan [FAR06, ROY05]. Koska ohjelmistojen laatua voi olla vaikea mitata niiden abstraktien luonteen vuoksi, projekteille ei ole helppoa asettaa yksiselitteisiä tavoitteita [ROY05, JUR99].

3 Projektinhallinnan perusteet

Projektinhallinta on toimintaa, joka tähtää projektin tavoitteiden saavuttamiseen mahdollisimman tehokkaasti [JUR99]. Se on suunnittelua, organisointia, ohjausta ja resurssien hallintaa suhteellisen lyhyen tähtäimen projektin etukäteen määriteltyjen tavoitteiden saavuttamiseksi [JUR99]. Projektinhallintaa voi pitää tärkeimpänä yksittäisenä ohjelmistoprojektin onnistumiseen vaikuttavana tekijänä [PET03].

3.1 Johtajalle tärkeät taidot

Projektipäälliköt ja vastaavassa asemassa olevat toimijat tarvitsevat niin teknisiä taitoja kuin vuorovaikutustaitojakin ohjelmistoprojektia johtaessaan. Pinkowska et al. [PIN11] jakavat nk. ”pehmeät” taidot (soft skills) kuuteen ryhmään. Henkilöstöhallinto (human resource management) kattaa henkilöiden sijoittamisen oikeaan rooleihin tiimissä [PIN11]. Tiimihallinto (team management) taas on henkilö- ja asiakassuhteiden ylläpitämistä ja luottamuksen rakentamista [PIN11]. Konfliktinhallinta (conflict management) on mahdollisten konfliktien tunnistamista ja ennaltaehkäisyä [PIN11]. Konfliktinhallinta on tärkeää, koska pitkittyessään konflikti tekee tiimistä täysin tehottoman [PIN11]. Viestinnän hallinta on niin markkinointia kuin informaation laajaa välittämistä ja viestin perille menon varmistamista [PIN11]. ”Itsehallinta” (self management) on projektipäällikön ja muiden tiimin jäsenten kriittistä itsearviointia,

toimintansa kehittämistä ja työn ja muun elämän tasapainossa pitämistä [PIN11]. Viimeisenä pehmeistä taidoista Pinkowska et al. [PIN11] nostavat esiin johtajuuden (leadership), joka merkitsee tässä yhteydessä erityisesti tuloskeskeisyyttä, voitontahtoa ja tiimin jäsenten motivoimista ja kontrollia tavoitteiden saavuttamiseksi. Johtajuuteen kuuluu Pinkowskalla et al. [PIN11] myös muutokset hyväksyvän toimintakulttuurin luominen.

Peters [PET03] korostaa teknisten taitojen toissijaisuutta luovuuteen ja innovaatioon nähden. Peters [PET03] pitää myös Ganttin kaavion kaltaisten teknisten projektinhallintatyökalujen hallintaa tärkeinä, mutta alisteisena strategian ymmärtämiselle. Lisäksi Peters [PET03] kritisoi ohjelmistoalan johtamisen opettajia käytännön kokemuksen puuteesta, mikä heijastuu heidän kouluttamiensa johtajien osaamisessa.

Ohjelmistojen kehitykseen liittyvän teknisen osaamisen tärkeydestä projektin johtamisessa vaikuttaa olevan vähän tutkimusta. El-Sabaan [ELS01] mukaan tekninen osaaminen eri alojen projektien johtamisessa on tärkeää, mutta suhteessa merkityksettömämpää vuorovaikutustaitoihin ja organisointikykyihin verrattuna. Oletettavasti ohjelmistokehitys ei ole poikkeuksellinen ala tässä suhteessa.

3.2 Johtamistyylit

Faraj ja Sambamurthy [FAR06] ovat tutkineet erilaisten johtamistyylien vaikutusta tietojärjestelmäprojektien onnistumiseen. Heidän mukaansa johtamiseen ohjelmistoprojektissa liittyy olennaisesti käytännön koordinointi ja johtajan tekninen osaaminen [FAR06]. Käytännönläheisyydestä johtuen ohjelmistoprojektin johtamisen painopiste on pikemminkin käytännön toiminnassa kuin inspiroivassa muutosjohtamisessa [FAR06]. Johtamista voi pitää jatkumona, jonka ääripäät muodostavat ohjaava (directive) ja voimaannuttava (empowering) johtajuus [FAR06]. Ohjaavat johtajat tekevät päätökset itse ja ohjaavat aktiivisesti tiimin toimintaa [FAR06]. He käyttävät valtaasemaansa johtaakseen tiimin jäseniä itsevaltaisesti [FAR06]. Voimaannuttavat johtajat taas jakavat valtaansa ja vastuuta muulle tiimille sekä rohkaisevat tiimin jäseniä toimimaan oma-aloitteisesti ja osallistumaan päätöksentekoon [FAR06]. Voimaannuttavaa johtajuutta voi pitää myöhemmin käsiteltävän egottoman tiimirakenteen modernina vastineena [FAR06].

Ohjaavaa johtamista pidetään usein voimaannuttavaa johtamista huonompana johtamistyylinä [SOM05]. Erilaiset ohjelmistoprojektit hyötyvät kuitenkin erilaisista johtamistyyleistä [FAR06]. Jos työtehtävät ohjelmistoprojektissa ovat monimutkaisia ja vaativat paljon koordinointia ja luovuutta, voimaannuttava johtaminen on tehokasta [FAR06]. Samoin voimaannuttava johtaminen on suositeltavaa, jos ohjelmistoprojektin jäsenet ovat kokeneita [FAR06]. Yhteinen päätöksenteko ja koordinointi vie kuitenkin myös aikaa, ja ohjaava johtaminen soveltuu paremmin ohjelmistoprojekteihin, joihin sisältyy paljon rutiininomaisia työtehtäviä [FAR06]. Kokemattomat ohjelmoijat eivät välttämättä hallitse itse projektipäällikön tehtäviä, eikä heillä ole tarpeeksi kokemusta osallistua päätöksentekoon [FAR06]. Tiimin koostuessa kokemattomista ohjelmoijista ohjaava johtaminen tarjoaa heille turvalliset puitteet työskennellä, ja on siksi tässä tilanteessa suositeltava johtamismalli [FAR06].

4 Perinteinen projektijohtaminen

Siinä missä ketterä projektinhallinta pyrkii mahdollisimman suureen joustavuuteen, perinteisessä projektinhallinnassa korostuvat tarkat etukäteissuunnitelmat [NER07, JUR99]. Ketterää kehitystä edeltävä projektijohtaminen vaikuttaa lukemieni artikkelien perusteella korostavan projektipäällikön roolia ja tämän toimintaa. Myös hyvin RUP-työkalun kaltaiset muodolliset ja kurinalaiset prosessimallit korostuvat perinteisestä projektijohtamista heijastavissa artikkeleissa.

4.1 Projektipäällikön tehtävänkuva

Yksi tärkeimmistä projektipäällikön tehtävistä on näkemyksen ja strategian muodostaminen [PET03, JUR99]. Strategian muodostamisen tärkeyden voi nähdä osana perinteisen projektijohtamisen hyvää etukäteissuunnittelua korostavaa perinnettä. Tehtävä on haastava: projektipäälliköillä on tyypillisesti ohjelmoijan tausta, ja heillä on usein vaikeuksia irroittautua aiemmasta roolistaan ja hahmotella strategiaa laajemmin [PET03, RID09]. Strategian toteuttamiseen liittyy osaltaan myös aikataulun ja budjetin hallinta, joka on iso osa projektipäällikön työtä [JUR99]. Perinteisissä ohjelmistoprojekteissa toimittiin 2000-luvulle asti nk. vesiputousmallin mukaan, jossa projekti on jaettu toisiaan seuraaviin eri vaiheisiin [JUR99]. Varsinaiseen toteutuk-

seen päästään vasta toiseksi viimeisessä vaiheessa [JUR99], mikä kuvaa hyvin perinteisen ohjelmistokehityksen suunnitelmakeskeisyyttä.

Projektipäälliköllä voi olla käytössään suhteellisen tarkkaa muotoa noudattavia työkaluja kustannusanalyysiin, tavoitteiden asetukseen, aikataulutukseen ja vastaaviin työtehtäviin [FAR06, JUR99]. Esimerkkinä mainittakoon tässä Ganttin kaavio, jolla havainnollistetaan projektin tavoitteiden toteutumista suhteessa käytettyyn aikaan [JUR99]. Muita vastaavia työkaluja ovat PERT, COCOMO ja CMM [PET03]. Näiden hallinnointityökalujen käytön on arvioitu vaikuttavan ohjelmistoprojektien tuloksiin käytetyistä johtamismalleista riippumatta [FAR06]. Yleinen näkemys on, että muodollisten prosessien noudattaminen on hyödyllistä erityisesti, kun

- tiimissä on paljon jäseniä,
- projektissa on mukana useita sidosryhmiä,
- laatuvaatimukset ovat tarkkoja tai
- projektissa ollaan kehityskaaren loppuvaiheessa [ROY05].

Myöhemmät Scrumin ja XP:n kaltaiset ketterät menetelmät ovat nostaneet muodollisten vaatimusten noudattamisen näkyväksi osaksi hallintomalliaan [HOD10]. On myös huomattava, että esimerkiksi hyvin muodollista RUP-menetelmää on käytetty yhdessä ketterien menetelmien kanssa [MON12].

Projektipäälliköltä vaaditaan hyviä vuorovaikutustaitoja [FAR06, RID09, PET03]. Projektipäällikkö joutuu olemaan paljon vuorovaikutuksessa asiakkaan kanssa [JUR99, RID09], ja toisinaan hän joutuu ratkomaan organisaation sisäisiä konfliktitilanteita [RID09]. Projektipäällikön työhön kuuluu hänen alaistensa säännöllinen arviointi: projektipäälliköt kokevat tämän työn kuitenkin usein niin epämiellyttäväksi, että välttelevät sitä [PET03]. Projektipäällikön tulisi myös motivoida alaisiaan, mutta Petersin [PET03] mukaan he usein epäonnistuvat siinä, koska eivät ymmärrä ohjelmoijien haastehakuista ajattelutapaa. Kehittäjien motivointi rahalla on kallista ja tehotonta [PET03]. Ilmeisesti omaksuttuaan johtajaroolin projektipäällikön on vaikea enää samaistua ohjelmoijan näkökulmaan. On myös mahdollista, että projektipäälliköiksi valikoituu henkilöitä, jotka eivät ole alun perinkään toimineet tyypillisen ohjelmoijan tavoin.

4.2 Tiimirakenteet

Faraj ja Sambamurthy [FAR06] näkevät, että ohjelmistokehityksessä on laajalti sovellettu kahta eri arkkityyppistä tiimirakennetta, egotonta ja pääohjelmoijajohtoista tiimiä. Perinteisesti ohjelmistoprojektitiimit oli järjestetty virkaiän ja kokemuksen perusteella, kokeneen ohjelmoijan toimiessa tiimin johtajana [FAR06]. Vallan ja vastuun jako oli selkeää, ja jokaista nuorempaa ohjelmoijaa arvioi esimies [FAR06]. Tiimin jäsenten välistä kilpailua pidettiin luonnollisena, ja sillä uskottiin olevan motivoiva vaikutus [FAR06]. Työnjako tiimin keskuudessa oli selkeää [HOD13].

Yksi perinteisen tiimin muoto on pääohjelmajohtoinen tiimi, jossa pääohjelmoija (chief programmer) toteuttaa ohjelmistojärjestelmän ytimen ja vastaa kehittäjien työtehtävien integroinnista järjestelmään [FAR06]. Pääohjelmoijat ovat täysin vastuussa projektin onnistumisesta, mutta vastaavasti heille on annettu suuri valta ohjata tiimiä haluamallaan tavalla [FAR06]. Pääohjelmoijan ja muun tiimin muodostamaa rakennetta on verrattu toiminnaltaan kirgurgin johtamaan leikkausryhmään [FAR06]. Hoda et al. [HOD13] mainitsevat nk. leikkausryhmän erityisenä pääohjelmoijajohtoisen tiimin sovelluksena, jossa roolijako on erittäin pitkälle vietyä.

Vaihtoehtona pääohjelmajohtoiselle lähestymistavalle on ”egoton” projektitiimi, jossa valtaa on hajautettu ryhmän jäsenille [FAR06, WEI99]. Kommunikointi on vapaata, ja ilmapiiri kannustaa ohjelmoijia myöntämään virheensä ja tekemään yhteistyötä [FAR06, WEI99]. Myöhemmin käsiteltävässä ketterässä ohjelmistokehityksessä on huomattavan paljon samoja elementtejä kuin egottomassa tiimirakenteessa.

Kaksi edellä esiteltyä tiimirakennetta ovat arkkityyppisiä rakenteita, joiden tapaan perinteiset projektitiimit on yleensä järjestetty [FAR06]. Näillä rakenteilla on toisistaan eroava vaikutus työympäristöön [FAR06]. Pääohjelmoijajohtoisessa tiimissä tiimityötä ei palkita ja tiimin jäsenten riippuvuus toisistaan on pientä, kun taas egottoman tiimin rakenne kannustaa vertaisia koordinoimaan työtään keskenään [FAR06]. Egottomassa tiimissä kuitenkin esimerkiksi tehtävänjako, tavoitteiden asetus ja päätöksenteko kestävät pidempään [FAR06], oletettavasti siksi, että kaikkien tiimin jäsenten mielipiteet halutaan ottaa huomioon.

5 Ketterä projektijohtaminen

Ketterä kehitys haastoi 2000-luvulla radikaalisti käsityksiä ohjelmistokehityksen toimintamalleista [NER07]. Kun perinteisessä ohjelmistokehityksessä haluttiin varautua eri ongelmiin ja tehdä kattavaa analyysia mahdollisista riskeistä [JUR99], ketterässä kehityksessä pyritään reagoimaan muutoksiin mahdollisimman nopeasti ja joustavasti [NER07]. Perinteisen ohjelmistokehityksen ylläpitämää mielikuvaa ohjelmistoprojektien toimintaympäristöstä on pidetty staattisena ja ennakoitavana [NER07], kun taas ketterässä kehityksessä korostetaan toimintaympäristössä tapahtuvia muutoksia ja ennakoimattomuutta [AUG05, NER07]. Ketterä kehitys ei pyri niinkään ennakoimaan muutoksia kuin vastaamaan niihin nopeasti [NER07]. Perinteisessä ohjelmistokehityksessä tärkeänä pidetty dokumentaatio on ketterässä kehityksessä selvästi pienemmässä roolissa [NER07].

Ketterässä kehityksessä kommunikaatio ja yhteistyö ovat tärkeässä asemassa, ja myös asiakkaiden halutaan osallistuvan kehitykseen aktiivisesti [NER07]. Mahdollisella projektipäälliköllä on ketterässä kehityksessä rooli toiminnan helpottajana ja mahdollistajana (facilitator), ei kontrolloijana [NER07]. Ketterässä kehityksessä johtaminen ei ole vain yksittäisen johtajan asia vaan kollektiivinen prosessi, johon koko tiimi osallistuu [MOE09].

Ketterän ja perinteisen kehityksen eroja havainnollistaa kuva 1.

	Perinteinen kehitys	Ketterä kehitys
Suunnitteluprosessi	Muodollinen, lineaarinen, suunnittelu ja toteutus erillään	Emergentti, iteratiivinen, epämuodollinen
Tavoite	Optimointi	Sopeutuminen, joustaminen, responsiivisuus
Lähestymistapa ongelmiin	Parhaiden keinojen löytäminen hyvin suunnitellun toiminnan kautta	Oppiminen kokeilun ja analyysin kautta, ongelman jatkuva uudelleen muotoilu
Näkemys toimintaympäristöstä	Vakaa, ennustettava	Muutoksille altis, vaikeasti ennustettava
Ominaispiirteitä	Kontrolli ja ohjaus	Yhteistyö ja kommunikaatio
	Konfliktin välttely	Hyväksyy konfliktin ja väittelyn
		Kannustaa kokeiluun, luovuuteen ja tilaisuuksiin tarttumiseen
	Projektipäällikkö kontrolloi muita	Projektipäällikkö on ”tienraivaaja” (facilitator)
	Suunnittelu edeltää toteutusta	Suunnittelua ja toteutusta ei voi erottaa toisistaan

Kuva 1: Perinteisen ja ketterän kehityksen erot (lähde: NER07)

5.1 Projektipäällikön rooli ketterässä kehityksessä

Vuonna 2003 järjestetyn Agile Management – an Oxymoron? -paneelin puhujien alustukset tuovat hyvin esille ketterän kehityksen suhteen projektipäällikön rooliin [AND03]. Koska ketterä kehitys korostaa niin paljon projektitiimin itseohjautuvuutta, projektipäällikön roolia on tarkasteltu kriittisesti ja määritelty uudelleen [AND03]. Ketterää kehitystä käsittelevästä kirjallisuudesta muodostuu kuva, jossa ketterää projektitiimin mukana toimiva projektipäällikkö lähinnä kyselee kysymyksiä tiimin jäseniltä ja varmistaa, että he saavat kaiken, mitä tarvitsevatkin [AND03]. Toisaalta esimerkiksi Extreme Programming (XP) -metodologian luojiin kuuluva Kent Beck antaisi projektipäällikölle vallan keskeyttää projekti tarvittaessa [AND03].

Koko tarve projektipäälliköille ketterässä kehityksessä on kyseenalaistettu [AND03]. Paneelissa puhuneet tietoteknologian ammattilaiset uskovat, että projektipäällikköä tarvitaan ohjelmistoprojektissa esimerkiksi vision ja strategian luontiin, muutoksien toteuttamiseen, projektitiimin ja sidosryhmien väliseen kommunikaatioon, budjetointiin sekä tiimin kouluttamiseen, valmentamiseen ja motivointiin [AND03]. Mainitut työtehtävät eivät juuri eroa perinteisessä ohjelmistokehityksessä korostetuista projektipäällikön tehtävistä. Ketterässä kehityksessä projektipäälliköltä odotetaan perinteisten hallintotehtävien hoitamista, tiimin työn helpottamista ja tiimin motivoimista, mutta hänen tulisi kuitenkin ”pysyä poissa tieltä” ja välttää tiimin kontrolloimista [AND03, ROY05]. Ottaessaan joihinkin ketteriin menetelmiin liittyvän valmentajan (Agile coach) roolin perinteisiin menetelmiin tottuneet projektipäälliköt ovat usein epävarmoja roolistaan [HOD13]. Syntyvä kuva projektipäällikön tehtävistä ketterässä kehityksessä ei ole yksiselitteisen selkeä, ehkä siksi, että ketterän kehityksen piirissä halutaan välttää yksittäisen johtajan roolin korostamista.

5.2 Sopeutuva johtajuus

XP:hen pohjautuvan Agile Project Management (APM) -metodologian kehittäjät Augustine et al. [AUG05] ovat pyrkineet kehittämään projektipäällikölle roolia ketterässä kehityksessä. APM:n koko lähtökohta on ohjelmistoprojektien toimintaympäristön alttius muutoksille [AUG05]. Projektipäälliköt turvautuvat ketterissäkin projekteissa poikkeuksetta perinteisiin tekniikoihin kohdatessaan toimintaympäristön ennakoimattomuuden [AUG05]. Tämä voi johtaa tehottomuuteen, kun organisaatio yrittää reagoida samaan aikaan sekä ympäristön muuttuviin vaatimuksiin että projektinhallintamenetelmiensä vanhentuneisuuteen [AUG05].

Augustine et al. antavat joitain suosituksia toimintaympäristöön reagoimiseen [AUG05]. Tiimien tulisi olla pieniä, jotta kommunikaatio säilyisi sujuvana, ja myös informaation kulun tulisi olla täysin vapaata [AUG05]. Tiimin pienuus antaa paremmat edellytykset myös tiimin demokraattisuudelle [HOD13]. Projektipäällikön tärkeänä tehtävänä on luoda projektista selkeä näkemys, johon yksittäiset kehittäjät voivat tukeutua [AUG05]. Projektipäällikön vastuulla on myös valvoa, että yhdessä sovittuja yksinkertaisia sääntöjä noudatetaan ja poistaa tarvittaessa esteitä näiden sääntöjen

noudattamiselle [AUG05]. Projektipäällikön tulisi johtaa ”kevyellä kosketuksella” (light touch), koska kokeneet ammattilaiset eivät pidä siitä, että heidän toimintaansa yritetään puuttua yksityiskohtia myöten [AUG05]. Augustinen et al. [AUG05] mukaan liiallinen kontrollin tavoittelu ja myös liian jäykkä muodollisten työkalujen soveltaminen johtavat huonoihin lopputuloksiin. Sen sijaan he suosittelevat ratkaisuksi sopeutuvaa johtajuutta, jossa pyritään mukautumaan muutoksiin ja ymmärtämään projektin toimintaympäristöä toisiinsa vaikuttavista osista koostuvana järjestelmänä [AUG05].

5.3 Scrum ja jaettu johtajuus

Moe et al. [MOE09] ovat jatkaneet ketterän projektijohtamisen tarkastelua Augustinen et al. [AUG05] pohjalta. Moe et al. [MOE09] analysoivat johtamista Scrum-menetelmässä, jossa projektipäällikön roolia vastaa suurin piirtein ns. Scrum Masterin rooli. Scrum Master poistaa esteitä projektitiimin tieltä, johtaa puhetta päivittäisissä lyhyissä tiimipalavereissa ja hyväksyttää palavereissa tehdyt päätökset organisaation johdolla [MOE09]. Lisäksi Scrum master vastaa siitä, että kehittäjät noudattavat Scrumia oikeaoppisesti [SCA10]. Scrum Masterin lisäksi scrumissa kuitenkin myös ns. Product Ownerilla on tärkeä rooli [MOE09]. Product Owner on asiakkaan, ylemmän johdon ja scrum-mestarin yhdessä valitsema vastuuhenkilö, joka hallinnoi Scrumissa tärkeää muodollista työkalua, Backlogia [MOE09]. Product Owner osallistuu Backlogin tehtäviin menevän ajan arviointiin, tekee lopullisen päätöksen siitä, mitä Backlogiin tulee ja kehittää Backlogiin merkityistä vaatimuksista valmiin tuotteen ominaisuuksia [MOE09]. Viime kädessä Product Owner on Moen et al. [MOE09] mukaan vastuussa koko projektista. Yksinkertaistaen voidaan sanoa, että projektipäällikön tehtävät on Scrumissa jaettu kahdelle henkilölle, Scrum Masterille ja Product Ownerille. Scrumissa johtajuus ulottuu kuitenkin näiden määriteltyjen roolien ulkopuolelle [MOE08].

Projektipäällikön tai Scrum Masterin tulisi ottaa perinteinen (vertical) johtamisvastuu projektinhallintaan liittyvissä kysymyksissä [MOE09]. Tiimin eri jäsenten tulisi kuitenkin voida johtaa muuta tiimiä projektin eri vaiheissa [MOE09]. Tiimin eri jäsenillä on erityisasiantuntemusta, jota pitää pystyä hyödyntämään ja jakamaan tiimin muille jäsenille [MOE09]. Johtajuus on siis kiertävä tehtävä, ja varsinainen

johtaja määrittyy kulloisenkin haasteen mukaan [MOE09]. Jaettu johtajuus on tärkeää myös ryhmän autonomian kannalta [MOE09].

Jaetun johtamisen soveltaminen tehokkaasti Scrum-tiimissä ei ole välttämättä helppoa [MOE09]. Yleisesti ottaen tiimin johtajan tulisi suunnitella ja välittää muille tiimin tarkoitus ja visio, huolehtia tarvittavista resursseista, valita tiimin jäsenet sekä määritellä tiimin työskentelytavat ja rajoitteet [MOE09]. Scrumissa visio on Moen et al. [MOE09] mukaan Product Ownerin tehtävä, mutta vastuu muista alueista on Scrum Masterilla.

Tiimin jäsenet tulisi, mikäli vain mahdollista, valita heidän johtajaominaisuuksiensa, yhteistyökykynsä ja teknisten taitojensa perusteella [MOE09]. Perinteisesti on ajateltu, että päällekkäisyys (redundancy) tiimin jäsenten taidoissa on tehotonta [KAR10]. Karhatsun et al. [KAR10] mukaan ketterissä tiimeissä tähän kuitenkin pyritään, jotta tiimin jäsenet pystyisivät ymmärtämään toistensa työtä. Tiimin jäsenten liika erikoistuminen myös tekee tiimistä haavoittuvan [KAR10]. Jotta jaettu johtajuus voisi onnistua, tiimissä täytyy Moen et al. [MOE09] mukaan olla oikeat henkilöt ja sen täytyy olla pieni. Koulutus on olennaista jaetun johtajuuden kehittämisessä; yksinäiseen työskentelyyn tottuneet ohjelmoijat eivät välttämättä siirry helposti yhteistyötä korostaviin työtapoihin [MOE09, HOD13].

Ketterää ohjelmistokehitystä käsittelevät artikkelit käsittelevät usein vallan hajautusta teoreettisella tasolla, mutta Scrum-menetelmä tarjoaa käytännön mallin siitä, miten valtaa voi ohjelmistoprojekteissa jakaa ja hajauttaa. Jaettu johtajuus luo pohjan Scrumissa tärkeille itseohjautuville tiimeille [MOE09].

5.4 Itseohjautuvat tiimit

Itseohjautuvien tiimien käsite on ketterää kehitystä vanhempi, mutta itseohjautuvat tiimit ovat olleet ketterän kehityksen alusta asti yksi sen pääperiaatteista [HOD13]. Moe et al. [MOE08] viittaavat itseohjautuvilla (self-managed) tai itseorganisoituvilla (self-organized) tiimeillä selkeisiin ryhmiin, joilla on suuri valta ja vastuu oman työnsä suhteen. Itseohjautuvat tiimit kannustavat tiimin jäseniä työskentelemään enemmän, sitoutumaan tiimin tavoitteisiin ja ottamaan vastuuta sen toiminnasta [MOE08]. Ketterää kehitystä edeltävissä kokeiluissa itseohjautuvien tiimien kannustavan

vaikutuksen on nähty johtuvan ryhmäpaineesta, ei organisaation säännöistä tai normeista [HOD13]. Tiimin jäsenet välittävät enemmän työstään, mikä voi johtaa suurempaan luovuuteen, avuliaisuuteen ja tuottavuuteen [MOE08]. Koska itseohjautuva tiimirakenne antaa tiimin jäsenille päätöksentekovallan yksittäisten pienten ongelmien ja epävarmuuksien kanssa, ongelmanratkaisun tarkkuus ja nopeus voivat parantua [MOE08]. Moe et al. [MOE08] antavat ymmärtää itseohjautuvuuden olevan tärkein periaate innovatiivisen ja yhteistyöhön perustuvan organisaation luomisessa, ja Hoda et al. [HOD13] pitävät itseohjautuvia tiimejä kriittisenä tekijänä ketterien projektien onnistumisessa.

Vaikka itseohjautuvat tiimit toimivatkin itsenäisesti, ne eivät ole täysin vailla ulkopuolista kontrollia [MOE08]. Ylemmän johdon tulisi tasaisin väliajoin arvioida tiimin tulosta, jotta itseohjautuvien tiimien toimintaan kuuluvat epävakaus ja epävarmuus eivät johtaisi kaaokseen [MOE08]. Augustinen et al. [AUG05] tapaan Moe et al. [MOE08] kuitenkin varoittavat liiallisen kontrollin huonosta vaikutuksesta tiimin luovuuteen ja oma-aloitteisuuteen. Jotta itseohjautuvasta tiimirakenteesta olisi hyötyä, tiimin täytyy todella päästä vaikuttamaan päätöksentekoon [MOE08]. Tiimin onnistumisen kannalta on merkityksellistä, missä asioissa tiimi oikeastaan on autonominen [MOE08]. Tiimin autonomia voi toteutua esimerkiksi tiimin oikeudessa määritellä omat tavoitteensa, identiteettinsä, resurssinsa tai henkilöstöasiansa [MOE08].

Moe et al. [MOE08] jakavat autonomian ulkoiseen, sisäiseen ja yksilölliseen autonomiaan. Ulkoinen autonomia viittaa Moella et al. [MOE08] tiimin riippumattomuuteen ylemmän johdon ja muiden tiimin ulkopuolella toimivien tahojen vaikutuksesta. Ulkoinen vaikutus nähdään itseohjautuvien tiimien yhteydessä yleisesti ottaen haitalliseksi, vaikka tiimin ulkopuolisilla vaikutteilla onkin joskus positiivinen vaikutus [MOE08]. Moen et al. [MOE08] ulkoisen autonomian käsite ei ole täysin johdonmukainen tai selkeä, ja voisi olla järkevää viitata vain tiimin autonomiaan ja ulkoiseen vaikutukseen Moen et al. epätarkasti lainaaman Hoeglin ja Parboteeahin [HOE06] tapaan. Sisäisessä eli ryhmätasoisessa autonomiassa tiimi tekee päätökset yhdessä, esimerkiksi äänestämällä tai konsensusta tavoitellen [MOE08]. Ryhmän autonomiassa päätöksenteko ei ole siis keskittynyt yksittäiselle johtajalle,

vaikka tiimi voikin tarvittaessa delegoida päätöksentekovaltaansa yksilöille tai pienemmille ryhmille [MOE08]. On kuitenkin merkille pantavaa, ettei ryhmän autonomia myöskään välttämättä merkitse, että yksilöllä olisi suurta vapautta vaikuttaa omiin työtehtäviinsä [MOE08]. Yksilön vapaus vaikuttaa omaan työhönsä on yksilön autonomiaa, joka voi olla ristiriidassa ryhmän autonomian kanssa [MOE08]. Vaikka itseohjautuvan tiimin on arvioitu tarvitsevan niin ryhmän kuin yksilönkin autonomiaakin, ryhmän ja yksilön autonomia voivat ristiriidassa ollessaan heikentää tiimin yhtenäisyyttä, mikä vaikuttaa myös tiimin tehokkuuteen [MOE08]. Tämän vuoksi itsenäisesti toimivat, omaan työnsä keskittyvät ammattilaiset saattavat haitata monitaitoisen (cross-functional) tiimin toimintaa [MOE08].

Moe et al. [MOE08] tutkivat kvalitatiivisilla metoideilla yhden Scrum-tiimin toimintaa, missä edellä esitetyt uhkakuvat pitkälti toteutuivat. Product Owner ei toiminut samalla paikkakunnalla kuin muu tiimi, minkä vuoksi hänen osallistumisensa jäi ongelmallisen vähäiseksi [MOE08]. Esimerkiksi koko tiimin toimintaa periaatteessa tarkasti määrittävä Backlog jäi tarkoitettua vähäisemmälle huomiolle [MOE08]. Scrum Master kannusti tehokkuuteen vedoten kehittäjiä työskentelemään jokainen oman moduulinsa parissa: tämä johti tilanteeseen, jossa jokainen kehittäjä teki omasta moduulistaan henkilökohtaisen ”valtakuntansa” [MOE08]. Kehittäjät keskittyivät omaan moduuliinsa välittämättä muun projektin etenemisestä ja tekivät paikoin työtään täysin itsenäisesti, ohittaen Backlogiin kirjatut osatavoitteet ja Scrum Masterin mielipiteen [MOE08]. Yksilön autonomia ohitti siis ryhmän autonomian. Scrum Master ei enää luottanut kehittäjiin ja alkoi antaa aikaisempaa suorempia ohjeita suoritettavista työtehtävistä [MOE08]. Tämä heikensi sekä yksilöllistä että ryhmän autonomiaa [MOE08]. Scrum toi tiimille kuitenkin perinteistä projektimallia enemmän suojaa ulkoisilta paineilta, vaikka tiimin jäsenet kohtasivat edelleen paljon projektiin liittymättömiä vaatimuksia [MOE08].

Tiimin vaikeuksille voi helposti löytää eri selityksiä. Tiimissä korostui yksilön autonomia, joka johti ryhmän autonomian heikkenemiseen [MOE08]. Scrumissa korostettu monitaitoisuus ei toteutunut kehittäjien erikoistuessa omiin moduuleihinsa. Toisaalta yksilön autonomiaa on pidetty tärkeänä yksilön motivaation ja tyytyväisyyden kannalta [MOE08]. Scrumin puolestapuhujat ovat suhtautuneet kielteisesti

Scrumin toimintamallien muokkaamiseen organisaation vaatimusten tai ongelmien vuoksi, uskoen siitä seuraavan ongelmia [HEI13]. Käytännön työ kuitenkin asettaa usein rajoitteita Scrumin kaikkien sääntöjen soveltamiselle [HEI13, MOE09]. Joe Blotner kritisoi ketterää kehitystä siitä, että koko työprosessia ajatellaan vain ohjelmoijan näkökulmasta, eikä muulle organisaatiolle tarjota tekniikoita, joita soveltaa [AND03].

5.5 Roolit ketterissä tiimeissä

Hoda et al. [HOD13] ovat eritelleet ketterien, itseohjautuvien tiimien eri rooleja. Ketterissä tiimeissä ohjelmistokehittäjän tai testaajan kaltaiset roolit eivät ole tarkasti rajattuja, mutta tiimin jäsenet omaksuvat oma-aloitteisesti epävirallisia ja väliaikaisia rooleja vastatakseen ketterissä ohjelmistoprojekteissa kohdattaviin haasteisiin [HOD13]. Ketterää tiimiä valmentava henkilö (Agile coach) saattaa toimia useissakin alla käsitellyissä rooleissa [HOD13]. Suurin osa alla käsitellyistä rooleista on myös luonteva osa perinteisen projektipäällikön tehtävää.

Koska ketterien toimintatapojen omaksuminen voi olla vaikeaa, projektin aluksi tarvitaan usein kokenutta kehittäjää toimimaan *mentorin* (opettaja, neuvonantaja) roolissa [HOD13]. Mentori opastaa ja ohjaa tiimiä ketterissä menetelmissä, puuttuen tarvittaessa myös väärinkäsityksiin ja negatiiviseen ilmapiiriin [HOD13]. Scrum Master ja XP-menetelmän XP-valmentaja (XP Coach) toimivat juuri mentorin roolissa [HOD13].

Koordinoijan tehtävänä on pitää yhteyttä asiakkaisiin ja tehdä yhteistyötä heidän kanssaan [HOD13]. Koordinoijan rooli on erityisen tärkeä, jos asiakkaan edustaja ei ole perehtynyt ketteriin menetelmiin tai ei anna tarpeeksi palautetta tiimille [HOD13]. Koordinoijan yksi tehtävä on ottaa selvää asiakkaan vaatimuksista, joiden on jatkuvasti oltava selviä tiimille [HOD13]. Kokeneissa ketterissä tiimeissä suurin osa tiimin jäsenistä on valmis ottamaan vastaan koordinoijan roolin ja kommunikoimaan suoraan asiakkaan kanssa [HOD13]. Asiakkaan kanssa kommunikoi myös *tulkki*, jonka tehtävänä on ”kääntää” projektitiimin tekniset käsitteet asiakkaan ymmärtämäksi arkikieleksi ja päinvastoin [HOD13]. Joskus koordinoijan ja tulkin roolit ovat samalla henkilöllä [HOD13].

Puolestapuhujan (Champion) tehtävä on puhua ketterien menetelmien puolesta

organisaation ylemmälle johdolle [HOD13]. Ketterä tiimi ei voi toimia ilman organisaation tukea, ja puolestapuhujan tulee ymmärtää taloudellisia näkökulmia voidakseen vakuuttaa organisaation johdon ketterän kehityksen eduista [HOD13]. Samankaltainen rooli on *markkinoijalla* (Promoter), jonka tehtävä on puhua ketterän kehityksen eduista asiakkaalle [HOD13].

Terminaattori on ehkä kiistanalaisin ketterän kehityksen rooleista [HOD13]. Terminaattorin tehtävä on tunnistaa tiimistä sille haitaksi olevia jäseniä ja vedota ylempään johtoon, jotta nämä jäsenet poistettaisiin tiimistä [HOD13]. Ongelmia aiheuttavat erityisesti ne tiimin jäsenet, jotka eivät sopeudu ketteriin työtapoihin [HOD13]. He työskentelevät yksin, eivät jousta ja ovat haluttomia kommunikoimaan muun tiimin kanssa [HOD13]. Myös liian dogmaattiset ja joustamattomat ketterien menetelmien kannattajat voivat muodostua ongelmaksi [HOD13]. Terminaattori on mukana myös valitsemassa jäseniä projektitiimiin [HOD13].

5.6 Muutosjohtajat ja ad hoc -johtajuus

Dubinsky ja Hazzan [DUB10] ovat analysoineet johtajuutta ketterissä projekteissa ja nostaneet esiin nk. muutosjohtajat (Change Leader) ja heille tunnusomaisen johtamistyylin, jota Dubinsky ja Hazzan kutsuvat ad-hoc -johtajuudeksi. Muutosjohtajat ovat yksilöitä, jotka tuovat organisaatioonsa ketteriä menetelmiä [DUB10]. He ymmärtävät niin projektitiimin, asiakkaan kuin organisaation johdonkin näkökulmaa [DUB10]. Muutosjohtajat välittävät tietoa ja viestivät kaikkien eri osapuolten välillä, ja pystyvät tarvittaessa johtamaan siirtymäprosessia perinteisistä menetelmistä ketterään kehitykseen [DUB10]. Muutosjohtaja pystyy tarvittaessa siirtämään toimintansa painopistettä niin projektitiimin, asiakkaan kuin organisaation johdonkin suuntaan. Muutosjohtajan roolissa yhdistyvät lukuisat Hodan et al. [HOD13] erittelemät roolit: muutosjohtaja toimii mentorin roolissa auttaessaan tiimiä, markkinoijan roolissa kertoessaan ketteristä menetelmistä asiakkaalle ja puolestapuhujan roolissa puolustaessaan ketteriä menetelmiä organisaation johdolle. Lisäksi hän toimii koordinoijana ja tulkkina viestiessään asiakkaalle. Puutteena Dubinskyn ja Hazzanin artikkelissa [DUB10] voi pitää sitä, etteivät he käsittele tarkasti, kuinka paljon muutosjohtaja on vastuussa varsinaisesta asiakkaalle toteutettavasta projektista ketteriin

menetelmiin siirtymisen lisäksi.

6 Yhteenveto

Ohjelmistoprojekteja ovat pitkään leimanneet epäonnistumiset, joiden on katsottu johtuvan ohjelmistokehityksen haasteista sekä epäonnistuneesta johtamisesta. Perinteisessä projektijohtamisessa ja projektinhallinnassa on korostettu taitavan projektipäällikön roolia ja huolellista ennakkosuunnittelua. Projektipäällikkö voi perinteisen mallin ihanteen mukaan muodollisia projektinhallintatyökaluja apunaan käyttäen varautua projektin aikana ilmeneviin ongelmiin ja saattaa projektin onnistuneesti loppuun. Projektipäällikön roolin keskeisyydestä huolimatta perinteinen projektijohtaminen ei välttämättä ole autoritaarista ja kontrolloivaa, vaikka tietyissä olosuhteissa myös tällainen johtamistyyli voi tuottaa hyviä tuloksia.

Perinteisiin menetelmiin verrattuna ketterät menetelmät hajoittivat johtamisvastuuta välillä hyvinkin radikaalisti projektitiimin eri jäsenille, ja ottivat myös asiakkaan mukaan kehitysprosessiin. Ketterissä menetelmissä ei yleensä keskitetä johtajan roolia vain yhdelle henkilölle, vaan kaikkien tiimien jäsenten oletetaan osallistuvan päätöksentekoon. Tiimin eri jäsenet omaksuvat tilanteen ja erityisosaamisensa mukaan erilaisia rooleja, joihin sisältyy myös perinteisesti projektipäällikölle kuulunutta johtamisvastuuta. Näihin rooleihin on liittynyt paljon myös ketteriin menetelmiin valmentamista ja niiden puolesta puhumista.

Ketteriin menetelmiin kuuluvat itseohjautuvat tiimit ja jaettu johtajuus eivät tarkoita, että tiimin jäsenet voisivat toimia täysin itsenäisesti, ottamatta huomioon muiden tiimin jäsenten näkemyksiä. Vaikka itseohjautuvissa tiimeissä yksilön autonomialla onkin merkitystä, ryhmän päätösvalta menee lopulta yksilön autonomian edelle.

Ketterät menetelmät ovat tuoneet uusia näkemyksiä ohjelmistoprojektien johtamiseen uudenlaista ajattelua. Yhteydenpito asiakkaaseen ja organisaation johtoon, tiimin valmentaminen ja motivointi, konfliktien purkaminen sekä aikataulun, vaatimusten ja resurssien hallinta ovat kuitenkin ketterissäkin menetelmissä projektijohtamisen ydintä.

Lähteet

AND03 Anderson, L., Alleman, G. B., Beck, K., Blotner, J., Cunningham, W., Poppendieck, M., & Wirfs-Brock, R. (2003, October). Agile management-an oxymoron?: who needs managers anyway?. In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications* (pp. 275-277). ACM.

AUG05 Augustine, S., Payne, B., Sencindiver, F., & Woodcock, S. (2005). Agile project management: steering from the edges. *Communications of the ACM*, 48(12), 85-89.

DUB10 Dubinsky, Y., & Hazzan, O. (2010, May). Ad-hoc leadership in agile software development environments. In *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering* (pp. 32-38). ACM.

ELS01 El-Sabaa, S. (2001). The skills and career path of an effective project manager. *International journal of project management*, 19(1), 1-7.

FAR06 Faraj, S., Sambamurthy, V. (2006). Leadership of information systems development projects. *Engineering Management, IEEE Transactions on*, 53(2), 238-249.

HEI13 Heikkilä, V. T., Paasivaara, M., & Lassenius, C. (2013, October). Scrum-But, But Does it Matter? A Mixed-Method Study of the Planning Process of a Multi-team Scrum Organization. In *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on* (pp. 85-94). IEEE.

HOD10 Hoda, R., Kruchten, P., Noble, J., & Marshall, S. (2010, October). Agility in context. In *ACM Sigplan Notices* (Vol. 45, No. 10, pp. 74-88). ACM.

HOD13 Hoda, R., Noble, J., & Marshall, S. (2013). Self-organizing roles on agile software development teams. *Software Engineering, IEEE Transactions on*, 39(3), 422-444.

HOE06 Hoegl, M., & Parboteeah, P. (2006). Autonomy and teamwork in innovative projects. *Human Resource Management*, 45(1), 67-79.

JUR99 Jurison, J., Software project management: the manager's view. *Communi-*

cations of the AIS, vol. 2, issue 3es (November 1999), Article No. 2.

KAR10 Karhatsu, H., Ikonen, M., Kettunen, P., Fagerholm, F., & Abrahamsson, P. (2010, October). Building blocks for self-organizing software development teams a framework model and empirical pilot study. In *Software Technology and Engineering (ICSTE), 2010 2nd International Conference on* (Vol. 1, pp. V1-297). IEEE.

MOE08 Moe, N. B., Dingsoyr, T., & Dyba, T. (2008, March). Understanding self-organizing teams in agile software development. In *Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on* (pp. 76-85). IEEE.

MOE09 Moe, N. B., Dingsyr, T., & Kvangardsnes, O. (2009, January). Understanding shared leadership in Agile development: A case study. In *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on* (pp. 1-10). IEEE.

MON12 Monteiro, P., Borges, P., Machado, R. J., & Ribeiro, P. (2012, June). A reduced set of RUP roles to small software development teams. In *Software and System Process (ICSSP), 2012 International Conference on* (pp. 190-199). IEEE.

NER07 Nerur, S., & Balijepally, V. (2007). Theoretical reflections on agile development methodologies. *Communications of the ACM*, 50(3), 79-83.

PET03 Peters, L., Educating Software Engineering Managers. *Journal of the ACM*, vol. 51, issue 5 (September 2004), pages 780-799.

PIN11 Pinkowska, M., Lent, B., Keretho, S., Process based identification of software project manager soft skills, *2011 Eighth International Joint Conference on Computer Science and Software Engineering (JCSSE)*, (May 2011), pages 343-348.

RID09 Ridchenko, O., Specifics of entry-level IT project managers in Eastern Europe. *2009 5th Central and Eastern European Software Engineering Conference in Russia (CEE-SECR)*, (October 1999), pages 243-248.

ROY05 Royce, W., Successful software management style: steering and balance. *IEEE Software*, vol. 22, issue 5 (September-October 2005), pages 40-47.

SCA10 Scharff, C., & Verma, R. (2010, May). Scrum to support mobile application development projects in a just-in-time learning context. In P*roceedings of*

the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering (pp. 25-31). ACM.

WEI99 Weinberg, G. M. (1999). Egoless Programming. *Software, IEEE*, 16(1), 118-120.