

Julius Virolainen

KETTERÄN OHJELMISTOKEHITYKSEN KEHITTÄMINEN JA MITTAAMINEN

Kandidaatintyö
Johtamisen ja talouden tiedekunta
Tarkastaja: Laura Valtonen
Toukokuu 2023

TIIVISTELMÄ

Julius Virolainen: Ketterän ohjelmistokehityksen kehittäminen ja mittaaminen
Improving and Measuring Agile Software Development
Kandidaatintyö
Tampereen yliopisto
Tuotantotalouden tutkinto-ohjelma
Toukokuu 2023

Erilaisista ohjelmistoista on vuosien saatossa tullut merkittävä osa monenlaista yritystoimintaa. Täten myös ohjelmistokehityksen merkitys on kasvanut ja ohjelmistokehitykseen kohdistuvat vaatimukset ovat tiukentuneet. Vastatakseen näihin vaatimuksiin monissa ohjelmistotalouden organisaatioissa on otettu käyttöön niin kutsuttuja ketteriä menetelmiä. Näin onkin saavutettu parannuksia mm. joustavuuteen sekä asiakastyytyvyyteen. Alalla on kuitenkin tunnistettu tarve uusille tavoille saavuttaa kilpailuetua. Tämän kandidaatintyön tarkoituksena on tarkastella ohjelmistokehityksen systemaattisen kehittämisen hyödyntämistä ketterien menetelmien rinnalla. Tarkastelussa otetaan erityisesti suorituksen mittaamiseen keskittyvä näkökulma. Ohjelmistokehityksen systemaattinen kehittäminen on ei ole ilmiönä uusi, mutta sitä on ei ole yleensä pidetty kovin yhteensopivana ketterien menetelmien kanssa.

Työ on toteutettu kirjallisuuskatsauksena. Lähdemateriaalina on käytetty relevantteja alan tieteellisiä julkaisuja. Työn rakenne koostuu viidestä eri pääluvusta. Ensimmäisessä pääluvussa esitellään tutkimuksen aihe, käytetyt tutkimusmenetelmät ja työn rakenne. Toisessa pääluvussa esitellään ohjelmistoprosessi ja sen kehittämisen taustaa. Lisäksi esitellään ketterät menetelmät. Kolmannessa pääluvussa tarkastellaan johdon ohjausjärjestelmien ja suorituksen mittaamisen merkityksiä organisaatioissa. Neljännessä pääluvussa esitellään tutkimuksen tulokset ja vastataan tutkimuskysymyksiin. Viides pääluku sisältää tutkimuksen keskeiset päätelmät ja pohdintaa tutkimuksen rajoitteista sekä mahdollisista jatkotutkimusmahdollisuuksista.

Ketterät menetelmät ovat joukko erilaisia toimintatapoja, joilla perustuvat tiettyihin hyvänä pidettyihin ohjelmistokehityksen periaatteisiin. Suorituksen mittaaminen on rahamääräisen ja eiramääräisen mittaamisen muodostama johdon laskentatoimen osa-alue. Ketterälle ohjelmistokehitykselle tyypillistä on joustavuus, kun taas suorituksen mittaamisella koitetaan usein saavuttaa ennakoitavuutta ja toistettavuutta yrityksen prosesseihin. Systemaattiselle ohjelmistoprosessin kehittämiseksi on tyypillistä ohjaaminen ja byrokratia.

Tämä kandidaatintyö osoittaa, että systemaattista kehittämistä voidaan soveltaa myös ketterään ohjelmistokehitykseen. Työ tuo esiin, kuinka systemaattiseen kehittämiseen tiiviisti liittyvä suorituksen mittaaminen voidaan kokea erinäisten tekijöiden mukaan joko ohjelmistokehitystä tukevana tai rajoittavana. Lisäksi työssä on tunnistettu tarve perinteisiä systemaattisia ohjelmistoprosessin kehitysmalleja joustavammille ja kevyemmille systemaattisen kehittämisen malleille ja mittaristoille, jotka ottavat paremmin huomioon ketterälle ohjelmistokehitykselle tärkeitä tekijöitä.

Avainsanat: johdon laskentatoimi, suorituksen mittaus, ohjelmistotuotanto, ketterä ohjelmistokehitys

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck –ohjelmalla.

ALKUSANAT

Kulunut kevät on ollut kiirettä täynnä ja osittain tämän takia työni eteneminen on ollut lievästi sanottuna epävarmaa. Nyt kandidaatintyöni on kuitenkin vihdoinkin valmis.

Haluan kiittää professori Teemu Lainetta, työn ohjaajaa Laura Valtosta sekä kaikkia muita työni parissa auttaneita hyvästä ohjauksesta ja kärsivällisyydestä. Lisäksi haluan kiittää perhettäni ja ystäviäni tuesta kirjoitusprosessin aikana.

Tampereella, 23.5.2023

Julius Virolainen

SISÄLLYSLUETTELO

1. JOHDANTO	1
1.1 Tutkimuksen tavoitteet.....	1
1.2 Tutkimusmetodologia.....	2
1.3 Työn rakenne	3
2. OHJELMISTOPROSESSI JA KETTERÄT MENETELMÄT	4
2.1 Ohjelmistoprosessi ja sen kehittäminen.....	4
2.2 Ketterät menetelmät	4
3. SUORITUKSEN MITTAAMINEN	8
3.1 Mahdollistavat ja pakottavat johdon ohjausjärjestelmät	8
3.2 Suorituksen mittaaminen yleisesti	9
3.3 Suorituksen mittaaminen projektien johtamisessa	10
4. KETTERÄN OHJELMISTOPROSESSIN KEHITTÄMINEN JA MITTAAMINEN ...	12
4.1 Ohjelmistoprosessin kehittämisen ja mittaamisen taustaa	12
4.2 Ketterän ohjelmistoprosessin kehittämisen erikoispiirteet	13
4.3 Ketterän ohjelmistoprosessin mittaamisen erikoispiirteet	15
5. PÄÄTELMÄT	17
5.1 Tutkimuksen arviointi ja jatkotutkimusmahdollisuudet.....	18
LÄHTEET	19

1. JOHDANTO

Ohjelmistoista on tullut merkittävä osa liiketoimintaa monella eri alalla. Ohjelmistojen ja ohjelmistotuotannon merkityksen kasvaessa siihen kohdistuvat vaatimukset mm. joustavuuden, asiakas- ja arvolähtöisyyden, nopeampien julkaisusykliden sekä laadun suhteen ovat kasvaneet. (Küpper et al., 2019) Suurin osa ohjelmistoalan yrityksistä onkin ottanut käyttöön ns. ketteriä (engl. agile) menetelmiä (Coleman & O'Connor, 2008; Küpper et al., 2019) ja hyvällä menestyksellä (Serrador & Pinto, 2015). Ihmislähtöiset ketterät menetelmät keskittyvät asiakkaalle tärkeimpien ominaisuuksien nopeaan ja joustavaan toimittamiseen. Tämä tapahtuu kehittämällä ohjelmaa iteratiivisesti pienemmissä pyrkäyksissä. (Highsmith & Cockburn, 2001)

Ketterien menetelmien käytön lisäksi ohjelmistoprosessin systemaattinen kehittäminen on yleinen tapa saavuttaa kilpailuetua. Ohjelmistoprosessin kehittämällä tavoitellaan mm. laadukkaampia ja luotettavampia ohjelmistoja, parempaa asiakastytyväisyyttä sekä parempaa tuottoa sijoitetulle pääomalle. (Lee et al., 2016) Systemaattisessa lähestymistavassa ohjelmistokehitystä tarkastellaan prosessina, jota voidaan jatkuvasti kehittää mittaamalla, analysoimalla ja tekemällä muutoksia (Humphrey, 1988).

Mittaaminen on siis olennainen osa prosessien kehittämistä. Se toimii tukena prosessien ymmärtämiselle, arvioimiselle ja johtamiselle. Sillä on ohjaava vaikutus organisaation kehitysprosesseille ja -projekteille sekä niiden lopputuloksille. Mittaaminen mahdollistaa prosessien parantamisen sekä ennustamisen ja mahdollistaa täten paremman päätöksenteon. (Meidan et al., 2018). Toisaalta mittamisen kaltaiset ohjauskeinot voidaan kokea joustavuutta vaativissa projekteissa rajoittavana (Laine et al., 2020). Tämä on monissa tilanteissa voinut johtaa systemaattisen kehittämisen vieroksumiseen (Küpper et al., 2019).

1.1 Tutkimuksen tavoitteet

Tutkimuksen tavoitteena on tutkia ketterän ohjelmistokehityksen systemaattista kehittämistä ja siihen liittyviä haasteita ja menestystekijöitä. Aihetta on tarkasteltu erityisesti johdon ohjauksen ja mittamisen näkökulmasta. Täten pyritään luomaan ymmärrystä

siitä, mitä ketteriä menetelmiä hyödyntävän organisaation tulisi tehdä kehittääkseen systemaattisesti ohjelmistokehitystään ja saavuttaakseen tätä kautta kilpailuetua. Työssä vastataan seuraaviin tutkimuskysymyksiin:

1. Miten ohjelmistoprosesseja voidaan systemaattisesti kehittää ja mikä on mittauksen merkitys niiden kehittämisessä?
2. Mitä tulee ottaa huomioon etenkin ketterän ohjelmistoprosessin systemaattisessa kehittämisessä ja mittaamisessa?

1.2 Tutkimusmetodologia

Työ on toteutettu kirjallisuuskatsauksena ja työn aiheeseen liittyvää kirjallisuutta on haettu Scopus-tietokannasta. Lähteiden luotettavuuden arvioinnin apuna on hyödynnetty Julkaisufoorumi-palvelua. Taulukossa 1 on esitetty, miten työn tulososaan valikoituneet julkaisut löytyivät. Kuten kaaviosta 1 näkyikin, myös hakujen ulkopuolelta on otettu kirjallisuutta. Niin kutsutun helmenkasvatuksen avulla on tunnistettu yksi alan perusteoksista. Lisäksi tulososiossa käytettyä kirjallisuutta löytyi työn ketteristä menetelmistä kertovaa teoriaosuutta tehdessä.

Taulukko 1 Tulososassa käytetty kirjallisuus

Hakutapa	Hakulauseke	Tuloksia	Valitut
Scopus-haku	"maturity model"	5694	2
Scopus-haku	"software process improvement" OR SPI	19685	4
Scopus-haku	("software process improvement" OR SPI OR "software process") AND (control OR metric* OR performance OR measure* OR indicator OR kpi OR ppi)	9514	4
Scopus-haku	agile AND (control OR metric* OR performance OR measure* OR indicator OR kpi OR ppi)	15472	4
Scopus-haku	("software process improvement" OR SPI) AND agile	293	3
Helmenkasvatus (esim. Bititci et al., 2015)			1
Tunnistettu teoriaosuutta tehdessä			2

Tutkimuksessa käsiteltyjä aihepiirejä on alan tutkimuksessa käsitelty runsaasti ja monista eri näkökulmista. Tutkimukseen onkin valikoitunut tutkimuskysymykseen vastaamisen kannalta relevantein kirjallisuus. Koska ohjelmistoala kehittyy nopeasti, on kirjallisuuden tuoreudelle annettu merkittävä painoarvo relevanttiutta arvioidessa. Artikkeleiden lisäksi myös konferenssijulkaisuja on kelpuutettu osaksi tutkimusta. Perustelu näiden hyödyntämiselle työssä on näiden laaja käyttö alan muussa tutkimuksessa.

1.3 Työn rakenne

Tutkimuksen toisessa pääluvussa esitellään ohjelmistoprosessi ja sen kehittäminen sekä ketterät menetelmät. Kolmannessa pääluvussa esitellään johdon ohjauksen mahdollistavaan ja estävään ohjaukseen jakava viitekehys sekä muita tutkimuksen kannalta oleellisia näkökulmia suorituksen mittaamiseen. Neljäs pääluku sisältää työn tulososion. Siinä esitellään lisää ohjelmistoprosessin systemaattista kehittämistä sekä sovelletaan aiemmin esiteltyä johdon ohjaukseen ja suorituksen mittaamisen liittyvää viitekehystä ketteriä menetelmiä hyödyntävän ohjelmistoprosessin kehittämisen ja mittaamisen tarkasteluun. Viidennessä pääluvussa esitellään keskeisimmät päätelmät sekä pohditaan tutkimuksen onnistumista, tutkimukseen liittyviä rajoitteita ja mahdollisia jatkotutkimusmahdollisuuksia.

2. OHJELMISTOPROSESSI JA KETTERÄT MENETELMÄT

Tässä pääluvussa avataan lisää modernin ohjelmistokehityksen kontekstia ja pohjustetaan käsittelylukua. Aluksi esitellään lyhyesti ohjelmistoprosessin käsite ja sen kehittämisen kaksi lähestymistapaa. Pääluvun toisessa alaluvussa esitellään ketterät menetelmät ja niiden takana olevat periaatteet yleisellä tasolla. Esille tuodaan miten ketterät menetelmät auttavat vastaamaan modernin ohjelmistokehityksen haasteisiin.

2.1 Ohjelmistoprosessi ja sen kehittäminen

Ohjelmistoprosessilla tarkoitetaan niitä aktiviteetteja, metodeja, käytäntöjä sekä muunnoksia, joita tehdään uusien ohjelmistotuotteiden aikaansaamiseksi tai vanhojen ylläpitämiseksi. (Paulk et al., 1993). Jokaiseen ohjelmistoprosessiin sisältyy jossain muodossa ohjelmiston määrittely, kehittäminen, kelpuutus sekä ohjelmiston mukauttaminen asiakastarpeisiin (Sommerville, 2016 s.44).

Ohjelmistoprosessin kehittämiseen ja muuttamiseen voidaan tunnistaa kaksi eri näkökulmaa, joista toinen on tässä pääluvussa myöhemmin esiteltävä ketterien menetelmien käyttöönotto. Vaihtoehtoinen tapa kehittää ohjelmistoprosessia keskittyy prosessin kypsyyden systemaattiseen parantamiseen. Prosessin kypsyyteen voidaan vaikuttaa esimerkiksi ottamalla käyttöön uusia tekniikkaan ja johtamiseen liittyviä käytäntöjä. (Sommerville, 2016 s.66)

Jatkossa tässä työssä ohjelmistoprosessin kehittämisellä tarkoitetaan juuri tätä jälkimmäistä, systemaattisempaa keinoa. Tässä työssä ei keskitytä yksittäisiin ohjelmistoprosessin kehittämisen standardeihin kuten CMM- ja ISO-standardeihin vaan näiden takana oleviin periaatteisiin. Lisäksi työssä käytetään englannin kielen termistä software process improvement tulevaa lyhennettä SPI.

2.2 Ketterät menetelmät

Alan tutkimuksen mukaan ketterien menetelmien hyödyntämisen ja ohjelmistokehityksen onnistumisen välillä on havaittu yhteys (Serrador & Pinto, 2015). Suurin osa ohjelmistotalan organisaatioista on ottanut käyttöön ketteriä menetelmiä (Coleman & O'Connor, 2008; Küpper et al., 2019). Ketterät menetelmät tukevat onnistumista etenkin dynaamisissa projekteissa (Butler et al., 2020). Toisaalta ketterien menetelmien suunnitelmallisuuden puute koetaan joissain tilanteissa haasteena (Agrawal et al., 2016).

Ketterillä menetelmillä ei tarkoiteta mitään yksittäistä menetelmää, vaan se koostuu joukosta erilaisia menetelmiä, jotka jakavat samat perimmäiset periaatteet. Nämä periaatteet on esitetty niin kutsutussa ketterässä manifestissa (engl. Agile Manifesto). (Dingsøyr et al., 2012) Beck et al. (2001) tekemän ketterän manifestin periaatteet vapaasti suomennettuna ovat:

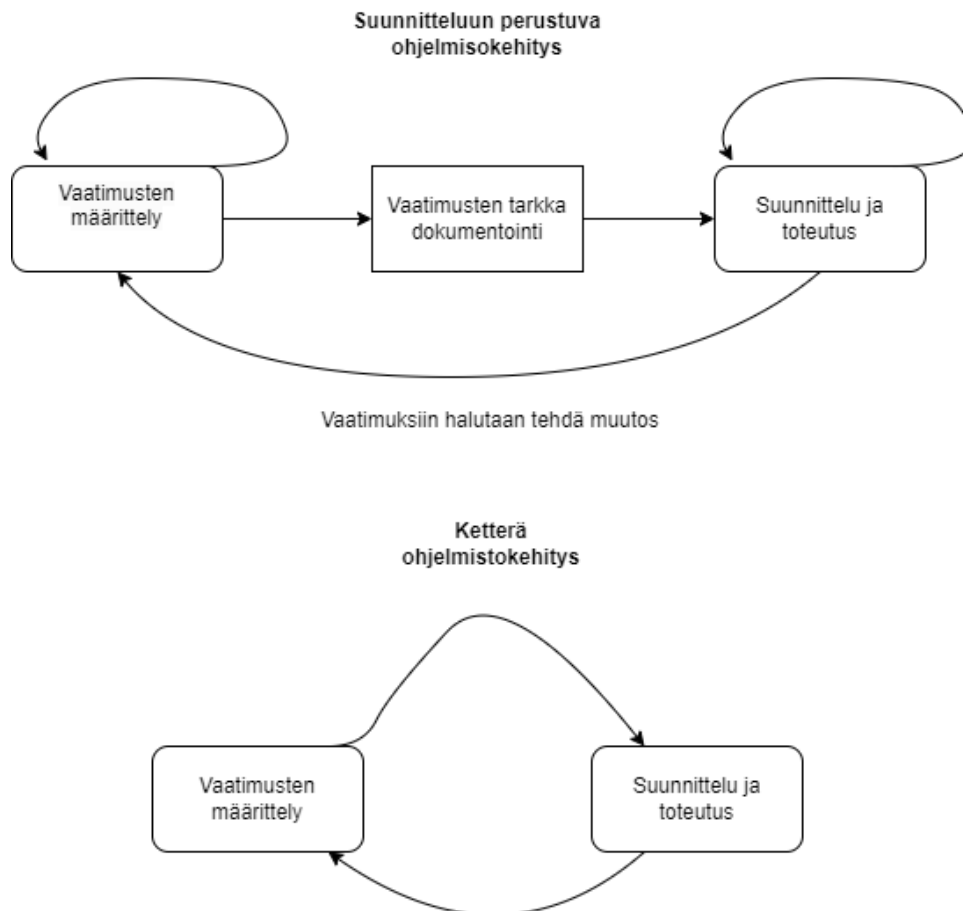
- Yksilöt ja vuorovaikutus ovat tärkeämpiä kuin prosessit ja työkalut
- Toimiva ohjelmisto on tärkeämpi kuin kattava dokumentaatio
- Yhteistyö asiakkaan kanssa on tärkeämpää kuin sopimusneuvottelut
- Muutokseen vastaaminen on tärkeämpää kuin suunnitelman noudattaminen

Käytännössä ketterässä ohjelmistokehityksessä ei noudateta tarkasti mitään tiettyä menetelmää (Coleman & O'Connor, 2008; Kuhmann et al., 2022; Küpper et al., 2019). Ketterään ajatustapaan kuuluukin ettei yksittäisiä menetelmiä ole tarkoitukseen noudateta tarkasti, vaan jokainen organisaatio voi muunnella niitä paremmin omaan käyttöön sopiviksi (Highsmith & Cockburn, 2001). Samoin kuin ohjelmistoprosessin kehittämistä myös ketteryyttä käsitellään työssä hyvin yleisellä tasolla sen käsitteellisen laajuuden sekä käytännön toteutuksien monimuotoisuuden takia.

Ketterissä menetelmissä ohjelmistoa toteutetaan lyhyissä, esimerkiksi noin 2–6 viikkoa pitkissä iteratiivisissa sykleissä. Ennen jokaista iteraatiota päätetään siinä kehitettävät ominaisuudet. Ominaisuuksien priorisointi on dynaamista, eli ominaisuuksien tärkeysjärjestys voi vaihtua iteraation lopuksi. Muutoksia prioriteetteihin ei kuitenkaan koskaan tehdä kesken iteraation. (Highsmith & Cockburn, 2001) Ketterissä menetelmissä vaatimuksia ei siis kerätä vain projektin alussa, vaan jokaisen iteraation välissä. Lisäksi on tyypillistä, että vaatimukset ovat luonteeltaan toiminnallisia. Tämä tapa toimia mahdollistaa paremman joustavuuden, mutta lisäksi tuo mukanaan joitain riskejä. Voi esimerkiksi olla mahdollista, että liika keskittyminen toiminnallisiin vaatimuksiin johtaa siihen, että ei-toiminnalliset vaatimukset, kuten ohjelmiston arkkitehtuuri, eivät saa riittävästi huomiota. Arkkitehtuuri lyödään usein lukkoon projektin varhaisessa vaiheessa ja ketterille menetelmille tyypillinen vaatimusten määrittely ei välttämättä ole aina myöskään riittävän pitkäkatseista. (Ramesh et al., 2010)

Kuvassa 1 on havainnollistettu suunnitteluun pohjautuvan ohjelmistoprosessien ja ketterän kehityksen eroja. Ketterässä kehityksessä ohjelmistoa kehitetään ja vaatimuksia tarkennetaan iteraatio kerrallaan. Perinteisessä suunnitteluun pohjautuvassa mallissa tehdään aluksi vaatimusten määrittely kokonaan loppuun. Tästä siirrytään suunnitteluun ja

toteutukseen, josta vaatimusten määrittelyyn palataan vain, jos ilmenee suunnitteleman tarve muutokselle. Tämän jälkeen joudutaan tekemään lisää vaatimusten määrittelyä ja muuttamaan dokumentaatiota



Kuva 1 Suunnitteluun perustuva ohjelmistokehitys ja ketterä ohjelmistokehitys (mukaan Somerville, 2014 s.74)

Ketterissä menetelmissä pyritään siihen, että jokaisen iteraation lopputuloksena syntyy toimiva ohjelma. Näin pysytään kartalla siitä, mitä kaikkea on oikeasti saatu aikaan ja nähdään tehtyjen päätösten seuraukset nopeasti. (Highsmith & Cockburn, 2001) Valmistu ohjelmaa voidaan esitellä eri sidosryhmille ja näin tuetaan tiimin oppimista. Lisäksi saadaan tärkeää palautetta vaatimusten määrittelyyn. Näin kyetään sopeuttamaan tuotetta ympäristön muutoksiin. (Nerur et al., 2005)

Ketterissä menetelmissä keskitytään asiakasarvon toimittamiseen paremman vuorovaikutuksen kautta. Asiakkaiden ja rahoittajien välistä keskustelua lisäämällä voidaan aiempaa paremmin ratkaista vaikeuksia, säätää prioriteetteja ja tarkastella vaihtoehtoisia etenemistapoja. Eri sidosryhmät tuodaan osaksi kehitysprosessia. Tausta-ajatus on, että läheisempi yhteistyö eri sidosryhmien kanssa voi johtaa parempiin ja edullisempiin ohjelmistoihin ja mahdollistaa ripeämmät suunnanmuutokset. (Highsmith & Cockburn,

2001). Asiakkaan osallistaminen onkin tunnistettu yhdeksi tärkeimmistä menestystekijöistä ketteriä menetelmiä käyttävissä organisaatioissa (Tam et al., 2020).

Ketterät menetelmät korostavat ihmis- ja tiimikeskeisyyttä prosessikeskeisyyden sijaan. Ajatellaan, että kun ihmisiä käytetään tehokkaasti, niin myös kehitystyö on tehokasta. (Highsmith & Cockburn, 2001). Samalla myös kehittäjien ammattitaidon ja tiimien toimivuuden merkitys korostuu. Voidaan ajatella, että osaavien kehittäjien onnistuminen ei usein vaadi ennalta määrättyä prosessia, eikä toisaalta hyväkään prosessi pysty paikkaamaan puutteellista osaamista. Tiukat ja standardoidut prosessit eivät myöskään pysty ottamaan huomioon jokaisen työntekijän ja tiimin vahvuuksia. (Cockburn & Highsmith, 2001). Tiimien osaamistaso onkin ehkä tärkein menestystekijä ketterälle projektille (Tam et al., 2020) Toisaalta osa tutkimuksesta on tullut myös siihen tulokseen, että ketterien projektien onnistuminen riippuu ajateltua vähemmän henkilöstön osaamisesta ja että ketterillä menetelmillä voi olla mahdollista korvata puutteellista osaamista (Serrador & Pinto, 2015)

Ketterissä menetelmissä tiimeiltä vaaditaan itseohjautuvuutta sekä kykyä mukautua muutoksiin. (Nerur et al., 2005) Itseohjautuvuus ei kuitenkaan tarkoita, ettei tiimeissä olisi johtajuutta. Tiimit ovat kuitenkin rakenteeltaan mukautuvia. Ketteryys vaatii niiltä yhteisiä tavoitteita, luottamusta, kunnioitusta, kykyä tehdä yhdessä nopeasti päätöksiä sekä kykyä sietää epävarmuutta. (Cockburn & Highsmith, 2001) Sen sijaan että noudatetaan tarkkaa suunnitelmaa, johon tulee kuitenkin projektin aikana muutoksia, keskitytään muutokseen vastaamiseen. Muutoksien tarkoituksellinen välttäminen ei ole hyvästä, sillä ajatellaan että muutoksia tulee tehdä juuri sen verran, mitä ympäristö vaatii. (Highsmith & Cockburn, 2001)

Ketterissä menetelmissä ohjelmistojen dokumentaation rooli on toissijainen perinteisempiin menetelmiin verrattuna. Ajatellaan, että tieto liikkuu paremmin ihmisten välisessä kanssakäymisessä kuin erilaisten dokumenttien kautta. Kun dokumentaatiota on vähän, muuttuu käsillä olevaan ohjelmistoon liittyvä tieto siis hiljaiseksi tiedoksi. Tätä hiljaista tietoa voidaan levittää organisaatioissa esimerkiksi siirtelemällä työntekijöitä eri tiimien välillä. Toisaalta epäonnistuminen hiljaisen tiedon levittämisessä voi johtaa vaikeisiin tilanteisiin. (Nerur et al., 2005) Dokumentaation toissijaisuus ei kuitenkaan tarkoita, etteikö ketterissä menetelmissä luotaisi olleenkaan dokumentaatiota, mutta isoa osaa dokumentaatiota pidetään hukkana, eli asiakasarvoa lisäämättömänä työnä (Dingsøyr et al., 2012).

3. SUORITUKSEN MITTAAMINEN

Mannisen & Suomalain (2018) mukaan johdon laskentatoimeksi voidaan kutsua sitä osaa laskentatoimesta, jonka tehtävänä on tukea yrityksen päätöksen tekijöitä erilaisissa tilanteissa tarjoamalla mielekästä tietoa valintojen tueksi. Se voidaan ymmärtää sisäisenä palvelutoimintona, jonka keskeisiä tuotoksia ovat erilaiset kustannus- ja kannattavuus-tarkastelut sekä mittarit ja mittaristot. Johdon laskentatoimi on myös kontrollin väline, ja se mielletään usein osaksi johdon ohjausjärjestelmiä (engl. management control systems). Johdon ohjausjärjestelmillä tarkoitetaan kaikkia niitä mekanismeja, joilla pyritään saavuttamaan organisaation tavoitteiden mukainen toiminta. (Suomala et al., 2018)

Tässä työssä keskitytään suorituksen mittaamisen tarkasteluun osana johdon ohjausjärjestelmiä. Tätä varten esitellään aluksi johdon ohjausjärjestelmiä selittävä viitekehys. Tämän jälkeen tarkastellaan suorituksen mittaamista yleisesti. Lisäksi tarkastellaan suorituksen mittaamista projektienhallinnassa.

3.1 Mahdollistavat ja pakottavat johdon ohjausjärjestelmät

Adler & Borys (1996) mukaan organisaation toimintaa määräävät säännöt ja määräykset voivat olla joko pakottavia (engl. coercive) tai mahdollistavia (engl. enabling). Tätä jakoa pakottavaan ja mahdollistavaan voidaan soveltaa johdon ohjausjärjestelmiin ja käyttää apuna johdon ohjausjärjestelmien ymmärtämiseen niin tehokkuuden kuin joustavuuden parantamisen näkökulmasta (Ahrens & Chapman, 2004).

Pakottava systeemi rajoittaa tapoja, joilla työntekijät voivat toimia, kun taas mahdollistavat systeemit auttavat organisaation työntekijöitä selviämään tehtävistään. Vaikutus organisaation suorituskykyyn määräytyy kolmen eri ulottuvuuden kautta. Nämä kolme ulottuvuutta ovat järjestelmän ominaisuudet, suunnitteluprosessi sekä käyttöönotto. (Adler & Borys, 1996) Järjestelmän luonteen neljä selittävää ominaisuutta ovat korjaamisen mahdollisuus (engl. repair), sisäinen läpinäkyvyys (engl. internal transparency), ulkoinen läpinäkyvyys (engl. global transparency) sekä joustavuus (engl. flexibility) (Adler & Borys, 1996).

Korjaamisen mahdollisuudessa on kyse siitä, ettei organisaation toimintaa voida täysin määritellä etukäteen. Organisaation toimintaa korjaavat sekä parantavat prosessit tulisi tuoda osaksi organisaation arkea. Lisäksi työntekijöihin tulisi luottaa ja heitä tulisi kan-

nustaa organisaation kehittämiseen. (Ahrens & Chapman, 2004) Joustavuudella tarkoitetaan sitä, että työntekijä voi tarvittaessa poiketa etukäteen määritellyistä toimintatavoista (Adler & Borys, 1996).

Sisäinen läpinäkyvyys tarkoittaa sitä, että organisaation prosessit ovat siinä toimivien havaittavissa. Tätä voidaan edistää esimerkiksi korostamalla prosessien tärkeimpiä osia ja kodifioimalla parhaita toimintatapoja. Työntekijän tulisi saada oman toiminnan kannalta tärkeä tieto organisaatiosta. Ulkoisella läpinäkyvyydellä taas tarkoitetaan sitä, että organisaation jäsenellä on näkyvyys siihen laajempaan kontekstiin, jossa hän organisaatiossa työskentelee. (Ahrens & Chapman, 2004)

Työntekijöiden osallistamisella suunnitteluprosessiin voi olla työmoraalia ja -suorituskykyä parantava vaikutus. Muita positiivisesti vaikuttavia tekijöitä ovat mm. suunniteltavan menettelytavan relevanssi työntekijälle sekä riittävät resurssit ja koulutus. (Adler & Borys, 1996)

3.2 Suorituksen mittaaminen yleisesti

Suorituskykymittarit eivät ole ainoastaan johdon päätöksenteon työkalu. Jordan ja Messner (2012) mukaan suorituksen mittaamisella ohjataan työntekijöiden toimintaa ja viestitään johdon tahtotilasta. Ohjaavien mittareiden ehkä näkyvin ilmentymä on erilaiset kannustinjärjestelmät sekä arvioinnit työntekijöiden suoriutumisesta. Jotta tämä ohjaava vaikutus saavutetaan, ei suorituksen mittaaminen saa olla luonteeltaan liian joustavaa. Ylemmältä johdolta tulevat joustamattomat toimintatavat saatetaan kokea operatiivisessa johdossa kuitenkin pakottaviksi, etenkin tilanteissa, joissa mittareiden ei koeta esittävän toiminnan tilaa riittävän kokonaisvaltaisesti. (Jordan & Messner, 2012) Johdolta tuleva ohjaus onkin tasapainottelua ohjaamisen ja mahdollistamisen välillä (Laine et al., 2020).

Mittaaminen on kriittinen osa organisaation oppimista (Garvin, 1993). Suorituksen mittaamisessa keskitytäänkin usein erilaisiin kriittisiin menestystekijöihin. Nämä kriittiset menestystekijät ovat niitä tekijöitä, joiden seuraaminen on välttämätöntä yrityksen pitkän aikavälin kilpailukyvyyn kannalta. Organisaation eri mittauskohteilla voi olla erilaisia kriittisiä menestystekijöitä. Kriittisten menestystekijöiden tunnistaminen auttaa keskittymään olennaiseen, kehittämään suorituksen mittausta sekä auttaa yrityksen päätöksen tekijöitä keskittymään suuresta tietomassasta päätöksenteon kannalta oleellisimpaan tietoon. (Rockart, 1979) Suorituksen mittaaminen liittyykin olennaisesti organisaation strategian toteuttamiseen sekä sen toteutumisen seurantaan, ja muutokset strategiassa tulee huomioida mittaristossa (Bourne et al., 2000).

Organisaation ei siis tule jämähtää mittariston luomisvaiheessa valittuihin toimintatapoihin. Mittaristoa ja sen käyttöä tulee säännöllisesti kehittää vastaamaan organisaation sisäisiin muutoksiin sekä muutoksiin toimintaympäristössä. Tätä kykyä sopeuttaa mittaristoa kutsutaan mittariston dynaamisuudeksi. Jotta mittaristo tukee organisaation toimintaa tehokkaasti, tulee mitata asioita, jotka ovat tärkeitä mittaushetkellä, ei menneisyydessä. (Henri, 2010) Käyttämällä niin tilanteen mukaan muovautuvia kuin pysyvämpiä mittareita saadaan tukea niin lyhyen kuin pitkän aikavälin tarkastelua varten. Mittariston kehittäminen voi olla systemaattista tai ad hoc tyylistä ja tarve muutoksille suorituksen mittaamisessa voi syntyä myös itse suorituksen mittaamisen ja sitä kautta tehtyjen havaintojen kautta (Korhonen et al., 2013).

Nykymaailmassa organisaation kriittiset menestystekijät voivat olla monenlaisia, eikä kaikkia niitä voida mitata vain taloudellisin mittarein. Kaikkien kriittisten menestystekijöiden mukaan ottaminen suorituskyvyn mittaamiseen on siis oleellista. Esimerkiksi niin kutsuttu tasapainotettu mittaristo ottaa huomioon perinteisten taloudellisten seikkojen lisäksi organisaation asiakkaat, sisäiset prosessit sekä oppimisen mahdollistaen taloudellisen mittaamisen lisäksi mm. kyvykkyyksien kehittämisen ja erilaisten aineettomien hyödykkeiden mittaamisen (Kaplan & Norton, 1996). Tasapainotettua mittaristoa ei välttämättä kannata kuitenkaan ottaa käyttöön täysin varauksetta (Norreklit, 2000). Tasapainotettu mittaristo voi kuitenkin auttaa päätöksentekijöitä ymmärtämään organisaation strategiaa ja ottamaan sen huomioon päätöksenteossa (Humphreys, 2023). Strategian ymmärtäminen on tärkeää strategiaan liittyvien väärin oletusten sekä strategian toteuttamiseen liittyvien epäkohtien tunnistamisessa (Bourne et al., 2000).

3.3 Suorituksen mittaaminen projektien johtamisessa

Erään laajasti hyväksytyyn määritelmän mukaan ”projekti on ennalta määritettyyn päämäärään tähtäävä, monimutkaisten ja toisiinsa liittyvien tehtävien muodostama ajallisesti, kustannuksiltaan ja laajuudeltaan rajattu ainutkertainen kokonaisuus” (Arto et al., 2008 s. 26). Projektinhallinta taas tarkoittaa projektin tavoitteiden ja päämäärän saavuttamiseen tähtäävien johtamistapojen soveltamista (Arto et al., 2008 s. 35). Oikein käytettynä suorituksen mittaamisella voidaan tukea niin projektin johdon onnistumista kuin organisaatioiden menestymistä kokonaisuudessaan (Korhonen et al., 2023). On erityisen tärkeää kehittää mittareita, joiden avulla saadaan yhdistettyä projektin ja organisaation onnistuminen (Cooke-Davies, 2002).

Joustavuuden mahdollistava ohjausjärjestelmä sisältää usein mahdollisuuden siirtää väliaikaisesti kontrollia pois keskusjohdolta, lähemmäs projektin operatiivista johtoa. Tätä varten on kuitenkin tarpeen luoda selvät toimintaohjeet. On havaittu, että eri johtotasojen

vuorovaikutus ja paikalliselle johdolle annettu sopiva joustavuus voivat johtaa parempaan validiteettiin, luotettavuuteen ja sitoutumiseen johdon ohjaukseen. (Laine et al., 2020)

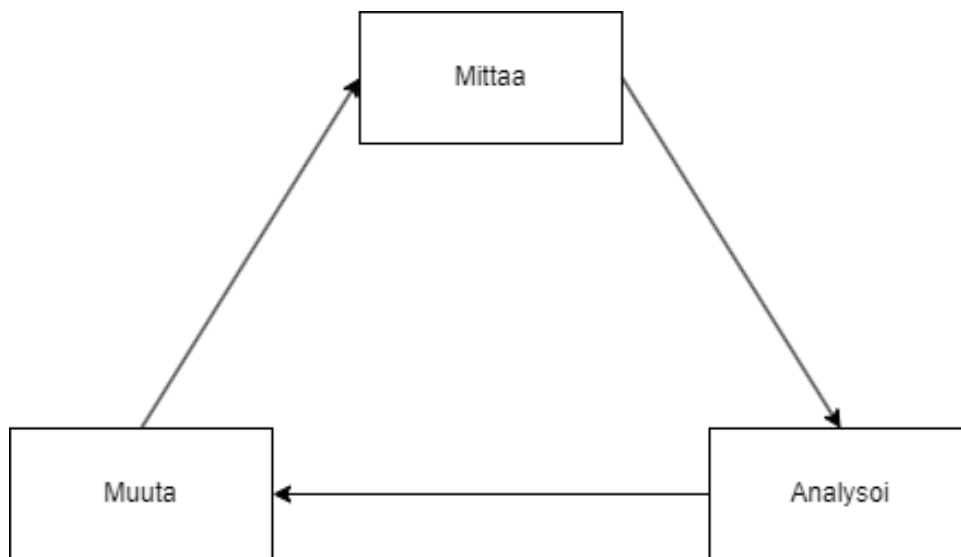
Projektien mittaamisessa onnistumisen moniulotteisuus voi tuoda haasteita. On mahdollista, että syntyy tilanteita, joissa projektin johdolle asetetut tavoitteet ovat ristiriidassa organisaation onnistumiselle tärkeiden tavoitteiden kanssa. Tällaisiin tilanteisiin voidaan tuoda tukea paremmalle päätöksenteolle lisäämällä projektille mittareita, jotka ottavat koko organisaation pitkän aikavälin menestymisen huomioon. Toisaalta organisaatiota ohjattaessa tulee ottaa projektien menestystekijät riittävästi huomioon. (Korhonen et al., 2023) Mittareiden tulee ottaa riittävässä määrin huomioon projektin eri sidosryhmät (Ben Abdallah et al., 2022). Ulkoinen läpinäkyvyys projektien johdon kontekstissa voi tarkoittaaakin sitä, että projektin johto ymmärtää projektin merkityksen organisaation laajemmassa kontekstissa ja tietää projektille keskusjohdon asettamat tavoitteet (Laine et al., 2020).

4. KETTERÄN OHJELMISTOPROSESSIN KEHITTÄMINEN JA MITTAAMINEN

Tässä pääluvussa esitellään aluksi ohjelmistoprosessin kehittämisen taustaa. Seuraavaksi käsitellään ohjelmistoprosessin kehittämisen periaatteiden soveltamista ketteriä menetelmiä hyödyntäviin ohjelmistoprosesseihin, joista käytetään tässä työssä myös nimitystä ketterä ohjelmistoprosessi. Lisäksi tarkastellaan tarkemmin, millainen on suorituksen mittaamisen rooli osana ketterien ohjelmistoprosessien kehittämistä sekä johdon ohjausjärjestelmää.

4.1 Ohjelmistoprosessin kehittämisen ja mittaamisen taustaa

Ensiaskel ohjelmistotuotannon kehittämisen on se, että sitä kohdellaan prosessina, jota voidaan ohjata, mitata ja kehittää. Ohjelmistoprosessia kehittäessä tulee ottaa huomioon prosessin eri vaiheiden väliset riippuvuudet, käytetyt työkalut ja menetotit sekä prosessissa mukana olevan henkilöstön taidot, koulutus ja motivaatio. (Humphrey, 1988)



Kuva 2 Ohjelmistoprosessin kehittämisen prosessimalli (mukaillen Sommerville, 2014 s.66)

Ohjelmistoprosessin kehittäminen kypsyyden kautta tarjoaa viitekehyksen toiminnan arviointiin ja oleellisimpien kehityskohteiden havaitsemiseen. (Humphrey, 1988) Keskittymällä pieneen määrään kehityskohteita kerralla voidaan saavuttaa tasaista parannusta ja pysyvää parannusta organisaation kykyyn tuottaa ohjelmistoja. (Pauk et al., 1993)

Jotta organisaatio voi päästä kypsyyden korkeammille tasoille tulee sen ottaa käyttöön kokonaisvaltaisesti ohjelmistoprosessia kuvaava mittaristo. (Humphrey, 1988)

Perinteisessä ajattelumallissa epäkypsässä ohjelmisto-organisaatiossa ohjelmistoprosessit eivät ole yleisesti ottaen johdon tai työntekijöiden ennalta määrittämiä, ja vaikka ennalta määriteltyjä toimintatapoja olisikin, niitä ei noudateta ja toimeenpanna riittävän täsmällisesti. Kypsällä ohjelmisto-organisaatiolla taas on koko organisaation leikkaava kyvykkyys johtaa ohjelmistotuotantoa ja ylläpitoa. Kypsässä organisaatiossa organisaation ohjelmistoprosessiin kohdistuvat ohjeet ovat helposti kommunikoitavissa toteutettavissa ja niitä noudatetaan. Prosessit ovat jatkuvan parantamisen kohteena, objektiivisesti arvioitavissa ja ne tuottavat helposti ennakoitavia lopputuloksia. (Paulk et al., 1993)

Erään tutkimuksen mukaan korkeamman maturiteetin saavuttaneet ohjelmistoprosessit tuottavat laadukkaampia ohjelmistoja, mutta hieman hitaammalla tahdilla, mahdollistaen kuitenkin kokonaisuudessa onnistuneemman lopputulokset (Harter et al., 2000). On olemassa näyttöä myös siitä, että ohjelmistoprosessin kehittämistä tekevät organisaatiot auttavat saavuttamaan liiketoiminnallista menestystä (Clarke & O'Connor, 2012).

Ohjelmistoprosessia ja siihen liittyvää tuotetta tai palvelua tulee mitata standardoidulla tavalla. Käytettyjen mittareiden tulee heijastua ohjelmistoprosessin kriittisiin menestystekijöihin. Ohjelmistoprosessin kehittämisen lisäksi kerättyä tietoa voidaan hyödyntää mm. riskien tunnistamisessa ja hallinnassa, kommunikoinnissa, toiminnan suunnittelussa ja ennakoimisessa, organisaation riippumattomassa ja tarkassa arvioinnissa sekä yleisesti erilaisissa päätöksenteko tilanteissa. (Jethani, 2013) Toisaalta heikkolaatuisten mittarien käyttö on tutkimuksessa yhdistetty SPI-ohjelmien epäonnistumiseen (Umarji & Seaman, 2008).

Prosessin lopputuotteen eli tässä tapauksessa ohjelmistotuotteen mittaaminen on myös oleellinen osa ohjelmistoprosessin laatujohtamista (Wallace & Sheetz, 2014). Itse ohjelmistoon liittyviä mittareita on myös vaikeampi vääristellä kuin prosessiin liittyviä. Lisäksi joissain tilanteissa ohjelmiston mittaaminen voidaan kokea vähemmän pakottavana (Umarji & Seaman, 2008). Mahdollisista hyödyistä huolimatta on ollut vaikeuksia myös ohjelmistoa kuvaavien mittaristojen käyttöönotossa (Wallace & Sheetz, 2014).

4.2 Ketterän ohjelmistoprosessin kehittämisen erikoispiirteet

Perinteisiin SPI-ohjelmiin kuuluu vahvasti dokumentaatio ja byrokratia siinä, missä moderneissa ketteriä menetelmiä käyttävissä organisaatioissa arvostetaan epävirallista kommunikaatiota, luovuutta ja joustavuutta (Coleman & O'Connor, 2008). Toisaalta tarve

ohjelmistoprosessin kehittämiseksi tuottavuuden ja laadun parantamiseksi on tunnistettu myös kehittäjien keskuudessa (Niazi et al., 2018).

Cockburn ja Highsmith (2001) mukaan ketterässä ohjelmistoprosessissa johdolta tulevat tavoitteet ja rajoitteet luovat raamit, joiden sisällä työskennellään. Ylemmän johdon tehtävä ei siis ole puuttua ruohonjuuritason päätöksen tekoon ja päätöksen teon tulisi tapahtua siellä, missä siihen on parhaat puitteet (Cockburn & Highsmith, 2001). Toisin sanoen erityisesti juuri ketterää ohjelmistoprosessia johtaessa ohjauksen ei tule olla liian estävää.

Tutkimuksessa on havaittavissa trendi, jossa ohjelmistoprosessin kehittäminen integroituu ketterän ohjelmistoprosessin muihin eri vaiheisiin mahdollistaen jatkuvan oppimisen (Kuhrmann & Muench, 2019). Perinteiset, verrattain raskaat ja kankeat SPI-lähestymistavat eivät ole kovinkaan käytettyjä. On kuitenkin havaittavissa, että ohjelmistoprosessin kehittäminen on ottanut joustavampia ja kevyempiä muotoja. Etenkin pienemmissä organisaatioissa erilaisten toimintatapojen käyttöön vaikuttaa vahvasti niiden mukana tulleet lisäkustannukset, mutta sama kehityssuunta on havaittavissa kaikenlaisissa organisaatioissa. (Küpper et al., 2019) Vaikuttaa siltä, että organisaatioissa on tunnistettu tarve ohjelmistoprosessin kehittämiseksi, joista on poistettu perinteisempien lähestymistapojen pakottaviksi koettuja ominaisuuksia.

Fontana et al. (2014) mukaan perinteisen kypsyyssajattelun mukaisesti kypsä ohjelmistoprosessi on niin tarkoin ohjattu ja ennalta määritelty, ettei se välttämättä voi olla ketterä. Vaikuttaa siltä, että ketterää ohjelmistoprosessia kehittäessä prosessin kypsyttävä tarkastellessa tulee kiinnittää huomiota myös subjektiivisempiin seikkoihin, kuten kykyyn tehdä yhteistyötä, kommunikoida, omistautua, välittää, jakaa ja itseohjautua (Kuhrmann & Muench, 2019). Kypsyysmalleilla on yhä tärkeä rooli osana organisaatioiden oppimista (Bititci et al., 2015). Ketterän ohjelmistoprosessin kehittämistä varten tarvitaan kypsyyssmalli, joka mahdollistaa prosessin kehittämisen lisäksi kehitystiimin erilaisten kyvykkyyksien kehittämisen (Fontana et al., 2014).

Osa organisaatioista on alkanut hyödyntämään ns. ketterää SPI-lähestymistapaa joustavampana ja kevyempänä vaihtoehtona perinteisille SPI-lähestymistavoille (Küpper et al., 2019; Poth et al., 2019). Tälle ketterälle ohjelmistoprosessin kehittämiseksi tyypillistä on iteratiivisuus, inkrementaalisuus sekä enemmän tai vähemmän epäviralliset retrospektiivit. Ketterän ohjelmistoprosessin kehittäminen voi olla perinteisemmälle organisaatiolle haastavaa, mutta se voi tapahtua varsin luonnollisesti niissä organisaatioissa, jotka hyödyntävät ketteriä menetelmiä jo valmiiksi. Perinteisiä SPI-malleja noudattaessa siihen osallistuvat voivat kokea prosessin irrallisena heidän työstään ja epämotivoivana

johtaen huonompiin lopputuloksiin (Poth et al., 2019) Ketterän ohjelmistoprosessin kehittämiseen käytetyn ohjauksen halutaan nähtävästi olevan melko dynaamista.

4.3 Ketterän ohjelmistoprosessin mittaamisen erikoispiirteet

Suorituksen mittaaminen auttaa ketteriä menetelmiä hyödyntäviä tiimejä arvioimaan suorituskkyä ja tuotettua laatua. Tiimit voivat pyrkiä parempiin suorituksiin tekemällä kokeiluja, saamalla niistä palautetta ja tekemällä tilanteen edellyttämiä toimia. On tärkeää tuntea tiimin toiminnan tarkka nykytilanne ja mittaaminen voi toimia myös motivaation lähteenä. (Ertaban et al., 2018) Lisäksi mittaamisen avulla voidaan luoda asiakasarvoa esimerkiksi pitämällä asiakas paremmin kartalla eri toiminnallisuuksien kehitystyön edistymisestä (Cheng et al., 2009).

Perinteisesti käytetyt mittarit eivät alan tutkimuksen mukaan vaikuta sopivan ketterän ohjelmistoprosessin mittaamiseen (Korpivaara et al., 2021). Ketterät ohjelmistoprosessit perustuvat yksittäisten työsuoritusten sijaan tiimityöhön. Suorituksen mittaaminen ja palkitsemisjärjestelmät tulee suunnitella tämä mielessä pitäen ketterän ohjelmistoprosessin onnistumiseksi. (Ertaban et al., 2018; Nerur et al., 2005) Ottaakseen huomioon organisaatioiden väliset niin kvantitatiiviset kuin kulttuurilliset erot tulee mittariston olla räätälöity käyttökontekstin mukaan. (Umarji & Seaman, 2008). Ketterään kehitykseen voidaan tuoda ennakoitavuutta kun tunnetaan tiimien kyvykkyydet ja kapasiteetit (Cheng et al., 2009). Toisaalta tiimien vertailu toisiinsa herättää usein vastustusta eikä ole yleensä suositeltavaa (Ertaban et al., 2018; Umarji & Seaman, 2008). Ketteriä menetelmiä käytävässä organisaatiossa käytettyjen mittareiden tulee olla muidenkin organisaation jäsenten kuin vain johdon tiedossa (Cheng et al., 2009).

Suorituksen mittaamisen tulee onnistuakseen olla riittävän kevyttä ja sen tulee sopia yrityksen prosesseihin. Lisäksi se, että kehittäjät ymmärtävät suorituksen mittaamisen hyödyt koko organisaatiolle ja kehitystiimin jäsenille edistää mittaamisen onnistumista (Umarji & Seaman, 2008). Ei siis välttämättä riitä, että suorituksen mittaamisella saavutetaan hyötyä, vaan tarvitaan riittävästi ulkoista ja sisäistä läpinäkyvyyttä, jotta kehittäjät havaitsevat nämä hyödyt. Lisäksi voisi olla hyödyllistä tehdä analyysi erilaisten mittareiden aiheuttamista kustannuksista ja saaduista hyödyistä ennen laajaa käyttöönottoa (Padmini et al., 2015).

Pakottaviksi koettujen mittaristojen käyttöönotto ohjelmistoalan organisaatioissa on johdanut muun muassa tilanteisiin, joissa suorituksen mittaukseen annettuja ohjeita ei noudateta ja mittaustuloksia vääristellään tahallisesti. On havaittu, että tiimien itse käyttöön ottamista mittaristoista on enemmän hyviä kokemuksia. (Umarji & Seaman, 2008) Onkin luontevaa, että tiimin itselleen luomaa mittaristoa ei koeta estävänä. Tätä oppia voitaisiin soveltaa organisaatiota laajemmin kattaviin mittaristoihin osallistamalla kehittäjiä mittariston luonnissa.

Korpivaara et al. (2021) mukaan on havaittavissa, että ketteriä menetelmiä hyödyntävän organisaation eri osia kiinnostaa erilaiset mittarit. Lähempänä ohjelmakoodia työskenteleviä kiinnostaa usein ohjelmistoprosessin tehokkuus sekä sitä tukevat tekijät, kun taas ylempää johtoa saattaa kiinnostaa enemmän tuotettu asiakasarvo ja toiminnan taloudelliset mittarit. Ongelmia syntyy erityisesti tilanteissa, joissa mittaristo korostaa organisaatioiden eri osissa erilaisia tavoitteita. Toisin sanoen ongelmallisia ovat tilanteet, joissa tiimejä koskevat mittarit eivät ole yhteneväisiä organisaation onnistumisen kannalta tärkeimpien mittareiden kanssa. (Korpivaara et al., 2021) Tämän perusteella vaikuttaa siltä, että mittariston luomista ei voi kuitenkaan täysin jättää ohjelmistokehitystä hoitavien tiimien vastuulle, vaan sen suunnittelemiseen tulee osallistua myös ylempää johtoa. Näin voidaan koittaa varmistaa, että suorituksen mittaaminen tukee tiimien onnistumisen lisäksi koko organisaation onnistumista.

5. PÄÄTELMÄT

Ohjelmistokehitys vaikuttaa olleen jo pitkään jatkuvassa murroksessa. Teknologiat kuten olio-ohjelmointi ja prosessi-innovaatiot kuten ketterät menetelmät ovat mahdollistaneet nykyisen tuottavuuden. Alan kilpailu on kuitenkin kovaa, ja monet organisaatiot etsivät tapoja parantaa ohjelmistokehityksen tuottavuutta ja kykyä vastata asiakastarpeisiin.

Työn tarkoitus oli vastata alussa esiteltyihin tutkimuskysymyksiin:

1. Miten ohjelmistoprosesseja voidaan systemaattisesti kehittää ja mikä on mittaamisen rooli niiden kehittämisessä?
2. Mitä tulee ottaa huomioon etenkin ketterän ohjelmistoprosessin systemaattisessa kehittämisessä ja mittaamisessa?

Perinteinen tapa kehittää ohjelmistokehitystä systemaattisesti keskittyy ohjelmistokehityksen näkemiseen hallittavana ja standardoitavana prosessina. Tätä ohjelmistoprosessia voidaan mitata, ymmärtää ja kehittää. Oleellinen osa näitä prosessikehityksen malleja on käsillä olevan ohjelmistoprosessin kypsyyden tunnistaminen. Tämä auttaa niiden osa-alueiden tunnistamisessa, jotka vaativat eniten huomiota ja kehitystoimenpiteitä.

On havaittavissa, että nämä perinteiset mallit koetaan epäsoviviksi modernille ketteriä menetelmiä hyödyntäville ohjelmistoprosesseille. Ne rajoittavat liikaa ketterille menetelmille tyypillistä ja tärkeää joustavuutta sekä ovat turhan raskaita esimerkiksi vaaditun dokumentaation osalta. Perinteiset mallit ovat hyvin prosessikeskeisiä eivätkä tästä syystä myöskään ota kunnolla huomioon ihmiskeskeisille ketterille menetelmille tärkeitä kyvykkyyksiä.

Alalla on kuitenkin havaittu tarve ketterienkin ohjelmistoprosessien kehittämiselle ja jatkuvalla oppimiselle. Tämän tarpeen takia on syntynyt vanhoja ohjelmistoprosessin kehityksen malleja kevyempiä keinoja kehittää toimintaa. Organisaatioissa on myös tunnistettu, että dynaamisten projektien parissa toimivat ketteriä menetelmiä käyttävät tiimit tarvitsevat dynaamisempaa johdon ohjausta edustavia ketteriä ohjelmistoprosessin kehitysmalleja. Näitä ketterämpiä lähestymistapoja voidaankin mahdollisesti hyödyntää myös perinteisemmissä ohjelmistoprosesseissa (Küpper et al., 2019).

Ketteriä menetelmiä käyttävää ohjelmistoprosessia kehittäessä tulee kiinnittää erityistä huomiota käytettyihin mittareihin. Suorituksen mittaaminen voidaan organisatorisen kontekstin ja käytetyn mittariston ominaisuuksista riippuen koeta pakottava ohjaamisena,

mikä johtaa usein ohjelmistoprosessin kehityshankkeessa heikompiin tuloksiin. Mittariston käyttöönottoa tukee se, että kehittäjille annetaan hyvä ulkoinen- ja sisäinen läpinäkyvyys mittariston merkitykseen. Suorituksen mittausta saatetaan pitää vähemmän pakottavana, jos kehittäjät pääsevät osallistumaan sen kehittämiseen. Toisaalta samalla tulee pitää huoli, että luotu mittaristo tukee koko organisaation eikä vain yksittäisten tiimien onnistumista

5.1 Tutkimuksen arviointi ja jatkotutkimusmahdollisuudet

Ohjelmistotuotannon kehittämiseen liittyvää kirjallisuutta on olemassa valtavat määrät. Tutkimuksen aikana saatiin tunnistettua hakutermejä, joilla löytyi tutkimuksen kannalta relevanttia kirjallisuutta. Kirjallisuushakuja tehdessä ei onnistuttu kuitenkaan löytämään sellaisia hakutermejä, joilla löytyisi vain relevanttia kirjallisuutta. Tämän takia relevantteja julkaisuja jouduttiin valitsemaan melko suuresta massasta, mikä johtaa siihen, että on hyvin todennäköistä, että myös relevanttia kirjallisuutta on voinut jäädä tutkimuksen ulkopuolelle. Yksi tutkimuksen rajoitteista on myös se, että kirjallisuutta on käytännössä etsitty vain Scopus-tietokannasta.

Alan kirjallisuudessa on havaittavissa joitain puutteita. Esimerkiksi ohjelmistoprosessin kehittämisestä ketteriä menetelmiä hyödyntävissä ohjelmistoprosesseissa ei vaikuta olevan ainakaan vielä julkaistu paljoa kiistattomia todisteita. Ohjelmistoalalla käsitys parhaista käytännöistä tuntuukin perustuvan usein käytännön kokemukseen eikä tieteellisiin menetelmin parhaiksi todettuihin toimintatapoihin.

Työn aikana on tunnistettu muutamia muitakin jatkotutkimusmahdollisuuksia. Ensinnäkin moderniin ohjelmistokehitykseen kohdistuviin vaatimuksin vastatakseen monet organisaatiot ovat ottaneet käyttöön globaaleja tiimejä. Tämä luo suuria haasteita ketterien menetelmien käytölle, ohjelmistoprosessin kehittämiseksi ja tuo uusia ulottuvuuksia suorituksen mittaamiselle.

Lisäksi pinnalla on monenlaisia uusia teknologioita, joista päällimmäisenä tällä hetkellä vaikuttavat olevan tekoäly ja koneoppiminen. Alan tutkimuksessa tarkastellaan muun muassa sitä, voisiko näiden avulla mitata ohjelmistoprosessissa syntyvää ohjelmistotuotetta tai auttaa ohjelmistoprosessissa työskentelevää kehittäjää saavuttamaan parempaa tuottavuutta ja laadukkaampaa koodia.

LÄHTEET

- Adler, P. S., & Borys, B. (1996). Two Types of Bureaucracy: Enabling and Coercive. *Administrative Science Quarterly*, 41(1), 61.
- Agrawal, A., Atiq, M. A., & Maurya, L. S. (2016). A Current Study on the Limitations of Agile Methods in Industry Using Secure Google Forms. 78, 291–297.
- Ahrens, T., & Chapman, C. S. (2007). Management accounting as practice. *Accounting, Organizations and Society*, 32(1), 1–27.
- Artto, K. A., Martinsuo, M., & Kujala, J. (2008). *Projektiliiketoiminta*. WSOY.
- Beck, K., et al. (2001) The Agile Manifesto. Agile Alliance. <http://agilemanifesto.org/>
- Ben Abdallah, S., El-Boukri, S., Floricel, S., Hudon, P., Brunet, M., Petit, M., & Aubry, M. (2022). A process-oriented framework to measure development performance and success of megaprojects. *International Journal of Project Management*, 40(6), 685–702.
- Bititci, U. S., Garengo, P., Ates, A., & Nudurupati, S. S. (2015). Value of maturity models in performance measurement. *International Journal of Production Research*, 53(10), 3062–3085.
- Bourne, M., Mills, J., Wilcox, M., Neely, A., & Platts, K. (2000). Designing, implementing and updating performance measurement systems. *International Journal of Operations & Production Management*, 20, 754–771.
- Butler, C. W., Vijayasathy, L. R., & Roberts, N. (2020). Managing Software Development Projects for Success: Aligning Plan- and Agility-Based Approaches to Project Complexity and Project Dynamism. *Project Management Journal*, 51(3), 262–277.
- Cheng, T.-H., Jansen, S., & Remmers, M. (2009). Controlling and monitoring agile software development in three dutch product software companies. 29–35.
- Clarke, P., & O'Connor, R. V. (2012). The influence of SPI on business success in software SMEs: An empirical study. *Journal of Systems and Software*, 85(10), 2356–2367.
- Cockburn, A., & Highsmith, J. (2001). Agile software development: The people factor. *Computer*, 34(11), 131–133.
- Coleman, G., & O'Connor, R. (2008). Investigating software process in practice: A grounded theory perspective. *Journal of Systems and Software*, 81(5), 772–784.
- Cooke-Davies, T. (2002). The “real” success factors on projects. *International Journal of Project Management*, 20(3), 185–190.
- Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, 85(6), 1213–1221.
- Ertaban, C., Sarikaya, E., & Bagriyanik, S. (2018). Agile performance indicators for team performance evaluation in a corporate environment. Part F147763.

- Fontana, R. M., Fontana, I. M., Da Rosa Garbuio, P. A., Reinehr, S., & Malucelli, A. (2014). Processes versus people: How should agile software development maturity be defined? *Journal of Systems and Software*, 97, 140–155.
- Garvin, D. A. (1993). Building a learning organization. *Harvard Business Review*, 71(4), 78–91.
- Harter, D. E., Krishnan, M. S., & Slaughter, S. A. (2000). Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development. *Management Science*, 46(4), 451–466.
- Henri, J.-F. (2010). The Periodic Review of Performance Indicators: An Empirical Investigation of the Dynamism of Performance Measurement Systems. *European Accounting Review*, 19(1), 73–96.
- Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. *Computer*, 34(9), 120–122.
- Humphrey, W. S. (1988). Characterizing the software process: A maturity framework. *IEEE Software*, 5(2), 73–79.
- Humphreys, K. A. (2023). The balanced scorecard: Do managers need a strategy map when evaluating performance? *Accounting and Finance*.
- Jethani, K. (2013). Software metrics for effective project management. *International Journal of System Assurance Engineering and Management*, 4(4), 335–340.
- Jordan, S., & Messner, M. (2012). Enabling control and the problem of incomplete performance indicators. *Accounting, Organizations and Society*, 37(8), 544–564.
- Kaplan, R. S., & Norton, D. P. (1992). The balanced scorecard—Measures that drive performance. *Harvard Business Review*, 70(1), 71–79.
- Korhonen, T., Jääskeläinen, A., Laine, T., & Saukkonen, N. (2023). How performance measurement can support achieving success in project-based operations. *International Journal of Project Management*, 41(1).
- Korhonen, T., Laine, T., & Suomala, P. (2013). Understanding performance measurement dynamism: A case study. *Journal of Management & Governance*, 17(1), 35–58.
- Korpivaara, I., Tuunanen, T., & Seppänen, V. (2021). Performance Measurement in Scaled Agile Organizations.
- Kuhrmann, M., & Münch, J. (2019). SPI is Dead, isn't it? Clear the Stage for Continuous Learning! 2019 IEEE/ACM International Conference on Software and System Processes (ICSSP), 9–13.
- Kuhrmann, M., Tell, P., Hebig, R., Klünder, J., Münch, J., Linssen, O., Pfahl, D., Felderer, M., Prause, C. R., MacDonell, S. G., Nakatumba-Nabende, J., Raffo, D., Beecham, S., Tüzün, E., López, G., Paez, N., Fontdevila, D., Licorish, S. A., Küpper, S., ... Richardson, I. (2022). What Makes Agile Software Development Agile? *IEEE Transactions on Software Engineering*, 48(9), 3523–3539.

- Küpper, S., Pfahl, D., Jürisoo, K., Diebold, P., Münch, J., & Kuhmann, M. (2019). How has SPI changed in times of agile development? Results from a multi-method study. *Journal of Software: Evolution and Process*, 31(11), e2182.
- Laine, T., Korhonen, T., & Suomala, P. (2020). The dynamics of repairing multi-project control practice: A project governance viewpoint. *International Journal of Project Management*, 38(7), 405–418.
- Lee, J.-C., Shiue, Y.-C., & Chen, C.-Y. (2016). Examining the impacts of organizational culture and top management support of knowledge sharing on the success of software process improvement. *Computers in Human Behavior*, 54, 462–474.
- Meidan, A., García-García, J., Ramos, I., & Escalona, M. (2018). Measuring Software Process: A Systematic Mapping Study. *ACM Computing Surveys*, 51(3), 1–32.
- Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM*, 48(5), 72–78.
- Niazi, M., Mishra, A., & Gill, A. Q. (2018). What Do Software Practitioners Really Think About Software Process Improvement Project Success? An Exploratory Study. *Arabian Journal for Science and Engineering*, 43(12), 7719–7735.
- Norreklit, H. (2000). The balance on the balanced scorecard a critical analysis of some of its assumptions. *Management Accounting Research*, 11(1), 65–88.
- Padmini, K. V. J., Dilum Bandara, H. M. N., & Perera, I. (2015). Use of software metrics in agile software development process. 312–317.
- Paulk, M. C., Curtis, B., Chrissis, M. B., & Weber, C. V. (1993). Capability maturity model, version 1.1. *IEEE Software*, 10(4), 18–27.
- Poth, A., Sasabe, S., Mas, A., & Mesquida, A.-L. (2019). Lean and agile software process improvement in traditional and agile environments. *Journal of Software: Evolution and Process*, 31(1).
- Ramesh, B., Cao, L., & Baskerville, R. (2010). Agile requirements engineering practices and challenges: An empirical study. *Information Systems Journal*, 20(5), 449–480.
- Rockart, J. F. (1979). Chief executives define their own data needs. *Harvard Business Review*, 57(2), 81–93.
- Serrador, P., & Pinto, J. K. (2015). Does Agile work? — A quantitative analysis of agile project success. *International Journal of Project Management*, 33(5), 1040–1051.
- Sommerville, I. (2016). *Software engineering* (Tenth edition., Global edition.). Pearson.
- Suomala, P., Manninen, O., & Lyly-Yrjänäinen, J. (2018). *Laskentatoimi johtamisen tukena* (Ellibs, Ed.; 2. painos.). Edita.
- Tam, C., Moura, E. J. D. C., Oliveira, T., & Varajão, J. (2020). The factors influencing the success of on-going agile software development projects. *International Journal of Project Management*,
- Umarji, M., & Seaman, C. (2008). Why do programmers avoid metrics? 129–138.

Wallace, L. G., & Sheetz, S. D. (2014). The adoption of software measures: A technology acceptance model (TAM) perspective. *Information and Management*, 51(2), 249–259.