Erno Pappinen

TIIMIN SISÄISET KETTERÄT VIESTINTÄKÄYTÄNTEET SCRUMIN JA XP:N MUKAISESSA OHJELMISTOKEHITYKSESSÄ

Tietojärjestelmätieteen kandidaatin tutkielma

23. kesäkuuta 2010

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS

TIIVISTELMÄ

Pappinen, Erno
Tiimin sisäiset ketterät viestintäkäytänteet Scrumin ja XP:n mukaisessa ohjelmistokehityksessä / Erno Pappinen
Jyväskylä: Jyväskylän yliopisto, 2010.
46 s.

Ketterien menetelmien viestintää on käsitelty kirjallisuudessa paljon, mutta suoranainen ketterän viestinnän tutkimus on melko vähäistä viestinnän haasteellisen luonteen vuoksi ja ketterien menetelmien ollessa vielä kehityksensä alkupäässä. Ketteriin menetelmiin ja sen viestintään liittyy monia haasteita niiden ollessa suurelta osin kytköksissä työtä ja viestintää harjoittaviin ihmisiin ja ihmisten erilaisuuteen, mistä johtuen näihin ongelmiin ei ole absoluuttisia ratkaisuja olemassa. Haasteista huolimatta ketterien menetelmien on kuitenkin osoitettu vaikuttavan positiivisesti ketterän ohjelmistokehitystiimin sisäiseen viestintään.

Tämä tutkimus erittelee kirjallisuuskatsauksena ketterien ohjelmistoprojektien tiimin sisäistä viestintää ja viestintäkäytänteitä sekä pyrkii listaamaan kriittisimpiä ketteriä viestintäkäytäntöjä ja näiden vaikutusta kehitystiimin sisäiseen erityisesti kasvokkain tapahtuvaan viestintään. Tässä tutkimuksessa selvisi, että kaikki ketterien menetelmien käytännöt ovat osaltaan myös viestintäkäytänteitä. Tämä osoittaa viestinnän olevan entistäkin tärkeämpi osa ketterää kehitystä.

Erityisesti kasvokkaisviestinnän merkitys ketterälle kehitykselle on todella merkittävä. Tästä johtuen monet ketterät käytännöt, kuten esimerkiksi avoin työskentelytila, tiivis kokoustamistahti, läsnä oleva ja tiimiin kuuluva asiakas sekä pariohjelmointi ovat kasvokkaisviestintää tukevia käytänteitä.

AVAINSANAT: Viestintä, ketterät menetelmät, kasvokkaisviestintä

Ohjaaja: Mauri Leppänen
Tietojenkäsittelytieteiden laitos
Jyväskylän Yliopisto

Tarkastaja: Mauri Leppänen
Tietojenkäsittelytieteiden laitos
Jyväskylän Yliopisto

SISÄLTÖ

1 JOHDANTO 5

2 VIESTINTÄ 9
2.1 Mitä on viestintä? 9
2.2 Prosessikoulukunta ja merkityskoulukunta 10
2.3 Organisaatioviestintä 12
2.4 Projektiviestintä 13
2.4.1 Projektiviestinnän prosessit 13
2.4.2 Ohjelmistoprojektin viestintämuodot 15

3 KETTERÄT MENETELMÄT 17
3.1 Mitä ovat ketterät menetelmät? 17
3.2 Scrum 19
3.2.1 Scrumin ominaispiirteitä 20
3.2.2 Scrumin prosessimalli 20
3.3 Extreme Programming (XP) 22
3.3.1 XP:n ominaispiirteitä 23
3.3.2 XP:n prosessimalli 27

4 VIESTINTÄ SCRUM- JA XP-MENETELMISSÄ 29
4.1 Viestinnän keskeinen merkitys ohjelmistokehityksessä 29
4.2 Ketterät viestintäkäytänteet ja niiden edut 30

5 YHTEENVETO 38

LÄHDELUETTELO 41

1 JOHDANTO

Ohjelmistojen ja tietojärjestelmien kehittäminen on aina yhteistyötä kehittäjien, asiakkaan, loppukäyttäjien sekä johdon välillä. Tämän vuoksi tietämyksen jakaminen eli viestintä on projektin onnistumisen kannalta ensiarvoisen tärkeää. Amblerin (2010) mukaan viestintä on tiedon siirtämistä yksilöltä toiselle. Hänen mukaansa tehokas viestintä nähdään ohjelmistokehityksen yhtenä tärkeimmistä osa-alueista. Tässä työssä *tietämyksestä* puhuttaessa tarkoitetaan pääsääntöisesti hiljaista tietoa ja *tietämyksen jakamisella* tarkoitetaan hiljaisen tiedon (tacit knowledge) siirtämistä ohjelmistokehitysprojektin osallisten välillä.

Perinteinen näkökulma tietämykselle ja sen jakamiselle on ns. ”tietämys objektina” (knowledge-as-object), missä tietämys nähdään dataobjekteina, joita voidaan luoda, kerätä, varastoida ja käyttää uudelleen eri tilanteissa. Vaikka tämä näkökulma on edelleen hallitsevassa asemassa erityisesti ohjelmistokehityksessä, niin myös toinen näkökulma on vakiinnuttamassa asemaansa. Tämän mukaan tietämys nähdään ihmissuhteena (knowledge-as-relationship), jossa tietämys on sosiaalisesti rakentunutta ja kollektiivisesti säilytettyä. (Melnik & Maurer, 2004)

Projektityöskentelyyn liittyy useita muuttujia, kuten sosiaaliset muuttujat, persoonallisuuksien eroavaisuudet, koulutuksellinen tausta sekä erilaiset tavat kommunikoida. Näiden merkitys tehokasta kommunikointia vaativissa projekteissa on verrattain suuri. Siemens Corporate Research (SCR) on tehnyt tutkimuksen viestinnän merkityksestä oman kehitysprosessinsa S-RaP:n (Siemens Rapid Prototyping) yhteydessä (Tai, 2005). Tutkimuksessa huomattiin, että prosessin mukainen ketterä kehitys vaatii todella tehokasta

viestintää, mutta tätä ei prosessin kuvauksessa oltu kuitenkaan määritelty (Tai, 2005).

Viestintä on erityisesti ketterän ohjelmistokehityksen ehkä jopa kriittisin menestystekijä (Ambler, 2010). Perinteisiä menetelmiä noudattavien projektien tärkein viestinnän keino on ollut dokumentaatio, mutta nykyisissä projekteissa dokumentaation on huomattu olevan aina vanhentunutta projektin tilaan nähden (Melnik & Maurer, 2004). Tämän vuoksi ketterissä menetelmissä korostetaan kasvokkaisviestintää (face-to-face communication). Ketterissä menetelmissä myös painotetaan vapautta ja vuorovaikutusta sääntöjen sijaan, mutta tästä huolimatta joissakin menetelmissä, kuten esiemerkiksi Scrumissa, on hyvin tarkkoja sääntöjä viestinnän osalta. (Marcal, Soares, & Belchior, 2007)

Ketteriä menetelmiä on useita ja niiden ketteräksi lukeutuminen vaihtelee näkökulman mukaan. Chown ym. (2008) mukaan kirjallisuudessa mainitaan yleensä Extreme Programming (XP) (Beck & Anders 2004), Scrum (Schwaber, 1995), Feature-Driven Development (FDD) (Coad, Luca, & Lefebvre, 1999), Dynamic System Development (DSDM) (DSDM Consortium, 2010), Adaptive Software Development (ASD) (Highsmith, 2002), Crystal (Cockburn, 2004) sekä Lean Software Development (LD) (Poppendieck & Poppendieck, 2003). Tässä kirjallisuuskatsauksessa keskitytään Scrumiin ja XP:hen, koska ne ovat kaksi selvästi suosituinta ketterää menetelmää (Fitzgerald, Hartnett, & Conboy, 2006), mutta viitatessa ketteriin menetelmiin yleisesti tarkoitetaan yllä mainittuja menetelmiä.

Ketteriä menetelmiä otetaan yrityksissä käyttöön kiihtyvällä tahdilla, koska niiden on todettu tehostavan viestintää yrityksessä (Pikkarainen, Haikara, Salo, Abrahamsson, & Still, 2008). Tätä tukevia tutkimuksia on kuitenkin olemassa melko rajallinen määrä (Pikkarainen ym., 2008). Ketterän kehityksen on

huomattu olevan tehokkaampi tapa kehittää ohjelmistoja tai järjestelmiä, mutta tieteellinen tieto niiden tehokkuudesta ja paremmuudesta on vähäistä ja hataraa (Chow & Cao, 2008). Chown ja Caon (2008) tutkimuskin käsittelee ketterien menetelmien menestystekijöitä yleisesti ja monet muutkin näitä menetelmiä käsittelevät tutkimukset näyttäisivät käsittelevän niitä ennemminkin yleisesti kuin keskittyen nimenomaan tiettyihin spesifeihin seikkoihin, kuten pelkkään viestintään. Muutamia viestintään näiden menetelmien kohdalla keskittyviä tutkimuksia on kyllä tehty, mutta systemaattinen ja syvällinen tieto ketterien menetelmien kohdalla kuitenkin alalta puuttuu edelleen (Pikkarainen ym., 2008). Tämän vuoksi viestintä ketterissä ohjelmistokehitysprojekteissa on tärkeä aihe tutkimukselle ja koska tehokas tietämyksen jakaminen on avain kilpailuedun rakentamiseen kaikissa nykyorganisaatioissa sekä –yrityksissä (Melnik & Maurer, 2004).

Tämä kirjallisuuskatsaus käsittelee pääasiallisesti Scrumin ja XP:n mukaisten ohjelmistoprojektien tiimien sisäisiä, pääasiassa kasvokkaisviestintäkäytänteitä, joista käytetään tässä tutkimuksessa nimitystä *ketterät viestintäkäytänteet*. *Tiimillä* tässä tutkimuksessa tarkoitetaan ketterän ohjelmistokehityksen ydintiimiä, johon kuuluu ohjelmistokehittäjät, projektipäällikkö ja läsnä oleva asiakas. Tarkoituksena on alan kirjallisuuteen tutustumalla tunnistaa ketterien menetelmien käytännöistä nimenomaan viestintään liittyviä käytänteitä ja eritellä näiden etuja ohjelmistokehitysprojektille. Tämän työn tutkimusongelma voidaan muotoilla seuraavasti: Millaisia ketteriä viestintäkäytänteitä ketterässä ohjelmistokehityksessä käytetään ja tarjoavatko nämä käytännön hyötyä perinteisiin menetelmiin verrattuna? Tutkimusongelman voi tiivistää seuraaviin kysymyksiin: 1) Mitä ovat ketterät viestintäkäytänteet? 2) Mitä etua ketterillä viestintäkäytänteillä saavutetaan?

Seuraavissa kahdessa luvussa käsitellään viestintää ja ketteriä menetelmiä yleisesti sekä esitellään Scrumin ja XP:n ominaispiirteitä sekä niiden prosessimallit. Neljännessä luvussa listataan ketteriä viestintäkäytänteitä ja eritellään niiden sisältöä ja käyttöä sekä perehdytään niillä saavutettaviin etuihin ketterissä ohjelmistoprojekteissa.

2 VIESTINTÄ

Tämä luku käsittelee yleisesti viestintää. Aluksi käsitellään viestinnän määritelmiä ja sen jälkeen syvennytään tarkemmin viestintään sen kahden koulukunnan, prosessikoulukunnan ja merkityskoulukunnan kautta. Tämän jälkeen käsitellään vielä lyhyesti organisaatioviestintää ja sitä kautta projektiviestintää, mikä on tämän työn pääteema.

2.1 Mitä on viestintä?

Viestintä on ihmisen lajityypille läheistä sekä ominaista toimintaa, joka opitaan lapsena ja joka jatkuu läpi elämän. Viestintä on myös perusta inhimilliselle vuorovaikutukselle ja välttämätön ehto ihmisen sosiaaliselle elämälle. (Juholin, 2009) Yksinkertaisimmillaan viestintä tarkoittaa tietämyksen vaihdantaa järjestelmien välillä. *Informaatio* on olennainen osa viestinnän käsitettä ja tarkoittaa viestiä, jolla on jokin sisältö, mutta toisaalta ilman informaatiota viestintä ei ole viestintää. Yleensä informaatiolla katsotaan olevan myös jokin arvo, mutta sen arvon määrittely on hyvin tulkinnanvaraista. Järjestelmä taas tässä tapauksessa tarkoittaa viestivää entiteettiä, esimerkiksi ihmisen keskushermostoa tai tietokoneen prosessoria. (Wiio, 1992)

Viestinnälle on olemassa useita satoja määritelmiä, mutta Wiion (1992) tutkimat yli kaksisataa määritelmääkään eivät kykene selittämään sitä yleispätevästi. Jotkut määritelmät kuten ”viestintä on informaation siirtoa paikasta toiseen” tai ”viestintä on viestintää” jättävät liikaa aukkoja sekä tulkinnanvaraisuuksia. Shannonin ja Weaverin (1949) määritelmä sisältää pitkälti tärkeimmät tekijät. He sanovat viestinnän sisältävän kaikki ne toiminnot, joilla yksi mieli vaikuttaa toiseen mieleen tai vielä laajemmin ajateltuna ne toiminnot millä yksi järjestelmä vaikuttaa toiseen järjestelmään. Tähän kun lisätään vielä Åbergin

(2000) määritelmä, minkä mukaan viestintä on tapahtuma, jossa annetaan merkityksiä, tulkitaan niitä ja saatetaan niitä muiden tietoisuuteen verkostojen kautta, on viestinnän määritelmä melko kattava. Puhuttaessa *viestinnästä* tässä tutkimuksessa tarkoitetaan siis tietämyksen siirtämistä eri järjestelmien välillä, tässä tapauksessa kehitystiimin sisällä, muokkaamalla viestejä ja antamalla niille oikeanlaisia merkityksiä, jotta tietämyksen siirtyminen olisi mahdollisimman tehokasta ja kokonaisvaltaista kaikkien osapuolten kesken.

2.2 Prosessikoulukunta ja merkityskoulukunta

Viestinnän tutkimuksessa on yleisesti eroteltu kaksi koulukuntaa, prosessikoulukunta ja merkityskoulukunta (Juholin, 2009). Prosessikoulukunta nimensä mukaisesti tutkii viestintäprosessia eli sanomien siirtoa ja niiden vaikutusta. Sen mukaan viestinnän sisällöllä ei ole merkitystä, vaan ainoastaan sillä, että viesti menee perille (Juholin, 2009). Åbergin (2000) sanoin prosessinäkemys viestinnän tehosta on ”ruusuinen”, koska sen mukaan tehokas viestiminen johtaa automaattisesti viestin perille menoon ja halutun toiminnan aktivoimiseen. Todellisuudessahan viestintä ei kuitenkaan ole lähellekään näin suoraviivaista. Viestejä torjutaan lukemattomista syistä eikä sillä ole mitään tekemistä sen kanssa, että viestiä ei olisi ymmärretty. On myös syytä pohtia, onko tällaista suoraviivaista viestintää kannattavaa edes tavoitella, koska keskustelu ja asioiden kyseenalaistaminen on se, mikä saa asiat oikeasti kulkemaan eteenpäin. Aulan (1999) mukaan prosessinäkökulmassa merkitys puetaan sanomiksi, jotka tarjotaan vastaanottajalle välittämättä siitä ymmärsikö vastaanottaja viestin vai ei.

Merkityskoulukunta on viestintää toisella tavalla ajatteleva näkökulma, vaikkakaan esimerkiksi Aula (1999) ei näe sitä prosessikoulukunnan vastakohtana. Merkityskoulukunnan mukaan viestintä ei ole hallittavissa oleva

suoraviivainen prosessi, vaan sattumanvarainen tapahtuma, jossa vastaanottajan elämänkokemus ja –tilanne vaikuttavat viestinnän tulkintaan hyvin paljon (Åberg, 2000). Myös Wiio (1978) on vakaasti sitä mieltä, että viestintä on sattumanvaraista, koska hänen muotoilemansa lain mukaan viestintä epäonnistuu aina, paitsi sattumalta. Aulan (1999) mukaan merkitysnäkökulmassa merkitystä ei välitetä, vaan se luodaan lähettäjän ja vastaanottajan kesken yhdessä. Lisäksi hänen mukaansa merkityksen syntyyn vaikuttavia tekijöitä ovat kieli, kulttuuri, ympäristö, menneisyys ja tulevaisuus. Erityisesti viestinnän tehokkuuteen vaikuttaa viestin lähettäjän ja vastaanottajan tuntemisen taso sekä heidän tarpeensa ja odotuksensa, minkä perusteella viestin voi esimerkiksi muotoilla paremmin tai käyttää viestimiseen oikeita kanavia ja keinoja (Juholin, 2009).

Prosessikoulukunta ja merkityskoulukunta rinnastuvat hyvin Melnikin ym. (2004) tutkimuksessa esitettyihin kahteen jäsennykseen ”tietämys objektina” (knowledge-as-object) ja ”tietämys ihmissuhteena” (knowledge-as-relationship). Tietämys objektina on hyvin pitkälti sama näkemys kuin prosessikoulukunnan näkemys viestinnästä, koska siinä tietämys nähdään dataobjekteina, joita voidaan luoda, kerätä, varastoida ja käyttää vaivattomasti uudelleen ja prosessikoulukunta keskittyy pelkästään viestien siirtoon eikä ota kantaa sisältöön. Tietämys ihmissuhteena taas on hyvin samankaltainen näkemys kuin merkityskoulukunnan näkemys, koska siinä tietämys nähdään sosiaalisesti rakentuneena ja kollektiivisesti säilytettynä ja merkityskoulukunnan näkemys perustuu nimenomaan viestijöiden yhteiseen merkityksen rakentamiseen, johon vaikuttaa kaikkien elämänkokemus ja –tilanne. (Melnik ym., 2004 ; Åberg, 2000)

2.3 Organisaatioviestintä

Wiio (1978) määrittelee organisaatioviestinnän näin: *"Organisaatioviestintä on sellaista informaation käsittelyä, joka liittää organisaation osajärjestelmät toimintaan organisaation ja sen jäsenten tavoitteiden saavuttamiseksi sekä liittää organisaation sen toimintaympäristöön!"*

Kuten kaikilla yhteisöillä, organisaatiolla on useita eri sidosryhmiä, joiden kanssa se on vuorovaikutuksessa. Organisaatiolla on myös tarve kertoa, kuunnella, keskustella ja olla vuorovaikutuksessa ympäristön kanssa ja sen olemassaolon ehto on, että se pystyy täyttämään sidosryhmiensä tarpeita. Ympäristön kanssa vuorovaikuttaminen on hyvin tärkeää, koska se voi johtaa uusiin ideoihin, joita organisaation sisällä ei välttämättä olisi keksitty. (Juholin, 2009)

Organisaatioviestintää on jaoteltu monin eri tavoin ja yksi nykypäivänä useimmin käytettyjä jaotteluja on jako formaaliin ja informaaliin viestintään. *Formaali* viestintä on organisaation toimintaan liittyvää viestintää, kuten esimerkiksi työohjeita tai turvallisuusmääräyksiä ja *informaali* viestintä taas organisaation jäsenten välistä sosiaalista kanssakäymistä. Nykyään formaalin viestinnän määritelmään sisällytetään myös joissain tilanteissa viestintätapahtumat, kuten palaverit, formaalit puhelinkeskustelut ja videokonferenssit sekä sähköpostiviestit (Paasivaara & Lassenius, 2003), ja informaaliin viestintään taas informaalit puhelinkeskustelut ja videokonferenssit sekä sähköpostit (Pikkarainen ym., 2008).

2.4 Projektiviestintä

Projektin hallintaa ja projektiviestintää on käsitelty hyvin kattavasti PMBOK (A guide to the Project Management Body of Knowledge) –oppaassa. Sen mukaan projektiviestintä jakautuu eri ulottuvuuksiin (PMBOK, 2004):

1. Kirjallinen ja suullinen, kuunteleminen ja puhuminen
2. Projektin sisäinen ja ulkoinen viestintä (esim. media)
3. Formaali (esim. raportit) ja informaali viestintä (esim. spontaanit keskustelut)
4. Vertikaalinen (esim. esimies-alainen -viestintä) ja horisontaalinen viestintä (esim. vertaisviestintä)

2.4.1 Projektiviestinnän prosessit

PMBOK (2004) jakaa projektiviestinnän neljään eri prosessiin: Viestinnän suunnittelu (communication planning), tiedonjakaminen (information distribution), tulosten raportointi (performance reporting) ja hallinnollinen päättäminen (administrative closure). Näiden viestintäprosessien tulisi esiintyä projektin jokaisessa vaiheessa (PMBOK, 2004). Seuraavassa esitellään nämä prosessit tarkemmin.

1. Viestinnän suunnittelu

Kaikille projekteille on yhteistä tietämyksen jakamisen tarve koko projektin ajan kaikille sen osapuolille. Viestinnän suunnittelu tarkoittaa projektin sidosryhmien tietämyksen ja viestinnän tarvetta ja pyrkii vastaamaan seuraaviin kysymyksiin: Kuka tarvitsee mitäkin tietämystä? Koska he

tietämystä tarvitsevat? Kuinka tietämys heille tarjotaan ja kuka sen heille antaa? Kuitenkin tietämyksen tarpeet sekä sen jakamistavat vaihtelevat projekteittain hyvin paljon. Sidosryhmien tietämystarpeiden tunnistaminen ja niihin vastaaminen on yksi kriittinen tekijä projektin onnistumisen kannalta. (PMBOK, 2004)

Suurimmassa osassa projekteja viestinnän suunnittelu suoritetaan pääsääntöisesti projektin alkuvaiheessa, vaikka todellisuudessa viestinnän onnistumista tulisi arvioida koko projektin ajan sen tehokkuuden varmistamiseksi. Viestinnän suunnittelu on yleensä tiukasti sidoksissa organisationaaliseen suunnitteluun, koska organisaation rakenteella on suuri vaikutus projektin viestintävaatimuksiin. (PMBOK, 2004)

2. Tietämyksen jakaminen

Tietämyksen jakaminen tässä yhteydessä tarkoittaa tietämyksen siirtämistä sidosryhmille oikea-aikaisesti. Se sisältää viestintäsuunnitelman käyttöönoton ja valmiudet vastata odottamattomiin tietämyksen tarpeisiin projektissa. Tietämystä voidaan jakaa usein eri keinoin: suullisesti, projektipalavereissa, paperidokumentteina, teknisenä dokumentaationa, elektronisissa tietokannoissa, fakseina, sähköpostina, puheviesteinä, videokonferensseissa ja projektien intraneteissä. Viestintätaitojen merkitys on hyvin suuri projektiviestinnän onnistumiselle. On tärkeää, että viestin lähettäjä pystyy esittämään viestin selkeästi ja mahdollisimman yksiselitteisesti, jotta vastaanottaja pystyisi ymmärtämään sen riittävän hyvin eli on kykeneväinen käyttämään saamaansa tietämystä oikein omaa työtään suorittaessa. (PMBOK, 2004)

3. Suoritusraportointi

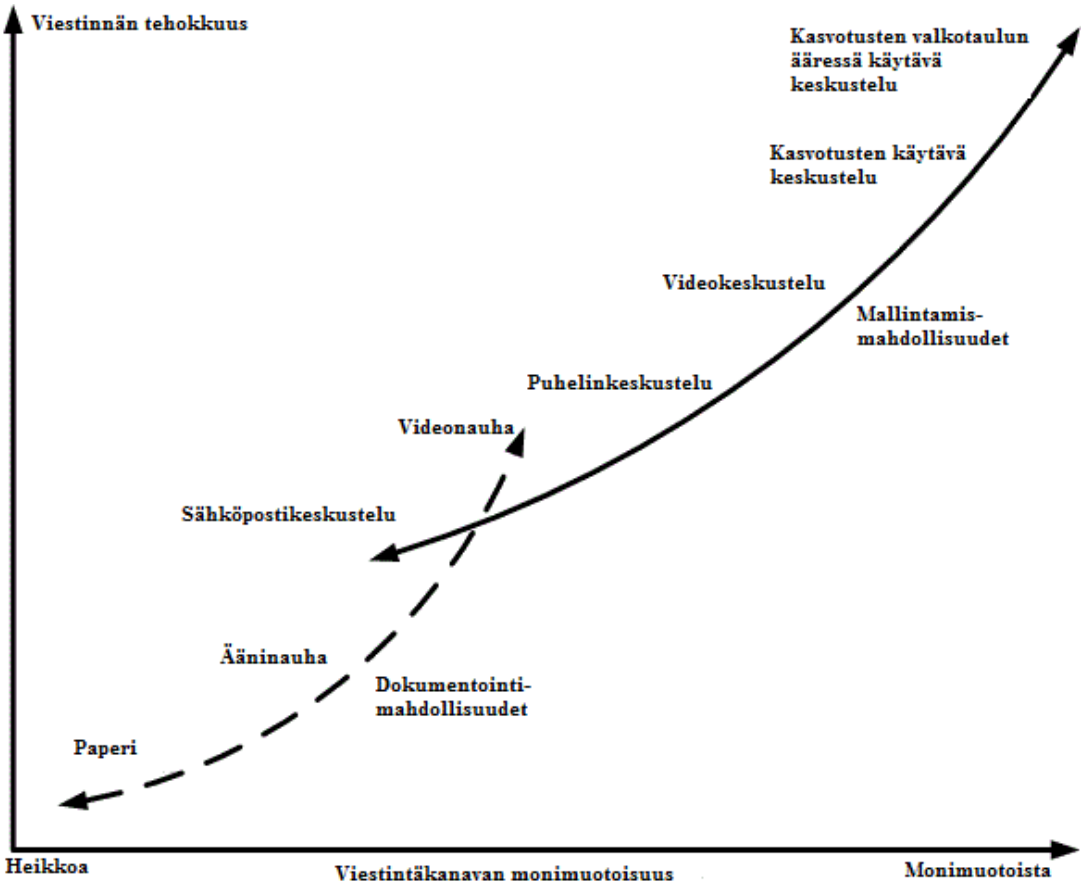
Suoritusraportointiprosessissa tarjotaan sidosryhmille tietämystä projektin edistymisestä ja resurssien käytöstä. Prosessi koostuu seuraavista osa-alueista: Projektin tilan raportointi, projektin edistymisen raportointi ja projektin tulevan edistymisen ennustaminen. Yleisesti ottaen tulosten raportoinnissa tulisi aina saada tietoa tavoitteista, aikataulusta, kustannuksista, laadusta ja projektin tuloksista sekä usein myös riskeistä ja hankinnoista. (PMBOK, 2004)

4. Hallinnollinen päättäminen

Projektin tai sen vaiheen päättäminen on tärkeää aina riippumatta siitä, loppuiko projekti tai sen vaihe tavoitteiden saavuttamiseen tai keskeytykseen. Tässä prosessissa esitellään ja dokumentoidaan saavutetut tulokset asiakkaalle projektiasiakirjojen muodossa. Tärkeää on varmistaa, että dokumentaatio vastaa lopullista tuotetta ja arvioida projektin onnistumista, tehokkuutta ja opittuja asioita sekä dokumentoida tämä tietämys tulevaisuuden varalle. Kaikissa projekteissa tulisi muistaa, että hallinnollinen päättäminen ei tapahdu pelkästään projektin lopussa, vaan jokaisen vaiheen loputtua tulisi suorittaa samat toimenpiteet, jotta käyttökelpoista tietämystä ei katoaisi. (PMBOK, 2004)

2.4.2 Ohjelmistoprojektin viestintämuodot

Ambler (2010) on käsitellyt erilaisia viestintämuotoja ketterässä kehityksessä käyttäen Alistair Cockburnin 2002 esittelemää alkuperäistä kaaviota seuraavan kaavion pohjana.



Kuva 1 Viestintämuodot (ketterässä) ohjelmistokehitysprojektissa alunperin Alistair Cockburnin (2002) mukaan Amblerin muokkaamana (2010)

Kuten kuvasta 1 selvästi näkyy kasvokkaisviestintää kohti mentäessä viestinnän tehokkuus ja monimuotoisuus lisääntyy huomattavasti verrattuna paperilla tapahtuvaan viestintään tai dokumentaatioon. Tämän työn laajuus ei anna mahdollisuutta tarkastella näitä kaikkia viestinnän muotoja, vaan tämä työ on rajattu käsittelemään nimenomaan kasvokkaisviestintää sen ollessa kaikista kriittisin viestinnän muoto ketterän projektitiimin sisällä. (Ambler, 2010)

3 KETTERÄT MENETELMÄT

Viime vuosina on huomattu, että ohjelmistoyritysten tulee pystyä reagoimaan dynaamisiin markkinoihin, uudenlaisiin ja muuttuviin asiakasvaatimuksiin sekä teknisiin innovaatioihin (Beck ym., 2008; Lycett, Macredie, Patel, & Paul, 2003; Pikkarainen ym., 2008). Chown ja Caon (2008) mukaan perinteinen kehitysprosessi ei ole riittävä nykyiseen ohjelmistokehitysympäristöön. Prosessin puutteista johtuen viime vuosina on noussut esiin ketteriä kehitysmenetelmiä perinteisten menetelmien tilalle, koska perinteisten menetelmien on huomattu olevan puutteellisia ja johtavan usein projektien viivästymiseen, epäonnistumiseen, hylkäämiseen tai torjumiseen (Chow & Cao, 2008). Myös menestyneiden projektien seurauksena syntyneet tuotteet vaativat usein kalliita päivityksiä ja jatkuvaa ylläpitoa (Chow & Cao, 2008). Tästä syystä ketteriä menetelmiä on ehdotettu tavaksi vastata muutokseen, lyhentää kehitysaikaa ja parantaa kommunikaatiota ja yhteistyötä erityisesti tapauksissa, missä ajoitus on kriittinen kilpailuetu yritykselle (Karlström & Runeson, 2006). Seuraavaksi käsitellään tarkemmin mitä ovat ketterät menetelmät ja perehdytään kahteen suosituimpaan ketterään menetelmään eli Scrumiin ja XP:hen syvällisemmin, käymällä läpi niille ominaisia piirteitä ja kuvaamalla niiden prosessimallit.

3.1 Mitä ovat ketterät menetelmät?

Ketterien kehitysmenetelmien (Agile Methods) ketteryys tulee niiden paremmasta valmiudesta suhteessa perinteisiin menetelmiin vastata muuttuviin vaatimuksiin projekteissa, missä lopputulos ei ole alusta asti kunnolla selvillä (Anderson, 2003). Ketterä lähestymistapa sai alkunsa, kun 17 eri ketteriin menetelmiin perehtynyttä henkilöä perustivat Agile Alliancen ja keräsivät

ketterien menetelmien parhaimmat käytännöt yhteen Agile-manifestiksi (Misra, Kumar, & Kumar, 2009). Agile-manifesti (Beck ym., 2008) tiivisti ketterän kehityksen neljään arvoon:

1. Yksilöt ja vuorovaikutus ennen prosesseja ja työkaluja
2. Toimiva ohjelmisto ennen dokumentaatiota
3. Asiakasyhteistyö ennen sopimusneuvotteluja
4. Muutokseen vastaaminen ennen suunnitelman noudattamista

Ketterien menetelmien filosofia perustuu parhaisiin käytäntöihin (best practices), joiden on todettu tehostavan ohjelmistokehitystä ja joista on poistettu ohjelmistokehitystä rajoittavia käytänteitä (Misra ym, 2009). Ketteristä kehitysmenetelmistä saadut kokemukset lupaavat paitsi nopeampaa kehitystä niin myös viestinnän ja yhteistyön parantumista tiimin sisällä sekä tiimin, asiakkaan ja muiden sidosryhmien välillä (Anderson, 2003).

Ketteriä menetelmiä on useampia, mutta kirjallisuudessa mainitaan yleensä Extreme Programming (XP) (Beck & Anders 2004), Scrum (Schwaber, 1995), Feature-Driven Development (FDD) (Coad, Luca, & Lefebvre, 1999), Dynamic System Development (DSDM) (DSDM Consortium, 2010), Adaptive Software Development (ASD) (Highsmith, 2002), Crystal (Cockburn, 2004) sekä Lean Software Development (LD) (Poppendieck & Poppendieck, 2003). Työn esimerkeiksi on valittu Scrum ja XP, koska ne ovat kaksi eniten käytettyä ketterää menetelmää (Fitzgerald ym., 2006) ja koska niiden lähestymistavat poikkeavat toisistaan verrattain paljon Scrumin ollessa projektinhallinnallinen ja XP:n taas käytäntöorientoitunut (practice oriented) menetelmä (Abrahamsson, Salo, Ronkainen, & Warsta, 2002).

3.2 Scrum

Kuten on jo aiemmin useampaan otteeseen mainittu, ohjelmistokehitys ei ole etukäteen tarkasti suunniteltavissa oleva prosessi (Schwaber, 1995). Scrum on yksi suosituimmista ratkaisumalleista kyseiseen ongelmaan. Schwaberin (1995) mukaan Scrum olettaa ohjelmistokehitysprosessin olevan huonosti ennustettava, monimutkainen prosessi, joka voidaan kuvata prosessin alussa vain pääpiirteittäin. Teollisesta prosessinhallinnasta on ohjelmistokehityksen avuksi otettu käyttöön kaksi erilaista prosessikäsitettä: teoreettinen eli täydellisesti määritelty ja empiirinen eli musta laatikko (black box). Mustalla laatikolla Schwaber tarkoittaa, että itse kehitysprosessi on määrittelemätön musta laatikko, joka toimii itsenäisesti sekä hallitsemattomasti. Musta laatikko – malli perustuu siihen, että kehittäjätiimi on riittävän ammattitaitoinen, jotta kunnollisilla työkaluilla ja parhailla käytänteillä varustettuna kaaoksenomainen prosessi tuottaa hyvin toimivan tuotteen tavoiteajassa. (Schwaber, 1995)

Scrum on tarkoilla säännöillä varustettu projektinhallinnallinen iteratiivinen ja inkrementaalinen ohjelmistokehityksen prosessimalli eli se jakautuu useisiin kehitysvaiheisiin (sprintteihin), jotka noudattavat täsmälleen samaa kaavaa ja toteuttavat joka jaksossa uuden valmiin tuotteen osan. Scrum määrittelee ohjelmistokehitysprosessin väljänä kokoelmana toimintoja, jotka koostuvat hyvin tunnetuista toimivista työkaluista ja joiden suorittamiseen tarvitaan hyvin motivoitunut kehittäjätiimi. (Schwaber, 1995) Seuraavaksi perehdytään tarkemmin Scrumin ominaispiirteisiin ja esitellään Scrumin prosessimalli.

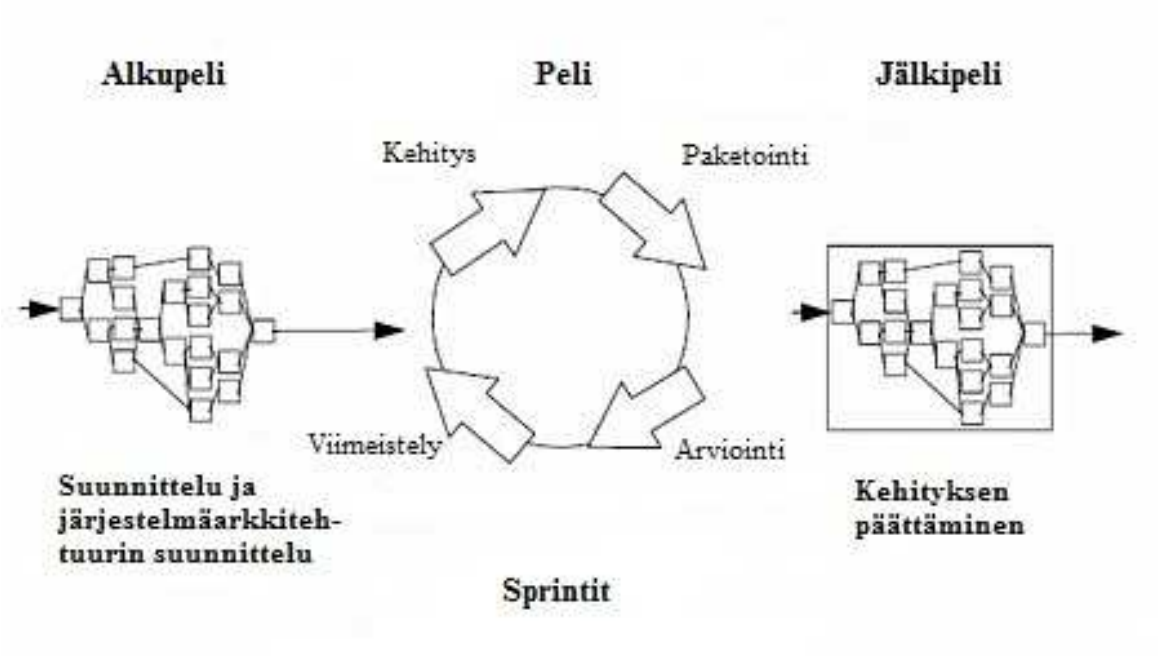
3.2.1 Scrumin ominaispiirteitä

Scrum-prosessille ominaisia piirteitä ovat (Schwaber, 2004):

- Kaaoksen omainen, itseohjautuva inkrementaalinen ja iteratiivinen prosessi (sprints)
- Muutokseen vastaaminen (Schwaber, 1995)
- Scrum-mestari (tiimin jäsen, joka pyrkii pitämään tiimin toiminnan niin tehokkaana kuin mahdollista)
- Läsnä oleva asiakas
- Itseohjautuva tiimi ja täydelliseen tiiminsisäiseen läpinäkyvyyteen pyrkiminen
- Scrum-säännöt: sprintin suunnittelukokous (Sprint Planning Meeting), päivittäinen Scrum-kokous (Daily Scrum Meeting), sprintit (Sprints), sprintin auditointikokous (Sprint Review Meeting) ja sprintin jälkiarviointi (Sprint Retrospective Meeting)

3.2.2 Scrumin prosessimalli

Scrum-prosessi jakautuu karkeasti kolmeen prosessiin (kuvassa 1): Alkupeli (Pregame), Peli (Game) ja jälkipeli (Postgame). Nämä kaikki kolme vaihetta jakautuvat pienempiin prosesseihin. Alkupeli sisältää suunnittelun (Planning) ja arkkitehtuurin suunnittelun (high level design), peli sisältää sprintit (Sprints), joihin kuuluu kehitys (Development), paketointi (Wrap), katselmus (Review) ja viimeistely (Adjust) ja jälkipeliin kuuluu kehityksen päättäminen (Closure).



Kuva 2 – Scrum-prosessimalli (Schwaber, 1995)

Alkupelin suunnitteluprosessi koostuu ensimmäisen julkaisun tiedossa olevien vaatimusten sijoittamisesta tehtävälistaan (product backlog), aikataulu- ja kustannusennusteen laatimisesta, järjestelmän käsitteellisestä mallintamisesta ja analysoinnista (uutta järjestelmää rakennettaessa) ja rajallisemmasta analysoinnista (olemassa olevaa järjestelmää parannettaessa). Arkkitehtuurin suunnittelu koostuu tehtävälistan vaatimusten toteuttamisen suunnittelusta sekä korkean tason arkkitehtuurin suunnittelusta. (Schwaber, 1995)

Pelivaiheen sprintit ovat läpivienniltään identtisiä, yleensä 1-4 viikon mittaisia kehityssyklejä, joissa kehitetään uusi julkaisu jostakin lopullisen järjestelmän osasta. Sprintit pohjautuvat jatkuvan ajankäytön, vaatimusten, laadun, kustannusten ja kilpailun hallintaan. (Schwaber, 1995) Jälkipelivaiheessa suoritetaan kehityksen päättäminen. Se sisältää julkaisun valmistelun,

lopullisen dokumentaation, ennakkojulkaisun ja sen testauksen sekä lopullisen julkaisun (Schwaber, 1995).

3.3 Extreme Programming (XP)

Extreme Programming (XP) menetelmä on Beckin ja Andersin (2004) mukaan kevyt, tehokas, vähäriskinen, joustava, hyvin ennustettava, tieteellinen ja hauska tapa kehittää ohjelmistoja. XP on puhtaasti ohjelmointilähtöinen menetelmä, jossa ohjelmointi nähdään ohjelmistokehityksen tärkeimpänä osa-alueena (Beck & Anders 2004). Beckin ja Andersin (2004) mukaan ohjelmistokehityksessä kaikki päätyy lopulta ohjelmointiin: useimmiten kommunikointi tapahtuu koodin kautta, uuden järjestelmän osan ymmärtäminen opiskellaan koodista, uusien osien elinkaari ja käytös määritellään testeissä, jotka myöskin ovat koodia, virheraportit tulevat usein koodina ja optimointi tapahtuu lähdekoodia hiomalla. (Beck & Anders 2004)

XP eroaa perinteisistä menetelmistä hyvin pitkälti samoilta osa-alueilta kuin Scrumkin, mutta sen lähestymistapa itse kehitykseen poikkeaa Scrumista verrattain paljon. Scrumin tavoin XP eroaa perinteisistä menetelmistä iteratiivisuudellaan sekä inkrementaalisuudellaan, missä konkreettista jatkuvaa palautetta saadaan hyvin aikaisessa vaiheessa. Myös osien implementointi on joustavaa ja jatkuva muutokseen vastaaminen on yksi sen kulmakivistä, kuten myös läsnä olevan (on-site) asiakkaan tärkeys kehityksen seurannassa ja jatkuvan palautteen antajana. Scrumista XP eroaa nimenomaan koodikeskeisyyden osalta, koska se pohjaa hyvin paljon testaukseen ja erityisesti automaatiotestaukseen. (Beck & Anders 2004)

3.3.1 XP:n ominaispiirteitä

XP:llä on sille ominaisia piirteitä aivan kuten kaikilla muillakin ohjelmistokehityksen menetelmillä. Erityisesti XP:lle ominaisia piirteitä ovat (Leffingwell, 2007):

- 5-10 ohjelmoijan tiimit, jotka työskentelevät samassa tilassa ja johon kuuluu myös asiakas
- Kehitys tuottaa lyhyissä iteraatioissa julkaisukelpoisia, toiminnallisuutta sisältäviä järjestelmän osia
- Vaatimukset kuvataan tarinoina (story), jotka sisältävät joukon asiakkaan vaatimuksia
- Ohjelmoijat työskentelevät pareittain, noudattavat tiukkaa koodausstandardia ja testaavat ohjelmat itse
- Järjestelmän vaatimukset, arkkitehtuuri ja suunnittelu muototuvat projektin aikana

Beck ja Anders (2004) määrittelevät XP:lle viisi ydinarvoa, kymmeniä pääperiaatetta sekä 12 avainkäytäntöä, jotka muodostavat menetelmän ydinfilosofian. XP:n ydinarvot ovat (Beck & Anders, 2004):

1. Viestintä (Communication)
2. Yksinkertaisuus (Simplicity)
3. Palaute (Feedback)
4. Rohkeus (Courage)

5. Kunnioitus (Respect)

Nämä ominaispiirteet kokoavat yhteen XP:n ydinfilosofian, joka koostuu tehokkaasta tiimin sisäisestä kommunikaatiosta, vain tarvittavan koodaamisesta, jatkuvasta ja välittömästä palautteesta, rohkeudesta kirjoittaa testit ennen koodaamista, yksinkertaisimman toteutustavan löytämisestä sekä tiimin sisäisestä kunnioituksesta (Leffingwell, 2007).

Beck ja Anders (2004) ovat listanneet XP:n pääperiaatteita useita kymmeniä, mutta Dean Leffingwell on poiminut näistä mielestään neljä tärkeintä tarkasteltavaksi kirjassaan. Nämä neljä tärkeintä pääperiaatetta ovat (Leffingwell, 2007):

1. Inhimillisyys (Humanity)
2. Reflektointi (Reflection)
3. Laatu (Quality)
4. Lapsen askeleet (Baby Steps)

Inhimillisyys tarkoittaa sitä, että ohjelmistot kehitetään ihmisiä varten ja niitä kehittämässä ovat ihmiset. Reflektointi niin ikään liittyy projektia toteuttaviin ihmisiin, koska tämä XP:n periaate kannustaa tiimin jäseniä miettimään syytä työskentelylleen pelkän tavoitteen sijasta. Sen tarkoituksena on tiimin kehittyminen paremmaksi ja tehokkaammaksi. XP prosessissa laatu on kiinteä osa XP-prosessia. XP:n käytännöt luovat ympäristön, jossa oikein toimittuna laatu syntyy itsestään, eikä kehittäjien tarvitse sitä erikseen miettiä. Lapsen askeleilla XP-prosessissa tarkoitetaan tiimin kehittymistä koko prosessin ajan. Rakennettaessa jotain, jonka lopputulos ei ole selvillä, on selvää, ettei kyseistä projektia pysty toteuttamaan prosessimaisella suoraviivaisella tavalla, vaan

XP:n periaate on tiimin tehokas kehittyminen ja oppiminen koko prosessin ajan. (Leffingwell, 2007)

Kaikki ohjelmistokehitysmenetelmät sisältävät niin sanottuja parhaita käytäntöjä ja Leffingwell (Leffingwell, 2007) on listannut XP:n parhaita käytäntöjä 13 kappaletta:

1. **Yhdessä istuminen (Sit Together):** Tiimin tulee työskennellä samassa täysin avoimessa tilassa.

2. **Koko tiimi (Whole Team):** Kaikkien tiimin jäsenten tulee olla sitoutuneita kyseiseen projektiin sataprosenttisesti.

3. **Informatiivinen työympäristö (Informative Workspace):** Projektin edistymiseen liittyvän tiedon tulee olla kaikkien nähtävissä aina ja vaivattomasti.

4. **Energiatehokas työskentely (Energized Work):** Normaali 40-tuntinen työviikko ilman ylitöitä on tehokkain tapa työskennellä XP-projektissa.

5. **Pariohjelmointi (Pair Programming):** Kaksi ohjelmistokehittäjää työskentelee saman tietokoneen äärellä.

6. **Tarinat (Stories):** Tapa kuvata käyttäjävaatimukset parhaimman mahdollisen vaatimusten ymmärtämisen saavuttamiseksi.

7. **Viikkosykli (Weekly Cycle):** XP:n määritelmässä sen sykli eli iteraatio (vrt. Scrumin Sprintti) kestää yhden viikon, joten tarinat tulee jakaa niin pieniin osiin, että ne on mahdollista toteuttaa viikossa.

8. **Neljännessykli (Quarter Cycle):** Viikkosykliä karkeampi aikataulun suunnittelu erityisesti isompien toimitusten kohdalla, missä kehitysaikataulu jaetaan neljään osaan.

9. **Väljyys (Slack):** XP-prosessin ollessa hyvin tiukka tarvitsee se väljyyttä aikatauluun yllättävien ongelmien varalta esimerkiksi tehtävien priorisoinnin avulla, jolloin osa alemman prioriteetin tehtävistä voidaan siirtää tarpeen vaatiessa seuraavaan iteraatioon.

10. **Testattu 10 minuutissa (Ten-Minute Build):** XP-tiimin tulisi pystyä käynnistämään koko järjestelmä ja ajamaan sen testit 10 minuutissa.

11. **Jatkuva integraatio (Continous Integration):** Tuotteen osia yhdistetään ja testataan useita kertoja päivässä.

12. **Testit ensin –ohjelmointi (Test-First Programming):** XP:ssä testit tulee kirjoittaa ennen itse koodin kirjoittamista.

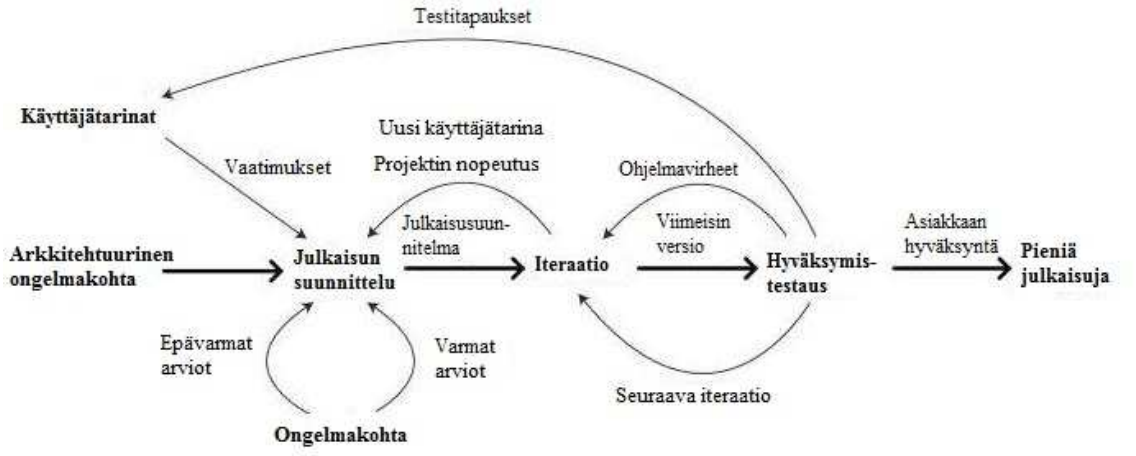
13. **Inkrementaalinen kokonaissuunnittelu (Incremental Design):** Järjestelmän kokonaissuunnittelu inkrementaalisesti varmistaa lopullisen oikeanlaisen arkkitehtuurin syntymisen.

Nämä 13 käytäntöä ovat todella kriittisiä XP projektin onnistumiselle (Leffingwell, 2007). Viestinnän kannalta keskeisiä käytäntöjä ovat yhdessä istuminen, informatiivinen työympäristö, pariohjelmointi, tarinat ja viikkosykli. Niitä käsitellään tarkemmin luvussa 4.

3.3.2 XP:n prosessimalli

XP:n prosessi rakentuu lineaarisen viisivaiheisen mallin ympärille (kuva 2) (Leffingwell, 2007):

1. Arkkitehtuurinen erityistehtävä (Architectural Spike) ja käyttäjätarinat (User Stories)
2. Julkaisun suunnittelu (Release Planning)
3. Iteraatio (Iteration)
4. Hyväksymistestaus (Acceptance Tests)
5. Pienet julkaisut (Small Releases)



Kuva 3 - XP prosessimalli (Leffingwell, 2007)

XP-prosessissa julkaisun suunnittelu perustuu käyttäjätarinoihin ja arkkitehtuurin erityistehtävien tulokseen. Käyttäjätarinat kuvaavat käyttäjän vaatimuksia seuraavalle julkaisulle. Arkkitehtuuriset muutokset tarkoittavat

työtä, missä kehittäjän täytyy muokata järjestelmän arkkitehtuurista pohjaa, etsiä korjattavaa lähdekoodista tai etsiä uusia teknologioita toteuttaakseen asiakasvaatimuksia. (Leffingwell, 2007)

Julkaisun suunnittelu antaa pohjan iteraatioille, joissa julkaisu tuotetaan. Iteraatioita seuraa hyväksymistestausvaihe, jossa iteraatioissa tuotetut järjestelmän osat testataan käyttäjätarinoihin perustuvilla ennakkoon kirjoitetuilla testeillä. Hyväksymistestauksen jälkeen aloitetaan uusi iteraatio, kunnes kaikki käyttäjätarinoiden vaatimukset on toteutettu. Tämän jälkeen lopullinen julkaisu testataan ja hyväksytetään asiakkaalla. (Leffingwell, 2007)

4 VIESTINTÄ SCRUM- JA XP-MENETELMISSÄ

Seuraavissa kohdissa käydään lyhyesti läpi viestinnän keskeinen merkitys ohjelmistokehitysprojektissa sekä eritellään lyhyesti ketterän tiimin erilaisia viestintämuotoja. Tämän jälkeen perehdytään syvemmin pääsääntöisesti Scrumissa ja XP:ssä määriteltyihin ketteriin viestintäkäytänteisiin ja niillä saavutettaviin etuihin ohjelmistokehityksessä perinteisiin käytänteisiin nähden.

4.1 Viestinnän keskeinen merkitys ohjelmistokehityksessä

Viestintä on kaikkien yhteistyökäytäntöjen ja –prosessien yksi keskeisimmistä tekijöistä (Paasivaara & Lassenius, 2003). Ohjelmistoprojekteissa viestintä on tärkeä menestystekijä erityisesti silloin, kun projektit ovat alttiita muutoksille ja tiimit vaihtuvia (Stelzer, Mellis, & Wiley, 1999). Viestintä on tärkeä tapa jakaa tietoa ja luoda tiiminsisäisiä toimintamalleja monimutkaisessa ohjelmistokehitysympäristössä (Espinosa & Carmel, 2003). Ketterien menetelmien on huomattu tarjoavan ratkaisuja viestintäongelmiin erityisesti ohjelmistokehitykseen orientoituneissa yrityksissä (Pikkarainen ym., 2008; Turner, 2003). Aiemmin luvussa 1 viitattiin Juholinin (2009) ajatukseen siitä, että vuorovaikuttaminen on tärkeää, koska se voi johtaa uusiin ideoihin, joita organisaation sisällä ei välttämättä olisi keksitty. Mielestäni Juholinin ajatus johtaa hyvin ketterän kehityksen alkulähteille ja sen tarpeelle ohjelmistokehityksessä, koska perinteisten menetelmien on huomattu olevan viestinnältään tehottomia ja kalliita ja johtavan usein projektien epäonnistumiseen tai keskeytymiseen (Chow & Cao, 2008), kun taas ketterät menetelmät nojaavat nimenomaan tiiviiseen viestintään tiimin sisällä.

Ketterien menetelmien suhde viestintään kiteytyy mielestäni Agile Alliancen 1. arvossa (Beck ym., 2008): ”Yksilöt ja vuorovaikutus ennen prosesseja ja

työkaluja". Melnikin ja Maurerin (2004) mukaan ketterien menetelmien viestintä pohjautuu nimenomaan tiimin jäsenten keskinäiseen ja erityisesti kasvokkain tapahtuvaan informaaliin viestintään. He esittävät myös, että keskustelu on ketterän kehityksen tärkein työkalu. Tämä aspekti tuo oman haasteensa liittyen projektin ihmisiin ja viestinnän tehokkuuden parantamiseen ja varmistamiseen, sillä viimeistään tämän ajatuksen myötä ihmisiä ei voida enää nähdä vain tayloristisen koneiston helposti korvattavina osina, vaan yksilöinä ja tärkeinä voimavaroina yritykselle (Melnik & Maurer, 2004).

Pikkaraisen ym. (2008) tutkimuksessa tutkittiin ketterien menetelmien vaikutusta tiimin viestintään. Tutkimus osoitti selvästi, että ketterillä menetelmillä on suuri vaikutus tiiminsisäisen viestinnän paranemiseen. Myös Melnikin ym. (2004) tutkimus osoitti, että riittävän kompleksisissa (esim. juuri ohjelmistokehitys) ympäristöissä johdannossa mainittu perinteinen näkemys tietämyksestä objektina (knowledge-as-object) ei ole riittävä vaan modernimpi näkemys tietämyksestä ihmissuhteena (knowledge-as-relationship) on selvästi tehokkaampi näkemys ohjelmistokehitystä ajatellen.

4.2 Ketterät viestintäkäytänteet ja niiden edut

Seuraavaksi olen listannut kirjallisuuden pohjalta mielestäni kriittisimmät Scrumin ja XP:n ketterät viestintäkäytänteet, jotka ovat pääasiallisesti kasvokkaisviestintään pohjautuvia käytäntöjä. Kasvokkaisviestinnän tullessa vastaan kaikkien tutkittujen ketterien käytäntöjen kohdalla voisi sanoa, että se on ennemminkin yksi ketterän kehityksen perusarvoista kuin pelkkä käytäntö muiden joukossa.

Kasvokkaisviestintä (face-to-face communication): Kasvokkaisviestintä on yksi tärkeimmistä periaatteista sekä Scrumissa että XP:ssä (Leffingwell, 2007; Schwaber, 1995). Henttosen ja Blomqvistin (2005) mukaan se on paras tapa rakentaa luottamusta ihmisten kesken ja tällä tavoin säilyttää hyvä tuottavuus ohjelmistokeskeisissä yrityksissä.

Kasvokkaisviestinnän merkitys ketterälle kehitykselle on Pikkaraisen ym. (2008) sekä Melnikin ja Maurerin (2004) tutkimusten mukaan elintärkeä. Melnik ja Maurer (2004) esittävät, että kasvokkaisviestintä tarjoaa rikkaampaa keskustelua sen sisältämien useiden eri viestintäkanavien vuoksi. Myös Ambler (2010) on samoilla linjoilla ja tarkentaa näkemystä, että miksi näin on; hänen mukaansa henkilökohtaisen kontaktin poistuminen viestinnästä pudottaa viestinnän tasoa, koska sen mukana viestijät menettävät tiedostetut ja tiedostamattomat äänettömät vihjeet, elekielen tulkitsemisen sekä monikanavaisen aistimisen. Näiden lisäksi hänen mukaansa poistuu mahdollisuus tulkita sanattomien vihjeiden perusteella, mitä toinen osapuoli todellisuudessa tarkoittaa. Lisäksi poistuu mahdollisuus suoraan kysymykseen vastaamiseen, mikä on vastapuolelle suoraa palautetta siitä, kuinka hyvin asia on todellisuudessa ymmärretty.

Melnik ja Maurer (2004) esittävät myös, että ketterässä kehityksessä tiedonsiirtoketju pyritään minimoimaan juuri kasvokkaisviestinnällä, verrattuna esimerkiksi dokumentaation kautta siirrettävään tietoon. Tämä johtaa heidän mukaansa todella nopeaan tiedonjakoon ja sitä kautta suoraan projektin parempaan menestykseen, kun taas pitkät ketjut usein johtavat olennaisen tiedon katoamiseen. Tämä on erityisen ratkaisevaa ohjelmistokehitysympäristössä, koska tuotteet ovat kompleksisia ja asiakkaan vaatimusten ymmärtäminen on yleensä hyvin vaikeaa ja alussa jopa mahdotonta. (Melnik & Maurer, 2004)

Avoin työskentelytila (open office space): Avoin työskentelytila on toinen sekä Scrumin että XP:n periaatteista löytyvä periaate. XP:ssä on menty vielä Scrumiakin pidemmälle, kuten XP:n käytäntö ”Yhdessä istuminen” kertoo, sillä XP:n työtilan tulisi olla täysin avoin ilman mitään väliseiniä (Leffingwell, 2007; Schwaber, 1995). Espinosan ja Carmelin (2003) mukaan samassa tilassa työskentelevät tiimit ovat merkittävästi tuottavampia kuin jos tiimin jäsenet työskentelisivät omissa huoneissaan. Pikkaraisen ym. (2008) tutkimuksessa avotila nähtiin tehokkaana tapana parantaa informaalia viestintää kehitysprojektin aikana.

Pikkaraisen ym. (2008) tutkimus osoitti, että avotila on tehokkain työtila ohjelmistokehitystä varten. Samassa tutkimuksessa huomattiin myös, että avotila vähentää formaalin dokumentaation tarvetta. Heidän tutkimuksensa projekteissa työskennelleet kehittäjät tietyissä tilanteissa jopa kyseenalaistivat koko dokumentaation tarpeen, koska he eivät havainneet sitä, kuka dokumentaatiota lukisi ja milloin, joten he eivät dokumentoineet näissä tilanteissa ollenkaan.

Melnik ja Maurer (2004) päätyivät hyvin pitkälti samaan lopputulokseen dokumentaation osalta, mutta he tekivät myös sellaisen huomion, että faktatiedoksi luettava informaatio kannattaa dokumentoida, koska faktat eivät muutu samalla tahdilla kuin perinteinen dokumentaatio. Espinosan ja Carmelin (2003) tutkimuksessa huomattiin, että samassa huoneessa työskentelevät tiimit ovat paljon tuottavampia kuin eri huoneissa työskentelevät. Tämäkin on seurausta samassa tilassa työskentelyn aiheuttamasta yhteistoiminnasta ja sen aiheuttamasta interaktiivisesta viestinnästä (Pikkarainen ym., 2008).

Pikkaraisen ym. (2008) tutkimuksesta selviää myös, että avotoimisto stimuloi kasvokkaisviestintää, jolloin päivittäinen ongelmanratkaisu helpottuu sekä

tehostuu huomattavasti. Näin ollen avotilan huomattiin aiheuttavan selkeästi paremmat edellytykset avoimelle keskustelulle ja ongelmanratkaisufoorumille. Lisäksi kyseisessä tutkimuksessa selvisi, että avotilan puute aiheutti ongelmia työskentelyssä hyvin äkkiä. Heidän tutkimuksesta selvisi myös, että ainoa negatiivinen asia avotilassa oli keskittymisen herpaantuminen esimerkiksi tilanteissa, joissa osa tiimin jäsenistä piti kokousta samassa tilassa kuin kehittäjä yritti kirjoittaa koodia, mutta tätä ei nähty ongelmana, koska avotila nähtiin muuten niin tehokkaana kehitystä edistävänä käytänteenä.

Läsnä oleva asiakas (on-site customer): Läsnä oleva asiakas on yksi yleisimpiä ketteriä periaatteita ja se on myös Scrumin ja XP:n ydinperiaatteita. Asiakkaan tulisi olla päivittäin koko tiimin käytettävissä ja osa koko tiimiä, koska Scrumin ja XP:n prosessi vaatii läheistä asiakasyhteistyötä koko kehityksen ajan (Grisham & Perry, 2005; Schwaber, 1995). Asiakkaan rooli on erityisen suuri nimenomaan sprinttien suunnittelussa ja hyväksymistestauksessa (Leffingwell, 2007). Samalla kun läsnä oleva asiakas on yksi tärkeimpiä ketteriä periaatteita on se myös yksi haastavimpia, koska itse asiakkaan työ on verrattain haastavaa ja vaativaa (Koskela & Abrahamsson, 2004).

Läsnä oleva asiakas on nähty yhdeksi kriittisimmistä tekijöistä useammassa tutkimuksessa (Koskela & Abrahamsson, 2004; Pikkarainen ym., 2008). Koskelan ja Abrahamssonin (2004) tutkimuksessa haastatellut kehittäjät arvottivat läsnä olevan asiakkaan yhdeksi tärkeimmistä käytännöistä projektin aikana. Kehittäjien mukaan ongelmanratkaisu oli nopeaa, koska heti ongelman tai kysymyksen ilmetessä pystyi konsultoimaan asiakasta. Lisäksi asiakkaan päivittäinen osallistuminen kehittämiseen nähtiin tärkeänä ja viikoittainen osallistuminen nähtiin täysin riittämättömänä (Koskela & Abrahamsson, 2004). Samoihin tuloksiin on päätynyt myös Grisham ja Perry (2005), joiden mukaan asiakkaan tulisi olla päivittäin koko tiimin käytettävissä. Läsnä oleva asiakas

nähtiin myös kehittäjien motivaatiota nostavana, koska läsnä oleva asiakas on suhteellisen suuri panostus heidän työhönsä (Koskela & Abrahamsson, 2004).

Päivittäiset palaverit: Sekä Scrumin että XP:n periaatteisiin kuuluvat päivittäiset tehokkaat palaverit (Leffingwell, 2007; Schwaber, 1995). Tässä esimerkkinä käsitellään Scrumin päivittäistä palaveria (daily Scrum), koska se on määritelty Scrumin säännöissä erittäin hyvin ja selkeästi. Päivittäinen Scrum palaveri on lyhyt 15 minuutin mittainen palaveri, jossa kehittäjät vastaavat kolmeen kysymykseen: Mitä olet tehnyt tässä projektissa viimeisen Scrum-palaverin jälkeen? Mitä aiot tehdä ennen seuraavaa tapaamista? Onko sinulla jotain ongelmia tai esteitä kyseisten tehtävien suorittamisessa? (Marcal ym., 2007) Tällä tavoin pyritään varmistamaan projektin eteneminen aikataulussa mahdollisimman tarkasti sekä varmistaa toimiva tiimityöskentely ja tiedonjakaminen (Melnik & Maurer, 2004).

Päivittäisten palavereiden on todettu vaikuttavan positiivisesti johdon resurssien allokointiin (Pikkarainen ym., 2008). Äsken mainittu päivittäinen Scrum-palaveri on hyvä esimerkki tästä, koska johto ei siihen osallistu, mutta tehtävien jako toimii itsestään. Scrum-palaverin idea ei ole ainoastaan kertoa, missä vaiheessa työssä ollaan menossa, vaan palaveri on tarkoitettu erityisesti itseohjautuvan tiimin tiedonjakamiseen, jotta heillä säilyisi projektin kokonaiskuva paremmin ja he pystyisivät koordinoimaan tehtäviä mahdollisimman tehokkaasti keskenään. (Deemer & Benefield, 2007)

Pikkaraisen ym. (2008) tutkimus osoitti, että päivittäin tapahtuva ongelmanratkaisu lisää tehokkuutta ohjelmistokehitysprojekteissa. Heidän tutkimuksessaan selvisi myös, että kehittäjät kokivat päivittäiset palaverit tehokkaammiksi kuin esimerkiksi viikoittaiset palaverit.

Iteraatioiden suunnittelukokoukset (Scrum: Sprint Planning, XP: Planning Game): Iteraatioiden suunnittelukokoukset ovat myöskin yhteinen periaate Scrumille ja XP:lle. Iteraatioiden suunnittelupalaverien avulla varmistetaan tiedonkulku koko tiimille siitä, mitä seuraavassa iteraatiossa ollaan tekemässä (Pikkarainen ym., 2008). Iteraatioiden suunnittelukokouksissa päätetään asiakkaan kanssa mitä seuraavassa iteraatiossa toteutetaan (Leffingwell, 2007; Schwaber, 1995). Näin ollen asiakkaan merkitys iteraatioiden suunnittelulle on todella kriittinen (Koskela & Abrahamsson, 2004).

Pikkaraisen ym. (2008) tutkimuksessa iteraatioiden suunnittelukokoukset paransivat tiedonkulkua sekä XP- että Scrum-projektissa. Ne vähensivät huomattavasti hämmennystä kehittäjien keskellä siitä, mitä seuraavaksi tulisi toteuttaa. Lisäksi nämä palaverit olivat osasyynä helpottamassa tehtäväriippuvuuksien hahmottamista erityisesti alhaalta ylöspäin tapahtuvassa kehityksessä (Pikkarainen ym., 2008).

Lyhyet iteraatiot: Lyhyillä iteraatioilla ketterässä kehityksessä pyritään jatkuvaan tiedonjakamiseen ja etenemisen tarkkaan seurantaan. XP:n määritelmän mukaan iteraatio on viikon mittainen, Scrumissa 1-4 viikkoa. Näin ollen XP prosessissa mitään ei toteuteta ilman sen testaamista ja yhdistämistä muihin osiin yhtä viikkoa pidempään ja Scrumissakin kohtalaisen lyhyissä sykleissä syklien ollessa yleisimmin 1-2 viikkoa. (Leffingwell, 2007; Schwaber, 1995)

Lyhyet iteraatiot nähtiin yksimielisesti sekä johdon että kehittäjien keskuudessa hyödyllisimmäksi ketteräksi käytännöksi, koska niiden vuoksi oli helppo muodostaa yhden ihmisen työmäärään sopivia tehtäviä joka iteraatioon. Lisäksi viestinnän osalta lyhyet iteraatiot nähtiin suurimpana syynä parantuneeseen viestintään tiimin sisällä. (Pikkarainen ym., 2008)

Tarinataulu (storyboard) XP:ssä ja tehtävälista (product backlog) Scrumissa: XP:n tarinataulu ja Scrumin tehtävälista ovat pääpiirteiltään sama asia eli lista toteutettavista asiakasvaatimuksista. Tarinataulun ja tehtävälistan avulla pyritään varmistamaan projektin läpinäkyvyys ja varmistamaan, että kaikki tiimin jäsenet ovat tietoisia projektin etenemisestä ja jäljellä olevista tehtävistä. Tarinataulussa ja tehtävälistassa on listattuna toteutettavat tehtävät prioriteettijärjestyksessä, jotka on määritelty asiakkaan kanssa iteraatioiden suunnittelupalavereissa. (Leffingwell, 2007; Pikkarainen ym., 2008; Schwaber, 1995)

Pikkaraisen ym. (2008) tutkimuksessa haastatellun kehittäjän mukaan XP:n tarinat tuovat asiakkaan selvästi lähemmäksi kehittäjiä ja auttavat näin ollen kehittäjiä ymmärtämään asiakasvaatimuksia paremmin. Samassa tutkimuksessa sekä kehittäjät että johto olivat yhtä mieltä siitä, että XP:n tarinat ja Scrumin tehtävälista paransivat projektin läpinäkyvyyttä huomattavasti, helpottivat projektin tilan seurantaa ja helpottivat jakamaan tehtäviä sopivan kokoisiksi kokonaisuuksiksi. Scrumin tehtävälista nähtiin myös arvokkaana työkaluna iteraatioiden auditointipalavereissa, missä projektin kokonaistilannetta arvioitiin (Pikkarainen ym., 2008).

Pariohjelmointi (pair programming): Pariohjelmointi on yksi XP:n mielenkiintoisimmista ja sille ominaisimmista periaatteista. Pariohjelmoinnissa kaksi kehittäjää työskentelee samalla tietokoneella. Tarkoituksena on varmistaa koodin laatu samanaikaisen vertaisarvioinnin avulla, koska siinä luodaan mahdollisuus keskustella koodista ja arvioida sitä. (Beck & Anders 2004)

Pariohjelmoinnin avulla kehittäjät saavat välitöntä vertaispalautetta jatkuvasti, jolloin koodin parantelu (refactoring) tapahtuu samanaikaisesti kuin itse

koodaus (Leffingwell, 2007). Pikkaraisen ym. (2008) tutkimuksessa kehittäjät näkivät pariohjelmoinnin parhaana tapana tarkastaa ja parannella koodia.

Yllä esitellyt käytännöt eivät olleet ainoita käytänteitä, jotka tehostivat tai paransivat tiimin viestintää, mutta yllä olevilla käytännöillä oli kaikista selvin vaikutus parantuneeseen viestintään ketterissä projekteissa useiden tutkimusten mukaan (Pikkarainen ym., 2008; Leffingwell, 2007; Melnik & Maurer, 2004; Koskela & Abrahamsson, 2004; Grisham & Perry, 2005; Schwaber, 1995; Beck & Anders 2004). Käsittelemättä jäi mm. jatkuva integraatio, joka nähtiin myös yhtenä viestintää tehostavana käytänteenä, mutta sen suoranainen vaikutus viestintään ei ollut niin selvä kuin tässä esitetyillä (Pikkarainen ym., 2008; Beck & Anders 2004). Myös Melnikin ja Maurerin (2004) esittämä ”kaikki voivat oppia kaikilta” –mentaliteetti jäi käsittelemättä, koska se on käytänteenä abstraktimpi ja näin ollen paljon vaikeampi myös ottaa käyttöön tai rakentaa yrityksen sisälle.

5 YHTEENVETO

Kasvokkaisviestintä on ketterän ohjelmistokehityksen ehkä eniten painotettu periaate. Sen tutkiminen on melko haastavaa, koska siitä on käytännössä mahdotonta saada esimerkiksi kvantitatiivista tutkimusaineistoa sen luonteen vuoksi. Siitä ei luonnollisesti myöskään jää minkäänlaista dokumentaatiota, koska se tapahtuu kasvotusten useimmiten informaalilla tavalla. Tämän tutkimuksen tarkoituksena oli pyrkiä selvittämään viestinnän ja erityisesti kasvokkaisviestinnän eri esiintymismuotoja ja sillä saavutettavia hyötyjä ketterässä ohjelmistokehityksessä sen kahdessa suosituimmassa menetelmässä Scrumissa ja XP:ssä.

Tämän tutkimuksen ongelma pyrki erottelemaan ketteriä viestintäkäytänteitä ja niillä saavutettavia etuja Scrumissa ja XP:ssä. Scrumn ja XP:n käytänteitä tutkimalla erityisesti viestinnällisestä näkökulmasta. Ensimmäinen huomio oli, että kaikki ketterät käytänteet ovat omalta osaltaan enemmän tai vähemmän ketteriä viestintäkäytänteitä. Iso osa näistä käytänteistä rajattiin kuitenkin tämän tutkimuksen ulkopuolelle, koska tutkimuksen paino oli nimenomaan tiimin sisäisessä viestinnässä monien käytäntöjen liittyessä usein projektin ja sen sidosryhmien väliseen viestintään. Tässä tutkielmassa esitetyt kahdeksan ketterää viestintäkäytännettä ovat pääasiallisesti tiimin sisäiseen viestintään pohjautuvia käytänteitä. Selvästi tärkeimmäksi käytännöksi muodostui kasvokkaisviestintä, joten sen korostaminen ketterissä menetelmissä ei ole ollenkaan liioiteltua. Vielä mielenkiintoisemman siitä tekee se, että lähestulkoon kaikki muut tässä tutkielmassa käsitellyt ketterät viestintäkäytänteet pohjautuvat myös nimenomaan kasvokkaisviestintään. Muita lähes yhtä tärkeäksi osoittautuneita käytänteitä olivat erityisesti avoin työskentelytila ja läsnä oleva asiakas. Asettamalla kehittäjät samaan tilaan pyritään

maksimoimaan kasvokkaisviestinnän määrä. Läsnä olevan asiakkaan vuoksi kehitys tehostui huomattavasti jokaisessa tässä tutkimuksessa käsitellyssä tapauksessa. Päivittäiset palaverit, iteraatioiden suunnittelupalaverit sekä lyhyet iteraatiot itsessään liittyvät myöskin hyvin läheisesti nimenomaan kasvokkaisviestintään ja sitä kautta tietämyksen jakamiseen.

Toisaalta näihin käytänteisiin liittyi myös joitain haasteita, joissa olisi riittävästi aihetta toiseen tämän laajuiseen tutkimukseen. Suurimmiksi haasteiksi osoittautuivat läsnä olevaan asiakkaaseen liittyvät suuret vaatimukset kyseisen projektin osaamiselle sekä projektille omistautumiseen lähes täysipäiväisesti. Muihinkin käytänteisiin liittyi tiettyjä haasteita kuten keskittymisvaikeudet avoimessa tilassa työskentelyyn liittyen ja koko projektiin keskittyminen pitemmällä tähtäimellä Scrumin tehtävälistaan liittyen. Pääsääntöisesti kuitenkin kaikki tässä tutkielmassa tutkitut käytänteet aiheuttivat moninkertaisen määrän positiivisia vaikutuksia ohjelmistokehitykseen suhteessa niiden luomiin haasteisiin.

Yksiselitteisesti parhaan tai parhaiden viestintäkäytäntöjen löytäminen on hyvin pitkälti mahdotonta jo viestinnän luonteesta johtuen. Tämän tutkimuksen tarkoituksena oli eritellä tunnetuista käytännöistä kirjallisuuden perusteella nimenomaan tiimin sisäistä viestintää tehostavia käytäntöjä ja pohtia syitä näiden tehostavaan vaikutukseen. Ehdottomasti tärkeimmäksi tulokseksi tässä tutkimuksessa nousi kasvokkaisviestinnän merkitys ketterälle ohjelmistokehitykselle sen ollessa kriittinen osa jokaista muuta tässä tutkimuksessa eriteltyä käytäntöä. Näin ollen kasvokkaisviestinnän voisi sanoa olevan tai sen ainakin tulisi olla ketterän kehityksen yksi perusarvoista, joihin kaikki ketterät käytänteet pohjautuvat. Tässä tutkielmassa eritellyistä käytännöistä syntyi käyttökelpoista tietoa tilanteisiin, joissa ohjelmistoprojektin viestintä vaikuttaa olevan puutteellista. Esitellyistä käytännöistä voi pyrkiä

tunnistamaan ensinnäkin kyseiseen projektiin sopivia käytäntöjä, sekä miettiä jo olemassa olevien käytäntöjen viestinnän tehokkuuden parantamista.

Viestintä ketterissä menetelmissä osoittautui lopulta hyvin laajaksi alueeksi, joten työtä on rajattu monella tavalla. Tästä työstä rajattiin pois tiimin ulkopuolinen viestintä, kuten esimerkiksi viestintä tiimin ja johdon välillä sekä viestintä tiimin ja projektin sidosryhmien välillä. Näissä molemmissa olisi hyvin paljon tutkittavaa jatkotutkimusta ajatellen. Toinen mielenkiintoinen tutkimusta vaille jäävä teema ovat ketterän kehityksen eri viestintämuodot, kuten esimerkiksi sähköposti, pikaviestimet, blogit ym. Kolmas jatkotutkimusaihe sivuaa hyvin paljon myös edellä mainittua eli viestintä hajautetussa ketterässä kehityksessä. Viestinnän osalta hajautettu ketterä kehitys olisi erityisen mielenkiintoinen ja tärkeä tutkimuksen kohde, koska viestinnän toimivuuden varmistamisen tarve kasvaa moninkertaiseksi kasvokkaisviestinnän määrän pudotessa vain murto-osaan suhteessa samassa tilassa toimivaan ketterään kehitystiimiin ja koska ohjelmistokehitys on hyvin suurelta osin hajautettua jo tälläkin hetkellä.

LÄHDELUETTELO

Abrahamsson, P., Salo, O., Ronkainen J., & Warsta J. (2002). Agile software development methods: Review and analysis. Espoo, Suomi: Otamedia Oy.

Ambler S. 2009, Communication on Agile Software Projects. [viitattu 23.11.2009] Saatavilla www-osoitteessa <URL:http://www.agilemodeling.com/essays/communication.htm>

Anderson, D. J. (2003). Agile management for software engineering, applying the theory and constraints for business results. Upper Saddle River, New Jersey, United States: Prentice Hall. Saatavilla osoitteessa <URL:http://www.google.com/books?id=hawMF31KCRsC&lpg=PR21&ots=ZjadMxaHSU&dq=Agile%20management%20for%20software%20engineering%2C%20applying%20the%20theory%20and%20constraints%20for%20business%20results.&lr&hl=fi&pg=PP1#v=onepage&q&f=false> [haettu 4.5.2010]

Aula, P. (1999). Organisaation kaaos vai kaaoksen organisaatio? Dynaamisen organisaatioviestinnän teoria. Helsinki, Suomi: Helsinki Loki-Kirjat.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas,

D. (2008). Manifesto for agile software development. Saatavilla www-osoitteessa <URL:http://agilemanifesto.org/> [haettu 15.11.2009]

Beck, K., & Andres, C. (2004). Extreme programming explained: Embrace change (2nd ed.). Upper Saddle River, New Jersey, USA: Addison-Wesley Professional.

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

Coad, P., Luca, J. d., & Lefebvre, E. (1999). Java modeling color with uml: Enterprise components and process with cdrom. Upper Saddle River, New Jersey, USA: Prentice Hall PTR.

Cockburn, A. (2004). Crystal clear a human-powered methodology for small teams. Addison-Wesley Professional.

Deemer, P., & Benefield, G. (2007). SCRUM primer: An introduction to agile project management with Scrum. Saatavilla www-osoitteessa <URL: http://www.rallydev.com/documents/scrumprimer.pdf>

Espinosa, J. A., & Carmel, E. (2003). The impact of time separation on coordination in global software teams: A conceptual foundation. Software Process: Improvement and Practice 8(4), 249-266.

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

Grisham, P. S., & Perry, D. E. (2005). Customer relationships and Extreme Programming. Teoksessa HSSE '05: Proceedings of the 2005 Workshop on Human and Social Factors of Software Engineering, St. Louis, Missouri. 1-6.

Henttonen, K., & Blomqvist, K. (2005). Managing distance in a global virtual team: The evolution of trust through technology-mediated relational communication. Strategic Change 14(2), 107-119.

Juholin, E. (2009). Communicare! : Viestintä strategiasta käytäntöön. Porvoo, Suomi: WS Bookwell.

Karlström, D., & Runeson, P. (2006). Integrating agile software development into stage-gate managed product development. Empirical Software Engineering 11(2), 203-225.

Koskela, J., & Abrahamsson, P. (2004). On-site customer in an XP project: Empirical results from a case study. Teoksessa: Proceedings 11th European Conference on Software Process Improvements, 10-12.

Leffingwell, D. (2007). Scaling software agility: Best practices for large enterprises. Upper Saddle River, New Jersey, USA: Addison-Wesley Professional.

Lycett, M., Macredie, R. D., Patel, C., & Paul, R. J. (2003). Migrating agile methods to standardized development practice. Computer, 36, 79-85.

Marcal, A. S. C., Soares, F. S. F., & Belchior, A. D. (2007). Mapping CMMI project management process areas to SCRUM practices. Proceedings of the 31st IEEE Software Engineering Workshop, 13-22.

Melnik, G., & Maurer, F. (2004). Direct verbal communication as a catalyst of agile knowledge sharing. Agile Development Conference/Australasian Database Conference, 21-31.

Misra, S. C., Kumar, V., & Kumar, U. (2009). Identifying some important success factors in adopting agile software development practices. Journal of Systems and Software, 82(11).

Paasivaara, M., & Lassenius, C. (2003). Collaboration practices in global inter-organizational software development projects. Software Process: Improvement and Practice, 8(4), 183-199.

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

PMBOK (2004). A guide to the project management body of knowledge (PMBOK guides) Project Management Institute.

Poppendieck, M., & Poppendieck, T. (2003). Lean software development: An agile toolkit. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Schwaber, K. (1995). SCRUM development process. Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA), 117-134.

Schwaber, K. (2004). Agile project management with scrum (microsoft professional) (1st ed.) Microsoft Press.

Shannon, C. E., & Weaver, W. (1949). The mathematical theory of communication. Urbana, Illinois, United States: University of Illinois Press.

Tai, G. (2005). A communication architecture from rapid prototyping. HSSE '05: Proceedings of the 2005 Workshop on Human and Social Factors of Software Engineering, St. Louis, Missouri. 1-3.

Turner, R. (2003). People factors in software management: Lessons from comparing agile and plan-driven methods

Wiio, O. A. (1978). Contingencies of organizational communication : Studies in organization and organizational communication. Helsinki, Finland: Osmo A. Wiio Helsinki School of Economics.

Wiio, O. A. (1992). Viestinnän tutkimussuuntia. Helsinki, Suomi: Yliopistopaino.

Åberg, L. (2000). Viestinnän johtaminen. Helsinki, Suomi: Inforviestintä.