Toni Räsänen

Asiakkaan rooli ketterän lähestymistavan mukaisessa kehittämistyössä

Tietojärjestelmätieteen
kandidaatintutkielma
28.5.2010

Jyväskylän yliopisto
Tietojenkäsittelytieteiden laitos
Jyväskylä

TIIVISTELMÄ

Toni Räsänen
Asiakkaan rooli ketterän lähestymistavan mukaisessa kehittämistyössä/ Toni Räsänen
Helsinki: Jyväskylän yliopisto, 2010.
41 s.
Kandidaatintutkielma

Ketterien menetelmien mukaisen järjestelmäkehityksen eräs kulmakivi on asiakasläheinen toiminta. Ketterän lähestymistavan ydinajatuksena on ottaa asiakas mukaan ohjelman kehitykseen, jolloin mahdollistetaan asiakkaalle parempi näkymä ohjelmiston kehitykseen ja projektin etenemiseen. Osallistuminen aktiivisesti projektityöhön tarjoaa asiakkaalle uusia vaikuttamismahdollisuuksia verrattuna suunnitelmapohjaisiin kehitysmenetelmiin. Ketterät menetelmät siis elävät vahvasti asiakkaasta. Tästä seuraa, että roolin haltijan on hyvä tiedostaa tehtävänsä tärkeys saadakseen täyden hyödyn ketteryydestä.

Tässä työssä tarkastellaan asiakkaan roolia ketterässä järjestelmäkehityksessä. Työn tavoitteena on hahmottaa asiakkuutta, sitä mitä asiakkaalla tarkoitetaan sekä minkälaisia mahdollisuuksia ketterät menetelmät antavat asiakkaalle kehitystyön hallintaan. Toisaalta tarkastelleen myös sitä mitä mahdollisuuksia kehitysprojekti saa ketterien menetelmien mukaisesta asiakkaasta – asiakkaasta, joka on aktiivinen ja valmis ohjaamaan kehitystä vaatimusten kautta. Työssä selvitetään myös tarkemmin, kuinka asiakkaan rooli on määritelty Scrum- ja XP-menetelmissä. Työn lopuksi kerrotaan aiempiin tutkimuksiin nojaten, millaisia kokemuksia asiakkaan roolista ketterässä kehittämisessä on saatu.

Tutkielman perusteella voidaan sanoa, että asiakkaan ottaminen läheisesti mukaan ohjelmiston kehitystyöhön auttaa useimmiten kehitystyötä. Läheinen yhteistyö on kuitenkin altis inhimillisille häiriöille, ihmisten on tultava toimeen keskenään ja haluttava tehdä sitä mitä tekevät. Ketterät menetelmät siirtävät asiakasta ja kehittäjiä samalle puolelle projektissa suosimalla avoimuutta ja rehellisyyttä. Ketterissä menetelmissä osallistuminen voi vaatia paljon, mutta aina on punnittava saadun hyödyn suhde kustannuksiin ja tehtävä sen mukaan johtopäätökset. Asiakas on lopulta oikeassa.

AVAINSANAT: järjestelmäkehitys, asiakas, ketterä menetelmä, XP, Scrum

Ohjaaja: Mauri Leppänen
Tietojenkäsittelytieteiden laitos
Jyväskylän Yliopisto

Tarkastaja: Mauri Leppänen
Tietojenkäsittelytieteiden laitos
Jyväskylän Yliopisto

SISÄLLYSLUETTELO

1. JOHDANTO

Ketterissä ohjelmistonkehitysmenetelmissä painotetaan muutoksiin valmistautumista ja jatkuvaa kommunikaatiota projektin osapuolten välillä (Highsmith 2002). Ketterissä menetelmissä on oleellista, että ohjelmiston kehitysprosessi on avoin, jolloin se on näkyvä kaikille projektin osapuolille. Ohjelmistoa kehitettäessä pyritään saattamaan välituloksia osapuolten tarkasteltavaksi tasaisin väliajoin. Tällöin yhteinen päämäärä eli tehdä mahdollisimman hyvä ohjelmisto etenkin asiakkaan, mutta myös tekijöiden näkökulmasta on mahdollista säilyttää kirkkaana mielessä sekä ohjata kehitystyötä mukautuen sillä hetkellä käytettävissä oleviin resursseihin ja tietämykseen.

Asiakkaan rooli on oleellinen ketterissä menetelmissä. Asiakkaan oletetaan osallistuvan aktiivisesti ohjelmistokehitykseen olemalla läsnä ja tavoitettavissa ja halukas antamaan lisätietoja ja tukea ohjelmiston kehittäjille (Turner ym. 2003). Ketterät menetelmät tuovat asiakkaalle lisää velvoitteita ja vastuuta. Aiemmin vaatimukset saatettiin lyödä lukkoon ohjelmistoprojektin alussa, minkä jälkeen aloitettiin suunnittelu- ja toteutustyö. Tällä tavoin edetessä suunnitelmiin on kuitenkin vaikea tehdä mahdollisia muutoksia. Onnistuneen ohjelmistoprojektin tuotos on yleensä asiakkaan toiveiden mukainen, mutta se voi olla täysin erilainen kuin ensimmäisissä suunnitelmissa oli kaavailtu (Highsmith 2002). Ketterissä menetelmissä asiakkaan ja ohjelmiston kehittäjien tavoitteena on kehittää ohjelmistoa vähä vähältä toimivina kokonaisuuksina ja hallita vaatimuksia sekä projektin kohtaamia muutoksia säännöllisillä tapaamisilla (Schwaber ym. 2001). Jos asiakas ei mukaudu aktiiviseen rooliin, kehittäjät saattavat jäädä pimentoon. jolloin vaarana on kehitystyön hidastuminen sekä virheiden määrän lisääntyminen, koska joudutaan työskentelemään enemmän oletusten ja tulkintojen varassa. (Korkala ym. 2006).

Tässä tutkielmassa on tarkoitus tuottaa kirjallisuuden avulla yleiskuva asiakkaan roolista ketteriä menetelmiä noudattavissa ohjelmistoprojekteissa. Tarkemmin selvitetään Scrum- ja XP-menetelmien asiakkaan mukaan ottamiseen tarjoamia välineitä. Ketterä lähestymistapa painottaa mahdollisimman läheisiä suhteita asiakkaaseen. Asiakas ei kuitenkaan aina ole yksikäsitteinen, helposti rajattava taho.

Tutkielman tutkimusongelma voidaan tiivistää seuraavaan muotoon: "Mikä on asiakkaan rooli ketterissä ohjelmistonkehitysprojekteissa, mitä vastuita ja etuja ketterien menetelmien kuvaama rooli asiakkaalle tuo?". Tutkimusongelma jaetaan työssä kolmeen tutkimuskysymyksen: 1) Mitä asiakkaalla itse asiassa tarkoitetaan? 2) Mikä vaikutus ketterillä menetelmillä on asiakkaaseen? Toisin sanoen mitä vastuita ja odotuksia asiakkaaseen kohdistuu, mutta toisaalta myös miten asiakas hyötyy uudenlaisesta asetelmasta? 3) Millaisia havaintoja asiakkaan toimimisesta ketteriä menetelmiä noudattavissa ohjelmistoprojekteissa on saatu empiirisillä tutkimuksilla?

Tutkielman ensimmäisessä osassa (Luku 2) tutustutaan ketterän menetelmän perusajatuksiin, siihen minkälainen roolijako on yhteinen ketterille menetelmille, erityisesti minkälainen rooli asiakkaalla on ketterän menetelmän mukaisessa kehitystyössä. Tästä siirrytään syvemmälle ja luodaan katsaus kahteen ketterään menetelmään, jotka ovat taajaan esillä kirjallisuudessa (Luku 3). Tässä osiossa kerrotaan lähemmin roolijaosta XP- ja Scrum-menetelmissä, keskittyen asiakkaan rooliin kyseisissä menetelmissä. Välttämättä ei pyritä rankkaan vastakkainasetteluun, vaan pikemminkin vertaamaan näitä kahta menetelmää sekä pohtimaan mahdollisia yhtäläisyyksiä ja jopa yhteensopivuuksia. Työn viimeisessä osassa (Luku 4) ennen loppupäätelmiä käydään läpi joitain empiirisiä tutkimuksia, joista etsitään yhtymäkohtia teoriaan, jonkinlaista konkreettista näkemystä siihen, millaiseksi asiakkaan rooli koetaan ketterän kehittämistavan mukaisessa ohjelmistonkehitystyössä. Työ päättyy luvussa 5 annettavaan yhteenvetoon.

2. ASIAKAS KETTERÄN LÄHESTYMISTAVAN NÄKÖKULMASTA

Tässä luvussa kerrotaan lyhyesti ketteristä menetelmistä, niiden yleispiirteistä ja tarkoituksesta. Toisessa luvussa tarkastellaan yleistä roolijakoa ketterän lähestymistavan mukaisissa ohjelmistonkehitysmenetelmissä. Lisäksi hahmotetaan erilaisia mahdollisia ohjelmistoprojektitilanteita ja pohditaan asiakkuutta näissä tapauksissa.

2.1 Agile manifestin näkemys

Ketterät menetelmät (agile methods) kehittyivät kevyemmiksi vaihtoehdoiksi niitä edeltäville vesiputouksen (waterfall) omaisille menetelmille, joskus myös puhutaan suunnitelmapohjaisista menetelmistä (Plan driven methods) (Highsmith 2002). Erona aikaisempiin menetelmiin on halu tunnustaa, että projektilla on vastassa lukuisia muuttuvia tekijöitä, joihin on pyrittävä mukautumaan. Aikaisemmissa menetelmissä vaatimukset lyötiin lukkoon ennen suunnittelun ja toteutuksen alkamista. (McCauley 2001). Ketterissä menetelmissä lähtökohtana on, että vaatimukset voivat muuttua projektin aikana. Tätä ajatusmallia puoltaa myös se havainto, etteivät asiakkaat välttämättä osaa määritellä vaatimuksia kovinkaan tarkasti projektin alkuvaiheessa vaan ohjelmistonkehittäjien on muodostettava ymmärrys asiakkaan tarpeista projektin edetessä. (Davis 1994). Abrahamsson ym. (2002) listaavat ketteriä menetelmiä kokoavassa artikkelissaan neljä yhdistävää tekijää, joista ne voi tunnistaa. Kehitystyö on kokoavaa, ja se tapahtuu lyhyissä sykleissä. Kehitystyössä asiakas ja kehittäjät ovat läheisessä yhteistyössä. Menetelmä on suoraviivainen, helppo oppia ja muokata sekä hyvin dokumentoitu. Kehitystyö on myös mukautuvaa, muutosten tekeminen onnistuu myöhäisessä vaiheessa. Sillitti ym. (2005) toteavat, että ketterät menetelmät keskittyvät tuottamaan arvoa asiakkaalle ja varmistamaan, että asiakas ymmärtää arvon ja on tyytyväinen projektin lopputuotteeseen eli kehitettyyn ohjelmistoon.

Ketterät menetelmät pitävät ihmisiä projektien tärkeimpinä menestystekijöinä, ei niinkään prosessia (Cockburn ym. 2001). Ketterien menetelmien rakentuessa paljolti ihmisten ja heidän keskinäisen vuorovaikutuksensa ympärille ihmisten aktiivinen

osallistuminen nousee tärkeään rooliin. Cockburn (2002) neuvoo yksilöä ajattelemaan, mitä kulloinkin tekee ja kertomaan sitten muille, mitä tekee. Kevyen prosessin myötä myös oikeanlaisten ihmisten löytäminen on tärkeää. Sanotaan, että vankka prosessi voi korvata ihmisten heikkouksia. Ketterät menetelmät uskovat ihmisten voimaan: lopputuotteen kannalta on tärkeää saada osaavat, oikeanlaiset taidot omaavat ihmiset yhteen sen sijaan, että nojauduttaisiin vahvasti prosessiin (Highsmith 2004).

2000-luvun alussa joukko ketterien menetelmien johtohahmoja kerääntyi yhteen keskustelemaan, mitä yhteistä eri menetelmillä mahdollisesti on. Kokoontumisen tuloksena syntyi Agile manifesti (Agile Alliance 2001), johon koottiin ketterien menetelmien arvot ja periaatteet. Manifestin neljä arvoa ja kaksitoista periaatetta kuvaavat yleisesti ketterien menetelmien takana olevia ajatuksia. Seuraavassa käydään lyhyesti läpi asiakkaan kannalta keskeisimpiä periaatteita. Ensimmäisenä periaatteena on pitää asiakas tyytyväisenä toimittamalla toimivia versioita kehitettävästä ohjelmistosta säännöllisin väliajoin. Näin asiakas näkee selkeämmin ohjelmiston kehittymisen ja pääsee vaikuttamaan kehityksen kulkuun, mikäli ohjelmisto ei vaikuta halutunlaiselta. Toisena periaatteena on pyrkiä olemaan valmiina kohtaamaan muuttuvat vaatimukset. Hyväksytään, että vaatimukset voivat muuttua ja ollaan valmiita toimimaan niiden mukaan. Tarkoituksena on, että asiakas hyötyy projektin kyvystä toimia muutosten vaatimalla tavalla ja tuottaa asiakkaalle sitä kautta lisäarvoa. Kolmanneksi lupaudutaan toimittamaan ohjelmistosta toimivia versioita tasaisin väliajoin. Tämä on yhteydessä ensimmäiseen periaatteeseen ja lisää samalla tavalla kehitettävän ohjelmiston näkyvyyttä asiakkaan suuntaan. Asiakkaalla on mahdollisuus vaikuttaa kehitystyöhön varhaisessa vaiheessa. Lisäksi periaatteissa painotetaan eri tahojen säännöllistä kanssakäymistä koko projektin ajan. Säännöllisen kanssakäymisen kautta kaikilla osapuolilla on mahdollisimman yhteneväinen kuva projektista ja kehitettävästä ohjelmasta. Parhaimmillaan kommunikaatio vähentää väärinkäsitysten ja turhien olettamusten määrää. Lisäksi painotetaan kasvotusten tapahtuvaa yhteydenpitoa kehitysprojektin sisällä sekä kehitysprojektin ja muiden tahojen (esim. asiakkaan) välillä. (Agile Alliance 2001) .

Ketterien menetelmien laaja joukko voi hämmentää ja vaikeuttaa menetelmän

käyttöönottoa ja valintaa. Tiettyä menetelmää sovellettaessa sitä ei kuitenkaan tarvitse noudattaa sanatarkasti, vaan menetelmää voidaan mukauttaa kulloisenkin kehitysprojektin mukaan.(Sillitti ym. 2005).

2.2 Roolit ketterissä menetelmissä

Ketterissä menetelmissä voidaan erottaa ainakin seuraavat roolit: kehittäjät, asiakkaat ja johto (Highsmith 2002). Kehittäjät toteuttavat vaatimukset ja luovat ohjelmiston. Paremmin voisi puhua tiimistä, johon kuuluu ohjelmoijia, teknisiä kirjoittajia, käyttöliittymäsuunnittelijoita – kaikki kehittäjiä jotka kehittävät tuotetta (Schwaber 2004). Asiakkaat ovat kehitystä ohjaava taho, ja ketterissä menetelmissä läheinen suhde asiakkaiden ja kehittäjien välillä on oleellinen (Turner ym. 2003). Johto koostuu henkilöistä, jotka pitävät yllä virallisia suhteita esimerkiksi asiakkaisiin ja taustaorganisaatioon.

Ketterissä menetelmissä uskotaan yksilöiden voimaan. Kyvykkäät yksilöt sekä näiden keskinäinen kanssakäyminen on tärkeä osa ketterää menetelmää (Cockburn ym. 2001). Toinen roolien välistä suhdetta kuvaava havainto on, että asiakkaiden, käyttäjien ja johdon tuen puuttuminen vahingoittaa projektia riippumatta siitä kuinka kyvykkäitä henkilöitä projektissa on mukana (Cockburn ym. 2001). Standish Group on tutkinut projektien menestystekijöitä, ja sen tutkimusraportissa (Standish Group 2002) tärkeimmäksi menestystekijäksi on valittu käyttäjien osallistuminen. Asiakas tai käyttäjä, ne joille tuotetta tehdään, on otettava mukaan projektiin. Ketterien menetelmien vaalima samassa tai eri roolissa olevien henkilöiden välinen kanssakäyminen myös korostaa vuorovaikutustaitojen merkitystä (Sillitti ym. 2005). Halu ja kyky kommunikoida ovat oleellinen osa ketterän projektin onnistumista. Voidaan sanoa, että ketterien menetelmien mukaisissa projekteissa osallistujien tulee olla täysillä mukana, koska käytetty prosessi olettaa osallistujien olevan aktiivisia. Prosessin varaan ei voi samalla tavalla tuudittautua kuin käytettäessä vesiputousmallin mukaista menetelmää, jossa edetään tarkkojen ennakkosuunnitelmien mukaisesti.

Boehm (2002) on listannut joitain kuvaavia eroja kehittäjä- ja asiakasrooleissa sekä vaatimuksissa ketterien menetelmien ja vahvasti ennakkosuunniteluun perustuvien menetelmien välillä.

Taulukko 1. Ketterän ja suunnitelmapohjaisen menetelmän perustat. (Boehm 2002 s.68).

	Ketterä menetelmä	**Suunnitelmapohjainen menetelmä**
Kehittäjät	Ketteriä, perillä asioista, yhteistyössä, lähekkäin	Suunnitelmiin sitoutuneita, asianmukainen ja riittävä taitotaso, mahdollisuus tukeutua ulkopuoliseen tietolähteeseen
Asiakkaat	Omistautuneita, perillä asioista, yhteistyössä, lähekkäin, toimivat edustajina, omaavat päätösvaltaa	Mahdollisuus tukeutua ulkopuoliseen tietolähteeseen, yhteistyössä, toimivat edustajina
Vaatimukset	Lisääntyvät, muuttuvat ja tarkentuvat projektin edetessä. Mahdollisesti nopeita muutoksia.	Ennalta tiedossa. Säilyvät suurelta osin muuttumattomina projektin ajan.

Ketterissä menetetelmissä asiakkaalla on erityisen tärkeä rooli. Asiakkaan tulisi tuntea alue jolle ohjelmistoa kehitetään sekä kyky tehdä tärkeitä päätöksiä. (Boehm 2002, Sillitti ym. 2005). Asiakkaan tehtäviin kuuluu määrittää ja valita toteutettavat vaatimukset. Asiakkaat myös lopuksi hyväksyvät kehitetyn ohjelmiston sekä testaavat sen. Jokaisen kehitysjakson päättyessä julkaistaan toimiva ohjelmisto, jolloin asiakas pääsee testaamaan ohjelmistoa ja antamaan siitä palautetta. Asiakas hyötyy tästä, koska ohjelmisto paljastaa mahdolliset puutteet tarjoten asiakkaalle mahdollisuuden puuttua niihin ja pyytää muutosta jo varhaisessa vaiheessa. Asiakkaalle syntyy parhaimmillaan myös tunne, että hän hallitsee kehitystä ja on siinä mukana ja voi vaikuttaa kehitettävään ohjelmistoon jo matkan varrella sen sijaan, että ohjelmisto olisi piilossa julkaisuun saakka. Asiakkaan oletetaan osallistuvan kehitykseen aktiivisesti ja olevan tavoitettavissa ellei jopa läsnä projektitiimin luona. Asiakkaalta voi tällöin pyytää palautetta ja tarkennuksia vaatimuksiin. Suora läheinen yhteys asiakkaan ja projektitiimin välillä auttaa vähentämään dokumentaatiota sekä epäselvyyksiä vaatimusten taholla (Sillitti ym. 2005).

Asiakkaan osallistuminen on merkittävässä osassa ketterissä menetelmissä. Asiakkaan osallistumista tarvitaan, koska vaatimuksia ei ole määritetty lopullisesti projektin alussa ja ne voivat muuttua. Ketterät menetelmät vaativat asiakkaalta huomiota koko projektin ajan. Vastaavasti asiakas hyötyy tästä panoksesta kehitystyön paremmalla ohjattavuudella sekä sillä, että asiakkaan ei tarvitse tietää kaikkia ohjelmistolle asettamiaan vaatimuksia heti projektin alussa. Chown ym. (2007) mukaan yksi ketterän lähestymistavan mukaan toimivan ohjelmistoprojektin onnistumisen kannalta tärkeistä tekijöistä on asiakkaan vahva sitoutuminen projektiin. Artikkelissa kuvataan ketterien menetelmien mukaisten ohjelmistoprojektin kannalta kriittisiä menestystekijöitä. Kyselytutkimuksen tulosten perusteella päädyttiin siihen, että asiakkaan osallistuminen on yksi tällaisista tekijöistä. Menestyksen kannalta edullisiksi tekijöiksi osoittautuivat hyvät asiakassuhteet, asiakkaan aktiivinen osallistuminen ja läsnäolo sekä toimivaltainen asiakas, asiakas jolla on mahdollisuus tehdä päätöksiä suhteellisen nopeasti ja itsenäisesti (Chown ym. 2007). Seuraavassa kohdassa hahmotellaan tarkemmin erilaisia projektitilanteita ja kuinka asiakas niissä esiintyy.

2.3 Asiakas erilaisissa projektitilanteissa

Eri ohjelmistokehitysprojekteissa rooliasetelmat voivat vaihdella. Tässä määritellään muutamia mahdollisia tapauksia sen mukaan, kenelle ohjelmistotuotetta ollaan tekemässä. Näiden mukaan hahmotellaan vastaavaa asiakaskäsitettä. Alla on listattu neljä mahdollista asetelmaa, joissa ohjelmistoa kehitetään hieman erilaisille asiakkaille.

- Yrityksen sisäinen ohjelmistoprojekti
- Ohjelmistotalo kehittää ohjelmistoa tietylle asiakkaalle
- Ohjelmistotalo kehittää ohjelmistoa kuluttajamarkkinoille
- Ohjelmistotalo kehittää ohjelmistoa alihankintana toiselle ohjelmistotalolle.

Riippuen käytettävästä ohjelmistonkehitysprosessista asiakkaalle tarjoutuu erilaisia keinoja päästä vaikuttamaan kehitettävään tuotteeseen. Samoin riippuen käytetystä kehitysprosessista kehitysprojektille tulee erilaisia prosessin määrittämiä mahdollisuuksia olla tekemisissä asiakkaan kanssa. Asiakkaan roolia voidaan hahmottaa

osallistumisen kautta. Kaulio (1998) on jakanut asiakkaan osallistumisen kolmeen tasoon, jotka kuvaavat asiakkaan osallistumisen syvyyttä. Ensimmäinen taso tarkoittaa tilannetta jossa tuote tehdään asiakkaan puolesta asiakkaalle. Tällainen tilanne on esimerkiksi ohjelmistotalon kehittäessä ohjelmistoa kuluttajamarkkinoille. Ohjelmistolle asetetut vaatimukset voidaan kerätä monesta eri lähteestä. Vaatimusten tukena voi olla markkinatutkimuksia, galluppeja, käyttäjätutkimuksia ja yleisiä teorioita, joilla saadaan kerättyä tietoa asiakkaasta (Kabbedijk ym. 2009). Toinen aste tarkoittaa tilannetta, jossa asiakas on jo enemmän mukana tuotteen kehityksessä. Tämä tilanne voisi olla esimerkiksi kehitettäessä ohjelmistoa yrityksen sisäisenä projektina tai kehitettäessä ohjelmistoa tietylle asiakkaalle. Tieto asiakkaasta kerätään kuten edellä, mutta asiakkaalla on mahdollisuus valita eri vaihtoehdoista, tuotetta voidaan räätälöidä asiakkaan tarpeisiin (Kabbedijk ym. 2009). Kolmas taso tarkoittaa asiakkaan vahvaa osallistumista, tällöin asiakas on mukana suunnittelussa ja antaa kehitysehdotuksia ja vaatimuksia. Tällainen asetelma voisi olla esimerkiksi ohjelmistotalon kehittäessä tuotetta alihankintana toiselle ohjelmistotalolle, tai tilanteessa jossa ohjelmistoa kehitetään tietylle yhdelle asiakkaalle.

Se, kuka on asiakkaan roolissa vaihtelee kussakin edellä mainituista tilanteista. Yrityksen sisäisessä projektissa asiakastaho voi olla esimerkiksi yrityksen toinen osasto. Tilaavalta osastolta voidaan määrätä joku henkilö toimimaan yhteistyössä ohjelmistoa kehittävän osaston kanssa asiakkaan roolissa. Fraser ym. (2004) käyttää termiä "proxy customer" tarkoittaen henkilöä, joka toimii rajapintana asiakkaan ja kehittäjien välillä. Tämä nimetty asiakas voi hallinnoida vaatimuksia sekä kerätä tilaavan osaston henkilöiltä, siis mahdollisilta loppukäyttäjiltä, kehitettävään ohjelmistoon kohdistuvia toiveita ja vaatimuksia. Ohjelmistotalon kehittäessä tuotetta tietylle yritysasiakkaalle asiakkaan tarpeet ja vaatimukset ovat kehitystä eteenpäin vievä voima. Tässä tilanteessa asiakas määrää vaatimukset. Kuluttajamarkkinoille kehitettävällä ohjelmistolla ei välttämättä ole ainakaan aluksi asiakasta, joka määrittäisi vaatimuksia. Tällaisessa tilanteessa kehittäjäorganisaation sisällä projektilla voi olla asiakkaan rooliin nimetty henkilö tai henkilöitä, jotka toimivat kehitysprojektin suuntaan asiakkaan roolissa (Fraser ym. 2004). Neljäntenä mahdollisena tilanteena oli tapaus, jossa ohjelmistotalo kehittää ohjelmistoa alihankintana jollekin toiselle ohjelmistotalolle. Tässä tilanteessa

alihankinnan ostanut ohjelmistotalo toimii asiakkaana, ja tilaavan yrityksen puolelta voi olla nimetty joku henkilö asiakkaan rooliin ohjaamaan kehitystyötä. Tämä henkilö toimii alihankintayrityksen suuntaan asiakkaana edustaen tilaajaa (Paasivaara ym. 2009).

Riippumatta tilanteesta asiakkaan tulisi näkyä selkeänä roolina ohjelmistoprojektille. Asiakkaan tehtävänä on asettaa vaatimuksia kehitettävälle ohjelmistolle. Aiemmin asiakas saattoi näkyä vain vaatimuslistan muodossa, mutta ketterissä menetelmissä oletetaan asiakkaalta aktiivisempaa osallistumista.

3. ASIAKKAAN ROOLI XP- JA SCRUM-MENETELMISSÄ

Tässä luvussa tarkastellaan tarkemmin asiakkaan roolia kahdessa ketterässä menetelmässä, XP:ssä ja Scrumissa. Ensin kerrotaan yleisesti XP- ja Scrum-menetelmistä sekä roolijaosta kyseisissä menetelmissä. Sen jälkeen perehdytään tarkemmin asiakkaan rooliin ja määritelmään. Selvitetään, mitä käytäntöjä kyseiset menetelmät tarjoavat asiakkaan osallistumiseen. Lopuksi pyritään tarkastelemaan eroja asiakkaan roolien välillä näissä kahdessa menetelmässä.

3.1 eXtreme Programming

Extreme Programming (XP) -menetelmä on kokoelma ohjelmistonkehityskäytäntöjä (Beck 2000). Näiden käytäntöjen tavoitteena on tuottaa laadukasta lähdekoodia ja sitä kautta loisteliaita ohjelmia. XP-menetelmästä voi nostaa esiin neljä keskeistä arvoa: kanssakäyminen (ts. kommunikaatio), palaute, yksinkertaisuus ja rohkeus (Beck 2000). Kommunikaatiota korostetaan, koska usein juuri puutteellinen kommunikaatio ajaa projektin vaikeuksiin (Korkala ym. 2006). Palaute nostaa palautteen saamisen ja antamisen esiin. Kehittäjien on oltava valmiita palautteeseen; palautetta tulee asiakkaalta, järjestelmältä sekä toisilta kehittäjiltä, ja sitä on osattava arvostaa ja ymmärtää. Yksinkertaisuudella tarkoitetaan sitä, että on keskityttävä olennaiseen: tulee toteuttaa mahdollisimman yksinkertainen ohjelma, joka täyttää asiakkaan vaatimukset. Rohkeus on rohkeutta tehdä vaikeiltakin tuntuvia päätöksiä, jotka tukevat muita arvoja ja käytäntöjä. XP määrittelee neljän arvon lisäksi säännöstön, jossa on kaksitoista kehitystyötä ohjaavaa käytäntöä. Näitä ovat esimerkiksi pariohjelmointi ja testausvetoinen kehitys. XP-prosessi ei sisällä mitään hallinnollisia käytänteitä toisin kuin Scrum-menetelmä (Kane ym. 2002). XP perustuu viikoittaisiin kehitysjaksoihin, joiden lopussa ohjelmisto kootaan ja testataan. Viikon alussa valitaan toiminnallisuudet (user stories), jotka aiotaan seuraavan jakson aikana kehittää.

XP-menetelmä määrittelee roolijaon. Roolitus on jonkinlainen tavoitetila. Roolitusta ei pidä pyrkiä määrittämään väkisin, vaan tärkeää on mukautua tilanteeseen. XP:ssä voidaan erottaa seuraavat roolit (Beck ym. 2004):

- *Testaajat* vastaavat automaattisista systeemitesteistä, joilla testataan kehittettävää toiminnallisuutta. Viikottaisen syklin alussa kirjoitetaan testejä, joilla testataan viikon aikana toteutettavaksi valittuja tarinoita (user stories). Testaajat auttavat ja tukevat myös ohjelmoijia ja asiakkaita testien hahmottamisessa ja laatimisessa.

- *Vuorovaikutuksen suunnittelijat* (interaction designers) valitsevat kehitettävän systeemin sisältämät metaforat ja kirjoittavat mahdollisesti tarinoita (user stories) tai voivat auttaa asiakkaita hahmottelemaan ja terävöittämään tarinoita. He eläytyvät käyttäjän rooliin. Voitaneen puhua jonkinlaisesta käytettävyyssuunnittelusta.

- *Arkkitehdit* suunnittelevat järjestelmän laajempia osia. He toimivat myös ohjelmoijina ja kirjoittavat testejä. Testit voivat olla sellaisia, jotka testaavat ohjelmiston arkkitehtuurisia piirteitä kuten suorituskykyä.

- *Projektipäällikkö* ohjaa projektia pitämällä projektin asiakirjat ajan tasalla, tiedottamalla projektin tilasta ulospäin ja muistuttamalla projektilaisia projektin kulusta. Projektipäällikkö myös ohjaa ja luo kontakteja projektin ja ulkomaailman välillä.

- *Tuotepäällikköllä* (product manager) tai tuotevastaavalla on yleiskuva kehitettävästä tuotteesta, jonkinlainen liiketoiminnallinen näkemys. Hän kirjoittaa käyttäjätarinoita ja ylläpitää suhteita kehittäjien ja asiakkaiden välillä. Hän vastaa siitä, että ohjelmisto täyttää asiakkaiden valitsemien tarinoiden vaateet ja pitää samalla kokonaisuuden yhtenäisenä ja markkinoitavana. Hän tukee kehitystyötä vastailemalla ohjelmoijien kysymyksiin ja tarkentamalla epäselviä vaatimuksia ja kertomuksia.

- *Tekniset kirjoittajat* ovat vastuussa käyttöohjeiden kirjoitamisesta. Kirjoittajat pääsevät näkemään kehitettävän ohjelmiston toiminnallisuudet varhaisessa vaiheessa kirjoittaessaan ohjeita ja voivat antaa tärkeää palautetta. He ovat

mahdollisesti yhteydessä käyttäjiin ja voivat saada palautetta ohjelman käyttökokemuksista suoraan käyttäjiltä.

- *Ohjelmoijat* ohjelmoivat eli toteuttavat tarinoita (user stories). Kehittävät voivat myös kirjoittaa tarinoita ja kirjoittaa testejä. Ohjelmoijat valitsevat tarinoita toteutettavaksi kunkin kehitysjakson alussa.

- *Asiakkaat* kirjoittavat käyttäjäkertomuksia (user stories) ja valitsevat niitä kehitettäväksi. Asiakas vastaa hyväksymistestauksesta, tähän kuuluu testien määrittäminen ja suorittaminen. Asiakkaiden tulee olla myös valmiita tekemään liiketoiminnallisia päätöksiä ja selventämään vaatimuksia.

XP-menetelmän alkuaikoina, (Beck 2000) määriteltiin "läsnäolevan asiakkaan" rooli (onsite customer). Läsnäoleva asiakas on henkilö, jolla on tieto siitä mitä kehitettävältä ohjelmalta vaaditaan, ja hän on valmiina vastaamaan kehittäjien kysymyksiin ja tarkentamaan vaatimuksia. Läsnäolevan asiakkaan tulee tuntea ohjelmiston kohdealue hyvin. Hänellä tulee olla riittävästi päätäntävaltaa edustamansa tahon puolelta, jotta hän pystyy tekemään nopeita kehitystyötä ohjaavia ja kehitystyön etenemisen kannalta oleellisia päätöksiä ilman, että kehitystyö keskeytyy. Läsnäolevan asiakkaan roolista saa helposti sen kuvan, että paikkaan vaaditaan vain yksi henkilö (Martin 2009). Yhden henkilön sijaan on parempi puhua asiakastiimistä; on yleistä, että XP:n asiakkaalle määrittämiä tehtäviä hoitaa useampi henkilö yhdessä (Martin 2009).

Menetelmän uudemmassa kuvauksessa läsnäoleva asiakas terminä on pudotettu pois (Korkala ym. 2006). Läsnäolevan asiakkaan sijaan puhutaan ns. oikean asiakkaan mukanaolosta (real customer involvement) (Beck ym. 2004). Tällä tarkoitetaan, että kehitykseen kannattaa ottaa mukaan ne, kenelle ohjelmistoa tehdään – ohjelmiston tulevat käyttäjät – ja joiden liiketoimintaan ja elämään ohjelmistolla on vaikutusta. Idea ja motivaatio tämän taustalla on, että näin vältettäisiin turhaa työtä ja luodaan samalla parempi ohjelmisto, joka täyttää asiakkaan tarpeet. Ohjelmistoa tarvitsevien asiakkaiden on päästävä suoraan yhteyteen ohjelmistonkehittäjien kanssa (Beck ym. 2004). Ajatuksena tämä on jotakuinkin sama kuin läsnäoleva asiakas, mutta tapa jolla se

tuodaan esiin on ehkä hienovaraisempi ja pehmeämpi.

Beck ym. (2004) rohkaisee kommunikaatioon asiakkaan kanssa, koska "he ovat niitä joita yritetään miellyttää". Hän myös toteaa, että tietoa hukataan vääjäämättä mikäli käytettävissä ei ole asiakasta lainkaan tai asiakkaan ja kehittäjien välillä on jonkinlainen asiakasrajapinta. Toisin sanoen, jos ei tunneta ohjelmiston vaatimuksia tarpeeksi hyvin, voidaan päätyä kehittämään hyödyttömiä ominaisuuksia tai määrittämään hyväksymistestejä, jotka eivät todellisuudessa ole yhteydessä hyväksymiskriteereihin. Mitä kauempana todellisesta asiakkaasta ollaan, sitä enemmän lopulta päädytään nojaamaan olettamuksiin. On myös tärkeää tehdä ohjelmistosta asiakkaan tarpeita vastaava pyrkimällä auttamaan asiakasta vaatimusten määrittämisessä (van Deursen 2001). Korkala ym. (2007) tutki viestintää hajautetuissa ohjelmistonkehitystilanteissa. Tutkimuksessa käytetään termiä hyvin määritelty asiakas (well defined customer) ja korostetaan, että asiakkaan täytyy näkyä selkeänä kokonaisuutena kehittäjille. Artikkelissa todetaan, että tehokas yhteistyö asiakkaan kanssa on avain menestykseen, hyvin määritelty asiakas on aktiivinen ja osallistuu jatkuvasti kehitystyöhön.

3.2 Scrum

Scrumin alkujuuret löytyvät Japanista. Ensimmäisen kerran Scrum mainittiin 1980-luvun puolessa välissä artikkelissa, jossa kuvailtiin itseorganisoituvaa prosessia (Schwaber ym. 2001). Ken Schwaberin ja Martin Beedlen kirja Scrum-prosessista vuodelta 2001 on kuitenkin muodostunut menetelmän viralliseksi kuvaukseksi. Scrum on tuotekehitysmenetelmä, joka koostuu käytännöistä ja säännöistä joita voidaan käyttää kehitystyöstä saatavan hyödyn maksimointiin (Kane ym. 2002). Scrum-prosessin käytännöillä voidaan ohjata kehitystyötä siten, että itse kehitystyö rauhoitetaan välillä tietyksi ajaksi. Prosessin ideana on saada investoinneille katetta mahdollisimman varhain (Kane ym. 2002). Scrum-menetelmä pyrkii tarjoamaan ketterän tavan hallinnoida ohjelmistonkehitysprojekteja pyrkien lisäämään todennäköisyyttä, jolla ohjelmiston kehitys onnistuu. Scrum-menetelmä ei sisällä mitään käytännön ohjelmistotuotannollisia ohjeita tai käytäntöjä toisin kuin esimerkiksi XP-menetelmä (Salo ym. 2008).

Scrum nojaa ajatukseen, että kehitystyössä on monia muuttujia, joita ei pystytä hallitsemaan. Se pyrkii antamaan apuvälineitä tämän epävarmuuden keskelle, keinoja joilla muutoksiin pystyttäisiin mukautumaan. Scrum kontrolloi ohjelmistonkehitysprosessia siten, että päästäisiin mahdollisimman hyvään lopputulokseen. Scrum-menetelmä on empiirinen prosessinhallintamenetelmä, se perustuu ympäristöstä tehtyihin havaintoihin ja mukautuu niiden mukaan. (Schwaber 2004).

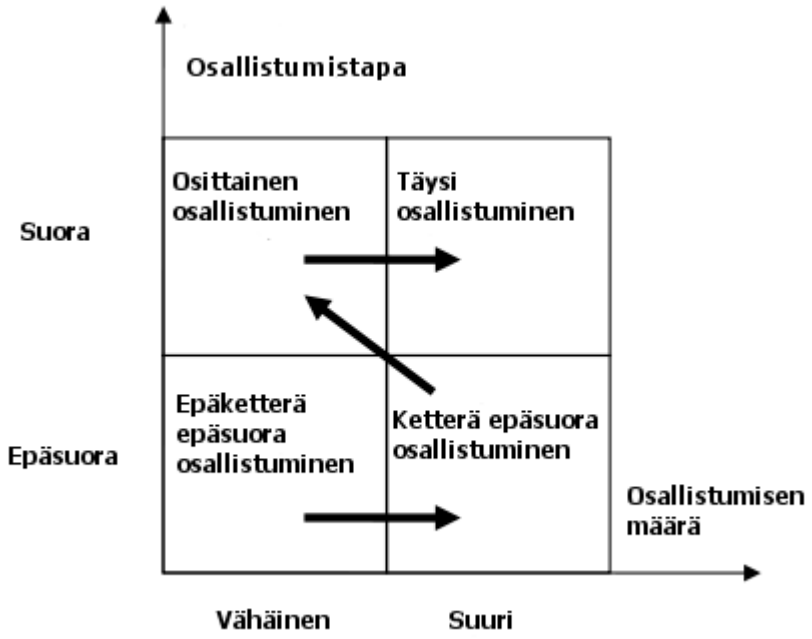
Scrum-menetelmässä ohjelmistoa kehitetään 30 kalenteripäivän jaksoissa, joita kutsutaan sprinteiksi. Scrum-prosessin alussa on jonkinlainen kuva siitä mitä halutaan tehdä, esimerkiksi lista ohjelmiston vaatimuksista. Tämä vaatimuslista, joka pitää sisällään toiminnallisia ja ei-toiminnallisia vaatimuksia, on nimeltään tuotetehtävälista (Product Backlog). Tässä listassa on listattu tärkeysjärjestyksessä kaikki kehitettävän ohjelmiston sillä hetkellä tiedossa olevat vaatimukset. Tätä vaatimuslistaa täydennetään projektin edetessä vaatimusten tarkentuessa tai muuttuessa. Tuotetehtävälista on Scrum-menetelmän selkäranka. Yhden sprintin aikana toteutetaan kehittäjien jakson alussa tuotetehtävälistasta valitsemat vaatimukset. Sprintin aikana tiimi kokoontuu päivittäin lyhyeen tilannepalaveriin (Daily Scrum meeting). Tämä kokous kestää korkeintaan viisitoista minuuttia, ja sen aikana kukin tiimin jäsen vastaa kolmeen kysymykseen. Kysymykset ovat: "Mitä olen tehnyt edellisen tilannepalaverin jälkeen?", "Mitä aion tehdä seuraavaksi?" ja "Onko jotain esteitä, jotka estävät etenemisen?". Tämän lyhyen tilannepalaverin aikana vain tiimin jäsenet ja Scrum-mestari saavat puhua, kaikkien muiden osallistujien on tyydyttävä vain kuuntelemaan. Kehitysjakson jälkeen aikaansaannokset esitellään ennen seuraavan jakson alkua tuotteen omistajalle, mahdollisille asiakkaille, käyttäjille, ylipäätään kaikille kiinnostuneille asianosaisille. Tämän kehitysjakson aikaansaannokset kokoavan sprintin katselmointikokouksen (Sprint Review meeting) aikana on mahdollista kommentoida kehitettyjä toiminnallisuuksia sekä esittää kysymyksiä ja kehitysehdotuksia. Kehitystyö päättyy kun tuotetehtävälistassa ei ole enää vaatimuksia, tai jos projekti päätetään lopettaa muista syistä. (Schwaber ym. 2001, Schwaber 2004).

Scrum määrittelee kolme roolia, jotka ovat tuotteen omistaja, Scrum-tiimi ja Scrum-mestari (Schwaber 2004). Seuraavassa lyhyesti kustakin roolista:

- Scrum-mestari (Scrum Master) on vastuussa Scrum-prosessista ja johtaa esimerkiksi Scrum-prosessiin kuuluvia kokouksia. Hän opastaa prosessiin liittyvissä kysymyksissä ja seuraa, että muut osallistujat noudattavat Scrumin käytäntöjä. Scrum-mestarin vastuulla on sulauttaa prosessi yrityksen kulttuuriin siten, että prosessista saadaan haluttu hyöty irti ilman suurta häiriötä. Hän on myös vastuussa kaikenlaisten esteiden poistamisesta, etenkin Scrum-tiimin ja tuotteen omistajan väliltä. Scrum-mestari vastaa siitä, että prosessista saadaan mahdollisimman suuri hyöty, jotta asetetut tavoitteet saavutettaisiin. (Schwaber 2004).
- Scrum-tiimi on vastuussa toiminnallisuuden kehittämisestä. Tiimiin kuuluu ohjelmistokehittäjiä, jotka yhdessä tuotteen omistajan kanssa valitsevat sprintin alussa tuotetehtävälistalta seuraavaksi toteutettavat vaatimukset. Tiimiin voi kuulua myös muita henkilöitä kehittäjien lisäksi esimerkiksi teknisiä kirjoittajia. Tiimillä on periaatteessa vapaat kädet päättää kuinka valittu toiminnallisuus toteutetaan ja se voi sen mukaan organisoitua ja toimia. Ideana on, että tiimi yhdessä ottaa vastuun kehitystyöstä ja pyrkii toteuttamaan seuraavalle sprintille asetetut tavoitteet. (Schwaber 2004).
- Tuotteen omistaja (Product Owner) on virallisesti vastuussa kehitettävästä ohjelmasta. Hänen tärkein tehtävänsä on ylläpitää tuotetehtävälistaa. Tuotteen omistaja huolehtii, että listalla olevat toiminnalliset ja ei-toiminnalliset vaatimukset ovat tärkeysjärjestyksessä ja siinä muodossa, että niitä voidaan valita listalta toteutettavaksi sprintin alussa. Tuotteen omistaja valitaan hallinnon ja asiakkaan toimesta projektin alussa. Tuotteen omistajan rooli on Scrum-prosessissa rooli jonka kautta asiakas pääsee projektiin mukaan, jos niin haluaa. (Schwaber 2004).

Svenbrant (2008) hahmottelee asiakkaan luokittelua Scrum-prosessissa sen perusteella, kuinka paljon asiakas on mukana kehitysprosessissa. Ajatuksena on, että asiakas voi olla osa Scrum-prosessia joko suoraan tai epäsuorasti. Lisäksi asiakkaan osallistumisen syvyys voi vaihdella, toisin sanoen asiakas voi olla mukana enemmän tai vähemmän

aktiivisesti. Asiakkuudet voidaan jakaa tämän ajatuksen mukaan karkeasti neljään ryhmään. Kuvassa 1 on havainnollistettu jakoa. Jako eri ryhmiin tehdään mukanaolon mukaan. Kuvassa oikealle ylös sijoittuvat asiakkaat ovat vahvimmin mukana ohjelmistonkehityksessä. Vastaavasti kuvassa alhaalle vasemmalle sijoittuvat asiakkaat ovat vähiten osallisina projektissa ja tietävät kehitysprojektin yksityiskohdista vähiten.



Kuva 1: Scrum asiakkaan luokittelu järjestelmä (SCS) (Svenbrant 2008, s. 28)

Suoraan mukanaolevat asiakkaat (Kuva 1 ylärivi) ohjaavat vahvasti kehitystä priorisoimalla ja hallinnoimalla tuotetehtävälistaa (Product Backlog). He myös osallistuvat Scrum-menetelmän mukaisiin tapaamisiin, esimerkiksi sprintin katselmointikokouksiin (Scrum Review meeting). Suora osallistuminen toimii silloin, kun projektilla on vain yksi asiakastaho tai yksi selkeä avainasiakas ja muita pienempiä. Suoraan mukanaolevat asiakkaat voidaan erotella osallitumisaktiivisuuden mukaan. Asiakkaat voivat kuulua projektiryhmään, tällöin sitoutuminen projektiin on vahvimmillaan. Asiakkaan edustaja toimii tällöin tuotteen omistajana. Asiakkaat voivat olla samassa tilassa projektin kanssa ja asiakkailla on suora näkyvyys kehittäjäorganisaatioon. Asiakkaan täytyy tällöin tuntea hyvin Scrum-prosessi sekä kehitettävä tuote sekä tuotteen aihealue, jotta hän voi tukea kehitystyötä. Vähäisemmän

suoran osallistumisen tapauksessa asiakas toimii tuotteen omistajana ja vastaa siten vaatimusten priorisoinnista, mutta hänellä on tukenaan kehittäjäorganisaatiossa kehittäjäorgainsaation projektipäällikkö tai avustava tuotteen omistaja. He toimivat välikappaleena tuotteen omistajana toimivan asiakkaan ja kehittäjien sekä muiden asianosaisten välillä. Suoraan mukana olevat asiakkaat ovat mukana Scrum-prosessissa, tietävät että se on käytössä ja osallistuvat itse prosessiin toimimalla tuotteen omistajina ja hallinnoivat tuotetehtävälistaa (Product Backlog). (Svenbrant 2008).

Tilanteesta jossa asiakkaat eivät toimi tuotteenomistajina tai missään muussakaan kolmesta Scrumin roolista, voidaan ajatella että heillä on *epäsuora suhde* (Kuva 1 alarivi) kehitysprojektiin. He eivät ole mukana Scrum-prosessissa yhtä tiivisti jos lainkaan. Tällöin tuotteen omistajana toimii joku kehittäjäorganisaatiosta. Hän edustaa yhtä tai useampaa asiakasta projektissa ja priorisoi asiakkaiden vaatimuksia, toimien rajapintana asiakkaan ja kehittäjien välillä (Fraser ym. 2004). On mahdollista, että asiakkaat ovat tietoisia siitä että projektia kehitetään ketterän menetelmän mukaisesti ja tietävät kehityksen etenevän Sprint-kehitysjaksoissa, joiden alussa valitaan toteutettavat vaatimukset ja lopuksi esitellään aikaansaannokset. Tällöin voidaan puhua, että asiakkailla on *kettärä epäsuora suhde* projektiin. Asiakkaat osallistuvat säännöllisesti tapaamisiin esimerkiksi tuotetehtävälistan priorisointiin ja sprint-katselmointitilaisuuksiin (Sprint Review meeting), joissa esitetellään kehitysjakson tuotoksia. Asiakkaat saattavat osallistua myös päivittäisiin tilannepalavereihin (Daily Scrum meeting), joskin Scrum-prosessin (Schwaber ym. 2001) mukaisesti vain kuuntelijan roolissa. *Alimman osallistumisaktiivisuuden* asiakkaat (Kuva 1 vasen alakulma) eivät tiedä kuinka ohjelmistotuotetta kehitetään, eli minkälainen prosessi on käytössä. Nämä asiakkaat ostavat valmiin tuotteen tai määrittelevät vaatimukset projektin alussa kuten suunnitelmapohjaisissa tai vesiputousmallia noudattavissa ohjelmistonsuunnitteluprosesseissa on tapana. Nämä asiakkaat saattavat tietää, että ohjelmistoa kehitetetään kokoavasti kehitysjaksoissa, asiakkaat eivät kuitenkaan osallistu vaatimusten priorisointiin, esimerkiksi tuotetehtävälistan (Product Backlog) katselmointeihin. Näille asiakkaille saatetaan räätälöidä valmista ohjelmistotuotetta jossain määrin. Tällaisissa tapauksissa kehittäjäorganisaation tuotteeen omistaja (Product Owner) vastaa asiakkaille tehtävien räätälöintien priorisoinnista ja sisällöstä.

Hän on mahdollisesti jossain määrin tekemisisssä asiakkaiden kanssa ja selvittää halutuimpia ominaisuuksia. (Svenbrant 2008).

3.3 Menetelmien yhtäläisyyksiä ja eroja

XP- ja Scrum-menetelmien välillä on vaikea löytää suuria eroja asiakkaan osallistumisen suhteen. Molemmat menetelmät ovat ketteriä menetelmiä. Ketterät menetelmät perustuvat yleisesti kommunikaatioon ja viestintään projektin eri osapuolten välillä. Tärkeäksi mainitaan kasvotusten tapahtuva viestintä joka tarkoittaa, että osapuolten on oltava läsnä samassa tilassa. (Cockburn ym. 2001).

XP-menetelmä määritteli läsnäolevan asiakkaan (Onsite Customer) käsitteen, joka vahvasti ajoi asiakkaan läsnäolon tärkeyttä koko ohjelmiston kehityksen ajan (Beck 2000). XP-menetelmä määrittelee asiakkaalle selkeitä tehtäviä. Asiakkaan tehtävänä on kirjoittaa käyttäjäkertomuksia, joiden pohjalta ohjelmistoa kehitetään. Asiakkaan vastuulla on suunnitella ja suorittaa hyväksymistestaus, jolla varmistetaan, että ohjelmisto toteuttaa käyttäjäkertomuksissa kuvatun toiminnallisuuden. Asiakkaan kuuluu lisäksi valita kehitysjaksoissa toteutettavat käyttäjäkertomukset. (Beck 2000).

Scrum-menetelmässä prosessi on kuvattu tarkemmin (Kane ym. 2002). Siinä on määritelty tarttumakohtia, joiden kautta prosessia voidaan ohjailla ja lopputuotetta mukauttaa. Asiakkaat voivat lisätä vaatimuksia tuotetehtävälistaan. Mikäli asiakas on tuotteen omistajana (Product Owner) vaatimuslista on kokonaan asiakkaan hallittavissa, ja asiakas vastaa silloin myös vaatimusten priorisoinnista (Schwaber 2004). Tuotteen omistajan roolissa asiakas on melko lähellä XP:n määrittämää läsnäolevan asiakkaan roolia. Tuotteen omistajan tehtävänä on vastata tuotetehtävälistasta ja kehityksen etenemisestä. Tuotteen omistajan kuuluu selventää vaatimuksia ja muotoilla niitä toteutuskelpoisiin kokonaisuuksiin (Schwaber 2004). Osallistumalla Scrum-prosessissa (Schwaber ym. 2001) määritettyihin kokouksiin asiakas saa hyvän kuvan kehitystyön etenemisestä. Scrum tarjoaa asiakkaalle mahdollisuuden ohjata toteutuksen kulkua toteutusjakson pituisin väliajoin. Allaolevassa taulukossa on listattu joitain asiakkaiden tehtäviä ja vaikutusmahdollisuuksia XP- ja Scrum- menetelmissä.

Taulukko 2. Asiakkaan tehtäviä ja vaikutusmahdollisuuksia XP- ja Scrum-menetelmissä.

	Asiakkaan tehtävät ja vaikutusmahdollisuudet
Scrum	• vaatimusten lisääminen tuotetehtävälistaan (Product Backlog) • vaatimusten priorisointi jos toimii tuotteen omistajana (Product Owner) • osallistuminen sprintin suunnittelukokouksiin (Sprint Planning Meeting) joissa valitaan seuraavaksi toteutettava toiminnallisuus • osallistuminen sprintin katselmoitikokouksiin (Sprint Review Meeting) joissa esitellään toteutusjakson lopputulos • mahdollisuus seurata päivittäisiä tilannekokouksia (Daily Scrum Meeting)
XP	• käyttäjäkertomusten kirjoittaminen • toteutettavien käyttäjäkertomusten valinta viikottain • hyväksymistestien suunnittelu ja suorittaminen

Scrum-menetelmä tarjoaa määrämuotoisemman tavan seurata projektin etenemistä. Se, kuinka syvällä projektiorganisaatiossa asiakas toimii, on tapauskohtaista eivätkä menetelmät ota siihen juuri kantaa. XP tosin korostaa läsnäolon tärkeyttä (Beck ym. 2004), mutta riippuu tapauksesta onko se mahdollista ja tarpeellista. Scrum-menetelmä mahdollistaa myös asiakkaan aktiivisen osallitumisen; tuotteen omistajana (Product Owner) asiakas pääsee mukaan kehitysprosessiin tavalla, joka vastaa hyvin XP-menetelmän läsnäolevaa asiakasta.

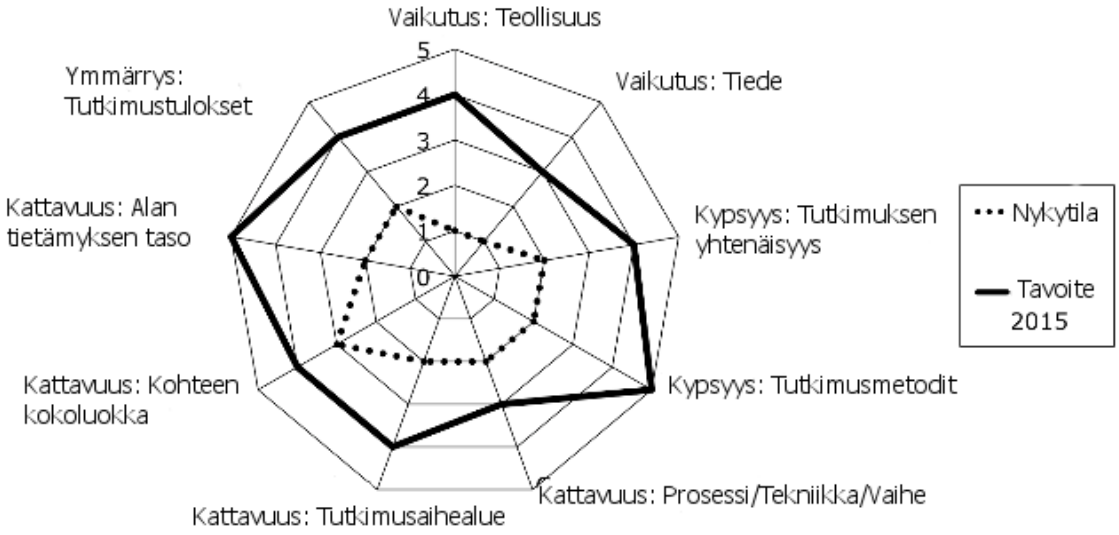
4. KOKEMUKSIA ASIAKKAAN ROOLISTA

Tässä luvussa esitellään tutkimuksiin pohjautuvia kokemuksia asiakkaan roolista ketterissä menetelmissä, erityisesti XP- ja Scrum-menetelmissä. Ensiksi valotetaan ketteriä menetelmiä koskevaa empiiristä tutkimuskenttää yleisesti. Sen jälkeen kerrotaan viidestä empiirisestä tutkimuksesta, jotka ovat koskeneet asiakkaan toimimista ketterissä kehittämisprojekteissa. Lopuksi esitetään yhteenveto havainnoista.

4.1. Ketterien menetelmien tutkimus

Ketterien menetetelmien tutkimus on varsin varhaisessa vaiheessa ja suhteellisen hajanaista. Tarvetta lisätutkimukselle on sekä määrällisesti, että laadullisesti (Dybå ym. 2008). Suurin osa ketterästä tutkimuksesta on keskittynyt XP-menetelmään (Dybå ym. 2008). Useimmat tutkimukset osoittavat, että ketterät menetelmät yleisesti ottaen ovat helposti omaksuttavia ja toimivat hyvin. Menetelmien hyötyinä mainittiin muun muassa asiakkaan osallistuminen ja yhteistyö asiakkaan kanssa. Tutkimuksissa on havaittu, että ketterissä menetelmissä ihmissuhdetaidot ja henkilöiden välinen luottamus ovat tärkeässä roolissa. Asiakkaat olivat yleisesti tyytyväisiä ketterien menetelmien tuomaan mahdollisuuteen antaa ja saada palautetta. Toisaalta läsnäolevan asiaakaan rooli taas koettiin vaativaksi ja raskaaksi ja jokseenkin kestämättömäksi paikaksi mikäli roolissa on oltava kauan (Dybå ym. 2009). Dybån ym. (2008) katsauksen mukaan läsnäolevan asiakkaan roolin pitkittäistutkimus oli puutteellista ja liian lyhytjänteistä. Kritiikkiä sai myös se, että tutkimus oli keskittynyt liiaksi yksinkertaisiin ja pieniin projekteihin; ketterien menetelmien soveltamista laajoissa ja monimutkaisissa projekteissa tulisi myös tutkia (Dybå ym. 2009). Tällöin tutkimustulokset tukisivat paremmin tosielämää ja niitä olisi mahdollista hyödyntää ohjelmistoteollisuudessa.

Dingsøyr ym. (2008) ovat hahmotelleet ketterien menetelmien tutkimuksen tilaa ja todenneet sen olevan monilta osin puutteellista. Kuvassa 2 nähdään tutkimuksen nykytilan hahmotelma, joka on hyvin kapea kaikilla osa-alueilla. Artikkelissa todetaan, että XP-menetelmä on ainoa jonka tutkimuksen taso on keskitasoa – minkään ketterän menetelmän tutkimuksen tasoa ei vielä luokiteltu kypsäksi (Dingsøyr ym. 2008).



Kuva 2. Alustava kartta ketterien menetelmien tutkimuksen nykytilasta ja tavoitteesta. (Dingsøyr ym. 2008 s. 91)

Seuraavaksi käydään läpi muutamia empiirisiä tutkimuksia joissa on tarkasteltu asiakkaan roolia ja kokemuksia. Kustakin tutkimuksesta pyritään kertomaan lyhyesti tutkimusasetelma ja käytetyt tiedonkeruumenetelmät sekä asiakkaan tuntemukset.

4.2. XP-asiakas alihankintaprojektissa

Martin ym. (2004) tarkastelee asiakkaan roolia alihankintana toteutettavassa XP-projektissa. Tutkitussa projektissa oli mukana kolme organisaatiota: tilaajaorganisaatio, kehittäjäorganisaatio sekä infrastruktuurin toimittava organisaatio. Tutkimustieto kerättiin haastattelemalla henkilöitä tilaaja- ja kehittäjäorganisaatiosta. Haastateltavia oli kustakin XP-menetelmän ydinroolista: asiakas, kehittäjä, valmentaja ja testaaja (Beck 2000). Kukaan projektin osapuolista ei ollut aiemmin käyttänyt XP-menetelmää. Projektin tavoitteena oli kehittää sisällönhallintajärjestelmä. Asiakkaan roolin valittiin henkilö tilaajaorganisaatiosta. Tällä henkilöllä oli vankka osaaminen alueelta, jolle ohjelmaa kehitettiin, hän tiesi mitä ohjelmiston piti tehdä. Henkilöllä oli myös tuntemus ja osaaminen organisaatiostaan eli siitä kuinka asiat saa hoitumaan. Tällä henkilöllä oli siis rooliin tarvittavaa päätäntävaltaa (Beck 2000). Asiakkaan rooliin valitulla henkilöllä

oli tukenaan henkilö omasta organisaatiostaan hallinnonpuolelta. Tämä henkilö toimi tukena vaatimusten priorisoinnissa ja avusti projektia koskevien päätösten läpiviennissä. Asiakkaalla oli tukenaan myös oman organisaation hyväksymistestaustiimi, joka oli vastuussa käyttäjätarinoiden testaamisesta kehitysjaksojen lopussa. Lisäksi toimittajaorganisaation puolella oli henkilö, joka kirjoitti käyttäjätarinoiden ensimmäiset versiot alustavien määrittelyiden ja asiakasedustajan kanssa käymiensä keskustelujen perusteella. XP:n määrittelemän asiakkaan roolissa oli siis yksi nimetty henkilö, jolla oli alatuntemusta ja päätäntävaltaa (Beck 2000). Henkilö ei kuitenkaan toiminut yksin, vaan hänellä oli tukenaan muita henkilöitä ja yhdessä henkilöt muodostivat asiakastiimin (Martin ym. 2009), joka täytti XP:n asiakkaalle määrittämät tehtävät.

Projekti alkoi käyttäen perinteistä vesiputousmallia, mutta alustavien määrittelyiden jälkeen siirryttiin käyttämään XP-menetelmää. Alustavia määrittelyjä käytettiin ensimmäisten käyttäjätarinoiden pohjana. Projekti noudatti XP:n iteratiivista prosessia. Asiakkaan osalta ei noudatettu täysin läsnäolevan asiakkaan käytäntöä. Nimetty asiakas oli läsnä kehitystiimin kanssa puolet ajastaan. Asiakas koki, että XP:n tapa (Beck ym. 2004) määrittää vaatimuksia projektin edetessä oli hyvä. Käyttäjätarinoita muuttaen ja lisäten asiakas koki pystyvänsä ohjaamaan kehitystyötä. Ohjelmiston kehittyessä vaiheittain asiakas myös pystyi seuraamaan sen kehittymistä ja tekemään korjaavia ja täydentäviä vaatimuksia. XP-prosessille ominainen dokumentaation vähyys (Cockburn 2002) tuli asiakkaalle hienoisena yllätyksenä johtuen siitä, että XP oli uusi menetelmä kaikille osapuolille. Dokumentoinnin vähyys on yksi ketteriä menetelmiä yhdistävä tekijä (Cho 2008).

Hieman poiketen XP-prosessin käytännöstä tässä projektissa kehittäjät kirjoittivat alustavat käyttäjätarinat ja asiakas hyväksyi ne. Yksi asiakkaan avaintehtävistä on ohjata kehitystä priorisoimalla vaatimuksia ja käyttäjäkertomuksia (Beck ym. 2004). Asiakas koki tämän tehtävän uuvuttavaksi ja raskaaksi (Martin ym. 2004). Asiakas huomasi myös priorisoinnin tärkeyden ja sen, että priorisointi kannattaa aloittaa heti kehitysprosessin alussa ja jatkaa sitä koko kehitysprosessin läpi. XP:n ajatuksena on toteuttaa vaatimuksia ja käyttäjäkertomuksia ts. tuottaa ohjelmistoon toiminnallisuutta liiketoiminnallisessa tärkeysjärjestyksessä (Beck 2000).

Nimetty asiakas suhtautui positiivisesti XP:n käyttöön ja olisi valmis myös jatkossa käyttämään sitä (Martin ym. 2004). Epävarmuutta kuitenkin aiheutti se, kuinka pitkään roolissa pystyisi toimimaan. Asiakas koki, että hän teki usean henkilön edestä töitä eikä rooliissa jaksaisi loputtomiin. Asiakas olisi halunnut olla läsnä 100% ajastaan kehittäjien kanssa, mutta hänen piti olla tekemisissä myös ohjelmiston loppukäyttäjien ja oman kotiorganisaationsa kanssa. Lisäksi tehtävänä oli järjestellä asioita kehittäjäorganisaation ja infrastruktuurin toimittavan organisaation välillä. Nimetyn asiakkaan vastuulla oli tehdä liiketoiminnalliset päätökset ja priorisoida vaatimukset ja kertomukset (Beck 2004). Noin 33% ajastaan hän oli läsnä kehittäjien luona. Jos asiakas olisi ollut enemmän läsnä kehittäjillä olisi parempi mahdollisuus tulla esittämään kysymyksiä ja tarkennuksia. Tässä projektissa asiakkaan piti kuitenkin lisäksi olla yhteydessä loppukäyttäjiin ja hoitaa yhteyksiä oman organisaationsa hallintoon, joten aika ei riittänyt enempään. Tästä syystä asiakas koki roolinsa pitkällä tähtäimellä hieman kestämättömäksi.

4.3. Läsnäoleva XP-asiakas

Koskela ym. (2004) tutki läsnäolevan asiakkaan roolia projektissa, jossa toteutettiin neljän kehittäjän voimin tutkimustulosten hallintaohjelmisto. Kehittäjät olivat viidennen ja kuudennen vuoden tietotekniikan opiskelijoita. Tutkimuksessa seurattiin asiakaan ajankäyttöä ja sen jakautumista. Tietoa kerättiin kehittäjien ja asiakkaan kirjoittamista projektipäiväkirjoista, jälkipalaverien nauhoituksista sekä haastattelemalla kehittäjiä. Kehittäjät eivät olleet aiemmin käyttäjeet XP-menetelmää. Kehittäjät tutustuivat menetelmään itsenäisesti lukemalla aihetta käsittelevää kirjallisuutta, ja lisäksi aiheesta järjestiin kaksipäiväinen koulutustilaisuus.

Kehittäjät ja asiakas työskentelivät samassa tilassa XP-käytännön mukaisesti (Beck 2000). Toinen tutkimusartikkelin kirjoittajista toimi asiakkaan roolissa. Hän osallistui suunnitteluun, hyväksymistestaukseen, jälkianalyyseihin, projektipalavereihin ja tukitehtäviin. Keskimäärin asiakas oli noin 80% ajastaan samassa tilassa kehittäjien kanssa. Projekti koostui kuudesta kehitysjaksosta, kolmesta kahden viikon ja kolmesta

viikon mittaisesta.

Koskela ym (2004) havaitsi, että ajankäytön jakautumisen kannalta vaativimmat eli aikaa vievimmät toimet olivat alun suunnitteluvaihe sekä hyväksymistestaus. Seuraavina tulivat jokaisen iteraarion jälkeen pidetty jälkianalyysitilaisuus ja erilaiset projektipalaverit. Asiakasta häiritsi hieman pariohjelmoinnista syntynyt puheensorina, hän koki vaikeaksi tehdä oikeita töitään. Asiakas myös havaitsi, että kehittäjät tulivat usein kysymään kysymyksiä. Asiakkaan mielestä tämä häiritsi jälleen oikeiden töiden tekemistä. Toki tämän jälkeen todetaan, että läsnäolevan asiakkaan roolissa koko järjestelyn idea on siinä, että asiakas on valmis vastaamaan kysymyksiin ja tavoitettavissa (Beck 2000). Asiakas oli sitä mieltä, että rooli oli vaativa.

Kehittäjät pitivät läsnäolevaa asiakasta pääasiassa hyvänä asiana. Tyytyväisyyttä herätti mahdollisuus kysyä epäselviä asioita samantien ja keskustella epäselvyyksistä sen sijaan, että asiaa joutuisi selvittelemään sähköpostin välityksellä. Mutkaton kommunikaatio asiakkaan ja kehittäjien välillä on eräs ketterien menetelmien perusasioista (Turner ym. 2003). Kehittäjien mielestä asiakkaan ei välttämättä tarvitsisi olla kirjaimellisesti samassa huoneessa kehittäjien kanssa, vaan jossain lähellä helposti ja nopeasti tavoitettavissa (Koskela ym. 2004). Tällöin asiakkaalla olisi myös omaa rauhaa ja hän voisi keskittyä muihin tehtäviin paremmin sillä aikaa, kun projekti ei vaadi hänen osallistumistaan. Kehittäjiltä kysyttiin myös riittäisikö jos asiakas olisi paikalla vain yhden päivän viikossa. Kehittäjät olivat sitä mieltä, että jos asiakas on paikalla vain yhtenä päivänä viikossa, se on liian vähän. Koettiin, että olisi riittävää jos asiakas olisi läsnä päivittäin, muttei välttämättä koko päivää. Tätä tukee myös tutkimuksen tulos jonka mukaan asiakas oli 80% ajastaan läsnä, mutta tästä ajasta vain 20% oli aikaa, jolloin asiakasta tarvittiin kehitystyössä esimerkiksi vastaamaan kehittäjien kysymyksiin ja keskustelemaan asioista. (Koskela ym. 2004).

Asiakkaan ajankäytön jakauminen oli melko oletettua: suunnittelutilaisuudet ja hyväksymistestaus olivat aikaa vievimmät tehtävät (Koskela ym. 2004). Todettiin että asiakkaan olisi ehkä parempi olla toisessa tilassa kuin kehittäjät, kuitenkin helposti tavoitettavissa, jotta asiakas saisi työrauhan. Asiakasta ei kuitenkaan jatkuvasti tarvita

osallisumaan kehitystyöhön, joten menettämättä läsnäolevan asiakkaan hyötyä asiakas saisi mahdollisuuden keskittyä muihin tehtäviin sillä aikaa kun häntä ei tarvita aktiivisesti kehitystyössä. Havaittiin, että läsnäoleva asiakas saattaa luoda liian positiivisen kuvan asiakkaan organisaatioon päin. On tärkeää että asiakas on yhteydessä taustajoukkoihinsa ja saa tukea ja palautetta sieltä päätöksiensä tueksi (Koskela ym. 2004).

4.4. Viestinnän vaikutus virheiden määrään

Korkala ym (2006) tutki asiakkaan ja kehittäjien vuorovaikutuksen vaikutusta ohjelmiston kehitykseen, erityisesti ohjelmistossa esiintyvien virheiden määrään. Tutkimuksessa oli mukana neljä projektia joissa kussakin oli 5-6 kehittäjää, näistä kolmessa oli osittain läsnäoleva asiakas ja neljännessä läsnäoleva asiakas.. Kolmessa projekteista kehitettiin mobiiliohjelmistoja, neljännessa internet-sovellusta. XP ei ollut ennestään tuttu ryhmille. Aluksi järjestettiin koulutus, jossa tutustutiin ketteriin menetelmiin. Ensimmäisessä projektissa asiakas oli läsnä noin 80 % ajasta, toisessa asiakas oli samassa rakennuksessa kehittäjien kanssa, kolmannessa projektissa asiakas oli samassa kaupungissa. Neljännessä projektissa asiakas oli eri kaupungissa.

Väärin ymmärretyt tai jopa virheelliset vaatimukset ovat yleisesti syynä ohjelmistoprojektien viivästymiseen ja ohjelmistojen toimimattomuuteen (Bernstein 2005). Tutkijat väittivät, että tehokas ja rikas iteraatioiden aikainen kommunikaatio asiakkaan ja kehittäjien välillä on erityisen tärkeää silloin kun käytettävissä ei aina ole rikasta kommunikaatioväylää (Korkala ym. 2006). Rikkain kommunikaatioväylä on kasvotusten keskustelu (Keil ym. 1995).

Tukimuksessa selvisi, että noin 25 % kaikista projektin tehtävistä liittyi virheiden korjaamiseen projekteissa, joissa asiakas ei ollut läsnä. Yhden projektin tulokset viittasivat siihen, että noin 60 % virheistä olisi voitu välttää mikäli asiakas olisi ollut läsnä tai mikäli kehitysjaksojen aikainen kommunikaatio olisi ollut aktiivista ja käyttänyt tehokkaampia kommunikaatiomenetelmiä. Tämä viittaa siihen, että tilanteissa

joissa asiakas on vain osittain läsnä tai kokonaan poissa, iteraatioiden aikaiseen kommunikaatioon tulee kiinnittää erityistä huomiota, jotta kehitettävä ohjelmisto pysyy oikeassa suunnassa. Tutkimuksessa nojattiin MRT-teoriaan (Daft 1986), joka luokittelee erilaisia kommunikaatiokanavia. Kasvokkain keskustelu on kaikkein rikkain tiedonvaihtomenetelmä. Sitä seuraa videoneuvottelu, puhelinkeskustelu sekä sähköposti. Kanavien sopivuus riippuu tilanteesta, jossa sitä käytetään (Keil ym. 1995). Ketterät menetelmät kannustavat läheiseen kanssakäymiseen asiakkaan kanssa, sekä kehitystiimin sisällä (Cockburn ym. 2001). Mikäli kasvotusten keskustelu ei ole mahdollista, pitää huolehtia erityisesti siitä, että palautteen antaminen on mahdollista. (Korkala ym. 2006).

Ensimmäisessä projektissa kasvokkain kommunikointi oli yleistä, toisessa projektissa se oli tarvittaessa mahdollista, kolmannessa projektissa ajoittaista ja neljännessä projektissa vähäistä. Iteraatioiden aikainen kommunikaatio hoitui ensimmäisessä projektissa kasvotusten, toisessa projektissa kasvotusten ja sähköpostin välityksellä, kolmannessa projektissa sähköpostilla. Neljännessä projektissa kommunikoitiin sähköpostilla ja puhelimitse. Havaittujen ohjelmistovirheiden määrä noudatti oletettua asetelmaa, ensimmäisessä projektissa virheitä ja epäselvyyksiä oli vähiten, ja neljännessä projektissa, jossa kehittäjät olivat vähiten yhteydessä asiakkaaseen ja joutuivat kommunikaatiossa turvautumaan eniten sähköpostiin ja puhelimeen, epäselvyyksiä ja virheitä oli eniten (Korkala ym. 2006).

Tutkijat toteavat, että kasvotusten keskustelu on ketterien menetelmien oletuskommunikaatiomuoto ja sitä tulisi pyrkiä käyttämään (Korkala ym. 2006). Videoneuvottelulla voi yrittää paikata tämän puutetta, jos muuta vaihtoehtoa ei ole. Kaikilla kommunikaatiomuodoilla on toki hyvät ja huonot puolet, mutta kasvotusten välittyy eniten informaatiota (Keil ym. 1995). Tutkimuksen mukaan tukeuduttaessa vähemmän informatiivisiin viestintäkeinoihin, kuten sähköpostiin, kehitettävän ohjelmiston virheiden määrä kasvaa samassa suhteessa viestintäkanavan informatiivisuuden heikkenemisen kanssa (Korkala ym. 2006). Tämä tukee läsnäolevan asiakkaan (Beck 2000) tärkeyttä. Asiakkaan ollessa läsnä vain osa-aikaisesti on kiinnitettävä huomiota käytettäviin viestintämenetelmiin, jotta asiakkaan toiveet

välittyvät mahdollisimman hyvin kehittäjille (Keil ym. 1995).

4.5. Scrumin käyttöönoton vaikutus asiakastyytyväisyyteen

Mann ym (2005) seurasivat tutkimuksessaan Scrum-prosessin käyttöä ja käyttöönottoa kahdessa ohjelmistoprojektissa kahden vuoden ajan. Tutkimus jakautui kahteen osaan, määrälliseen ja laadulliseen. Määrällisessä osassa seurattiin kehittäjien ajankäyttöä ja Scrum-prosessin vaikutusta ylityötuntien määrään. Laadullisessa osiossa tarkkailtiin Scrum-prosessin vaikutusta asiakkaan tyytyväisyyteen. Seuraavassa käydään läpi vain tutkimuksen asiakkaaseen liittyvää osuutta.

Laadullisen tutkimuksen osuuden tiedot kerättiin kyselyillä, ja lisäksi projektitiloissa paikalla olleen tutkijan havainnot kirjattiin ylös. Projektissa työskenteli keskimäärin kuusi kehittäjää. Kaikki kehittäjät olivat samassa toimistokerroksessa. Myös projektin asiakas oli samassa rakennuksessa. Projektin projektipäällikkö toimi Scrum-mestarina ja asiakkaan edustaja oli tuotteen omistaja (Product Owner). Scrum-käytännöt (Schwaber 2004) otettiin käyttöön kokonaisuudessaan. Asiakkaita oli kolme, joista osa oli ollut mukana aikaisemmissa projekteissa ennen Scrum-prosessin käyttöönottoa.

Asiakkailta saatu palaute oli pääasiassa positiivista. Heidän mielestään Scrum oli tuonut parannusta ohjelmistonkehitysprosessiin. Asiakkaat kertoivat, että heidän suhtautumisensa ohjelmistoon oli muuttunut. He olivat kiinnostuneempia siitä mitä tehtiin, koska pääsivät vaikuttamaan kehitystyöhön Scrum-prosessin kautta. Aiemmin asiakkaiden osallistuminen oli ollut huomattavasti vähäisempää, he olivat vain lopuksi tarkistaneet projektin lopputuotteen. Scrum-prosessin käyttöönotto lisäsi asiakkaiden osallistumismahdollisuuksia mikä johti myös siihen, että asiakkaat olivat tyytyväisempiä kehitettyyn ohjelmistoon. Ketterissä menetelmissä asiakkaan osallistuminen on yksi tärkeistä tekijöistä (Chow ym. 2007). Scrum toi asiakkaille mahdollisuuden seurata projektia päivittäin Scrum tilannekokouksissa. Sprintin suunnittelukokous (Sprint Planning Meeting) koettiin hyväksi tilaisuudeksi, joskin raskaaksi. Vaikka suunnittelutilaisuus koettiinkin raskaaksi, siitä saadut hyödyt koettiin vielä suuremmiksi,

mikä teki tilaisuudesta vaivan arvoisen. Säännöllisten kokousten hyvä puoli on, että kaikille projektin osapuolille tulee jokseenkin yhteneväinen kuva projektin vaatimuksista sekä rajoituksista, joita kehitystyöllä mahdollisesti on (Paasivaara ym. 2009). Tämän myötä kehitystyöhön kohdistuvat odotukset ovat yhteneväiset sekä kehittäjillä että asiakkailla. Myös sprintin katselmointikokous koettiin hyväksi, koska sen aikana kehitettävän ohjelmiston tila tuli näkyviin ja toteutusta oli mahdollista kommentoida ja muuttaa kehityksen suuntaa tarvittaessa (Schwaber 2004).

Kehittäjät kokivat, että Scrum-prosessin käyttöönotto oli lisännyt asiakkaiden osallistumista projektiin ja mahdollisti paremman kommunikaation asiakkaiden ja kehittäjien välillä. Suora kasvokkain tapahtuva viestintä on paras, koska tällöin häiriötekijöitä on vähiten ja informaatiota siirtyy eniten (Keil ym. 1995). Joskus ongelmia aiheutti se, että asiakkaat eivät tienneet mitä halusivat tai eivät osanneet kertoa, mitä haluavat. Tunne siitä, että asiakas tietää missä projekti kulloinkin on menossa lisäsi kehittäjien luottamusta kehittämäänsä ohjelmaan. Scrum-prosessi toi varmuutta siitä, että oltiin kehittämässä sellaista ohjelmistoa, jonka asiakas haluaa. (Mann ym. 2005).

4.6. Scrum-menetelmän vaikeudet

Cho (2008) tutki yritystä, jossa Scrum-menetelmä oli käytössä lukuisissa pienissä ja keskikokoisissa ohjelmistoprojekteissa. Tutkimusaineisto kerättiin haastattelemalla yhdeksää yrityksen työntekijää; haastateltujen joukossa oli kaksi johtajaa, projektipäällikkö, Scrum-mestari ja viisi kehittäjää. Haastattelut paljastivat lukuisia haasteita ja ongelmakohtia projekteissa, joissa Scrum-prosessi oli käytössä.

Havaittiin, että Scrum-menetelmän käytännöt (Schwaber 2004) tukevat tiimin sisäistä viestintää hyvin (Cho 2008). Päivittäinen Scrum-tilannepalaveri mahdollistaa tiimin jäsenten päivittäisen vähittäisviestinnän. Kommunikaatio asiakkaan kanssa voi kuitenkin olla ongelmallista. Kehittäjät valittivat, että asiakkaan kanssa ei ollut mahdollisuutta keskustella tai saada palautetta. Osittain tämä ongelma johtui siitä, että

asiakkailla oli myös muita päivittäisiä työtehtäviä. Projektipäällikkö mainitsi, että asiakkaat eivät osallistuneet päätöksentekoon. Myös hän oli sitä mieltä, että asiakkaalta ei saada riittävästi huomiota vaikka sitä pyydetään. Kehittäjien oli tyydyttävä paljon vähäisempään kanssakäymiseen asiakkaan kanssa kuin he toivoivat. Svenbrantin (2008) luokittelun mukaan asiakkaan osallistuminen voitaisiin luokitella epäsuoraksi epäketteräksi asiakkuudeksi.

Haastattelujen mukaan näytti siltä, että asiakkaalla ei ollut selkeää kuvaa siitä mitä he ohjelmistolta haluavat. Tämä esti heitä osallistumasta prosessiin, tai ainakin vähensi osallistumisintoa. van Deursen (2001) toteaa, että asiakkaiden vastuulla on vaatia sellainen ohjelmisto kuin he haluavat, mutta mainitsee myös, että asiakkaita tulee auttaa tässä määrittelytehtävässä kaikin tavoin. Tällä tarkoitetaan sitä, että myös kehittäjien tulee toimia aktiivisesti ja pyrkiä auttamaan asiakasta, mikäli selkeästi on jotain epäselvyyttä tai tietämättömyyttä, joka estää eheän kuvan muodostamisen kehitteillä olevasta ohjelmistosta. Yksi haastatelluista Scrum-mestareista valitti, että epäselvien vaatimusten tarkentaminen vei paljon aikaa ja vähensi kiinnostusta kommunikaatioon. Toisaalta hän sanoi, ettei muutakaan vaihtoehtoa ole kuin selvittää epäselvät asiat (Cho 2008). Artikkelissa todetaan, että ennen projektin alkua tulisi neuvotella käytännöt selviksi; tällöin projektin edetessä olisi vähemmän ongelmia esimerkiksi asiakkaan osallitumisen kanssa (Cho 2008).

4.8. Yhteenveto tutkimuksista

Ketterästä menetelmästä riippuen asiakkaalle tarjoutuu erilaisia tapoja olla mukana kehitysprojektissa. XP määritteli läsnäolevan asiakkaan roolin (Beck 2000) ja myöhemmin todellisen asiakkaan osallistumisen (Beck 2004). Tällä korostetaan sitä, että asiakkaan on hyvä olla läsnä tai ainakin valmiina osallistumaan aktiivisesti kehitysprojektiin koko sen elinkaaren ajan. Asiakkaan aktiivinen osallistuminen on elintärkeää mille tahansa ohjelmistoprojektille (van Deursen 2001). Scrum-menetelmä ei määrittele asiakkaalle tehtäviä kuten XP-menetelmä, vaan lähinnä tarttumapintoja joiden kautta asiakas voi osallistua projektiin. Scrum-menetelmässä asiakas voi valita

osallitumisensa tason, asiakas voi ottaa tuotteen kehityksen kokonaan haltuun tuotteen omistajan roolin kautta tai tyytyä osallistumaan kehitystä ohjaaviin ja seuraaviin kokouksiin (Svenbrant 2008).

Ensimmäisessä käsitellyssä tutkimuksessa (Martin ym. 2004) kohteena oli XP-projekti, jossa ohjelmistoa kehitettiin alihankintana. Asiakkaan puolelta oli nimetty edustaja, joka oli läsnä kehitysprojektin kanssa (Martin ym. 2004). Asiakas pyrki olemaan läsnä ja tavoitettavissa mahdollisimman paljon, mutta lukuisat muut tehtävät estivät tämän täyden toteutumisen. Asiakkaan edustajalla oli apunaan joukko muita henkilöitä, ja yhdessä he muodostivat asiakastiimin (Martin ym. 2009), joka suoritti XP-menetelmän asiakkaalle asettamat tehtävät (Beck 2000). On tiedossa, että asiakkaat kokevat roolinsa XP-menetelmässä raskaaksi joskin palkitsevaksi (Dybå ym. 2008). Kaikissa edellä esitellyissä XP-projekteissa asiakkaat kertoivat, että asiakkaan rooli oli raskas, hyvin pitkälti moninaisten tehtävien tuoman vastuun takia, ja pohtivat kuinka kauan siinä jaksaisi olla. Roolin vastuiden selkeä jakaminen usealle henkilölle, eräänlaisen asiakastiimin muodostaminen, voisi tuoda helpotusta tilanteeseen (Martin ym. 2009).

Kahdessa käsitellyssä Scrum-tutkimuksessa oli hyvin erilaiset kokemukset. Ensimmäisessä niistä (Mann ym. 2005) tutkittiin kahden vuoden ajan Scrum-prosessin käyttöönottoa ja käyttöä eräässä ohjelmistotalossa. Asiakkaat olivat tyytyväisiä Scrum-prosessin tarjoamiin mahdollisuuksiin seurata kehityksen kulkua. Scrum-menetelmän erilaiset kokoukset (Schwaber ym. 2001) lisäsivät asiakkaiden mielestä projektin seurattavuutta huomattavasti ja tarjosivat samalla tilaisuuden tavata kehittäjiä (Mann ym. 2005). Asiakkaat eivät toimineet tässä tuotteen omistajan roolissa vaan osallistuivat pelkästään Scrum-menetelmän (Schwaber ym. 2001) määrittelemiin kokouksiin. Siten tämän tutkimuksen asiakkaat voidaan Svenbrantin (2008) esittelemän luokituksen mukaan sijoittaa *ketteriksi epäsuoriksi asiakkaiksi.* Tällä tarkoitetaan asiakkaita, jotka ovat tietoisi käytetystä ketterästä menetelmästä ja osallistuvat ersimerkiksi sprintin suunnittelukokouksiin, mutta eivät toimi tuotteen omistajan roolissa (Svenbrant 2008). Mikäli asiakkaat tai kehittäjäorganisaatio haluaisivat lisätä asiakkaan vastuuta asiakas voisi ottaa tuotteen omistajan roolin ja hallinnoida tuotteen kehitystä tuotetehtävälistan kautta. Yhdeksi suureksi ongelmaksi kehittäjät mainitsivat sen, että asiakas ei aina

tiennyt mitä halusivat tai eivät osanneet kertoa mitä haluavat (Mann ym.2005). van Deursen (2001) puhuu samasta ongelmasta ja kirjoittaa, että vaikka asiakkaiden vastuulla on vaatia oikeanlainen ohjelmisto, niin kehittäjien tulisi pyrkiä auttamaan asiakkaita selkiyttämään näkemystään, mikäli se vain on mahdollista.

Toisessa Scrum-tutkimuksessa (Cho 2008) asiakkaat olivat huomattavasti pienemmässä roolissa. Svenbrantin (2008) luokittelulla asiakas oli alimmalla tai toiseksi alimmalla tasolla. Asiakas ei osallistunut aktiivisesti Scrum-prosessiin vaikkakin tiesi sen olevan käytössä. Sekä kehittäjät että projektipäällikkö kritisoivat asiakkaan vähäistä osallistumista. Asiakkuutta voisi pyrkiä syventämään siten, että asiakas osallistuisi säännöllisesti Scrum-käytännön (Schwaber ym. 2001) mukaisiin kokouksiin (Svenbrant 2008). Tällöin kehittäjien ja asiakkaiden välinen kanssakäyminen tulisi säännöllisemmäksi ja asiakkaan osallistumista saataisiin vahvistettua, jolloin täyttyisi yksi ketterien ohjelmistoprojektien kriittinen menestystekijä (Chow ym. 2007).

5. YHTEENVETO

Asiakas on se, joka maksaa tuotteen kehitystyön, joko heti tai myöhemmin ostaessaan valmiin tuotteen. Kehitystyö perustuu vaatimuksiin, joilla pyritään hahmottamaan kehitettävää ohjelmistoa siten, että valmiista ohjelmistosta olisi liiketoiminnallista hyötyä. Ketterät menetelmät kehittyivät suunnitelmapohjaisten, vesiputousmallia noudattavien suunnittelumenetelmien rinnalle, jotta turvattaisiin menestyksekäs kehitystyö ympäristön vaihteluista huolimatta. Ydinajatus on ollut pyrkiä paremmin kohtaamaan kehitettävää ohjelmistoa määrittävien vaatimusten muuttuminen. Mukautuminen muutokseen toisi hyötyä myös asiakkaalle, koska kehitystyö ei rampautuisi muutosten edessä niin helposti. Ketterät menetelmät lähtevät siitä, että kaikkia vaatimuksia ei välttämättä tiedetä kehitystyön alussa ja jo olemassaolevat vaatimukset voivat matkan varrella muuttua.

Ketterät menetelmät kannustavat kanssakäymiseen asiakkaan kanssa. Asiakkaalle annetaan mahdollisuus seurata ohjelmiston kehitystyötä ja havaintojen perusteella tehdä korjauspyyntöjä ja tarkentaa vaatimuksia. Toisaalta läheisemmän asiakasyhteistyön tarkoituksena on antaa ohjelmiston kehittäjille mahdollisuus keskustella asiakkaan kanssa vaatimuksista, pyytää selvennystä epäselviin vaatimuksiin tai ehdottaa uusia vaatimuksia tai kehitysehdotuksia. Toimivan keskusteluyhteyden luominen asiakkaiden ja ohjemiston kehittäjien välille tuo lisäarvoa ja mahdollistaa sen, että osapuolet ovat paremmin perillä kehitettävän ohjelmiston tilasta ja siihen kohdistuvista vaatimuksista.

Tutkielman tarkoituksena oli selvittää, mikä on asiakkaan rooli ketterissä ohjelmistonkehitysprojekteissa, mitä vastuita ja vaikutuksia ketterien menetelmien kuvaama rooli asiakkaalle tuo. Tarkastelun kohteeksi otettiin erityisesti XP- ja Scrum-menetelmät. Lisäksi tarkoituksena oli tuoda esille tuloksia empiirisistä tutkimuksista, joiden kohteena on ollut asiakkaan toiminta ketterässä kehittämistyössä.

Ketteristä menetelmistä XP esitteli käsitteen läsnäoleva asiakas, jolla korostettiin läheisen yhteistyön merkitystä asiakkaan ja kehittäjien välillä. Läsnäoleva asiakas olisi aina paikalla, valmiina selventämään kehitettävälle ohjelmistolle asetettavia

vaatimuksia, mutta myös valmiina mukauttamaan vaatimuksia, jos kehitystyön aikana nousee esiin esteitä tai uusia mahdollisuuksia. Myös Scrum tarjoaa asiakkaalle mahdollisuuksia osallitua kehitystyöhön, Scrum-prosessiin kuuluvat päivittäiset ja kuukausittaiset kokoukset antavat asiakkaalle mahdollisuuden havainnoida kehitystyön etenemistä. Seurattavuus auttaa näkemään, onko kehitystyö vaikeuksissa tai onko vaatimuksissa epäselvää. Scrum-prosessin tuotteen omistajan rooli mahdollistaa asiakkaan hyvinkin aktiivisen osallistumisen kehitystyöhön. Tuotteen omistajana asiakas pääsee hallinnoinmaan vaatimuksia ja priorisoinnin kautta määräämään niiden toteutusjärjestyksen.

Tässä tutkimuksessa käsitellyt empiiriset tutkimukset puoltavat käsitystä siitä, että ketterät menetelmät tukevat kehitystyötä. Tutkimuksissa suurimpana hyötynä nähtiin eri osapuolten läheisempi kanssakäyminen. Voisi sanoa, että ketterät menetelmät johtavat siihen, että asiakkaat ja toteuttajat ovat samalla puolella, molemmat aktiivisesti kehittämässä ohjelmistoa. Ketterät menetelmät antavat asiakkaalle eli tilaavalle osapuolelle mahdollisuuden osallistua ja toteuttavalle osapuolelle mahdollisuuden lisätietojen kyselyyn ja palautteen saamiseen. Eteneminen tapahtuu pienemmissä kokonaisuuksissa. Tutkimuksissa tuli esiin monesti se, että kehitystyöhön osallistuva asiakas on kovan paikan edessä. Asiakkaan on osattava toimia roolissaan oikein ja tunnettava alue jolle ohjelmistoa kehitetään. Monessa tutkituista projekteista asiakkaan rooliin valitulla henkilöllä oli tuntemusta alueesta, jolle ohjelmistoa kehitettiin. Henkilöllä oli myös tukijoukko, joka auttoi roolin vaatimissa tehtävissä, eräänlainen asiakastiimi. Varsinkin XP-menetelmä osoittaa asiakkaalle tehtäviä, kuten hyväksymistestien suorittamisen, tallaisten tehtävien suorittaminen yksin on ylitsepääsemätön taakka. Siksi hyvät taustajoukot ovat avainasemassa.

Tutkielman tulokset auttavat ohjelmistoyrityksiä ja asiakasorganisaatioita kehittämään toimintamalleja, joilla asiakkaan ääni saadaan tilanteeseen sopivalla tavalla kuulumaan järjestelmänkehityksessä. Kuten yllä kerrotusta ilmenee, ei ole olemassa yhtä oikeaa tapaa määritellä asiakkaan rooli, vaan osallistumisen intensiivisyys riippuu monesta tilannekohtaisesta tekijästä.

Jatkotutkimusaiheita voisi löytyä lähestymällä aihetta vaatimusten kautta. Taito hallita, määrittää ja ymmärtää vaatimuksia on keskeinen ohjelmistokehityksessä. Vaatimukset tulevat yleensä asiakastaholta ja yhtenäinen näkemys vaatimuksista on avainasemassa. Ketterät menetelmät mahdollistavat vaatimusten tarkentamisen yhdessä asiakkaan kanssa kehitystyön edetessä. Voisi siis kysyä, mitä ketterät menetelmät tuovat vaatimusten hallintaan ja määrittämiseen. Onko ketteristä menetelmistä hyötyä vaatimusten määrittämisessä ja hallinnassa?

Ketterien menetelmien empiirisen tutkimukseen on kohdistettu jonkin verran kritiikkiä ja toivottu lisää tutkimusta pitkäkestoisista projekteista, joissa on mukana kokeneita kehittäjiä. Jatkotutkimuksena voisi seurata pitemmän aikaa, millä tavalla asiakkaan rooli kehittyy sekä yksittäisen projektin aikana että saman ohjelmistoyrityksen ja saman asiakkaan välisissä jatkoprojekteissa.

LÄHTEET

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2002). Agile software development methods. Review and analysis [online]. Helsinki. VTT, [viitattu 17.12.2008]. VTT:n julkaisuja 478. Saatavilla pdf-muodossa: <URL: http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf>

Agile Alliance. 2001. Agile Manifesto [online]. Agile Alliance, [viitattu 10.5.2010]. Saatavilla osoitteessa: <URL:http://agilemanifesto.org/>

Beck, K. 2000. Extreme programming explained: Embrace change. Reading, Mass., Addison-Wesley.

Beck, K., Andres C. 2004. Extreme Programming Explained: Embrace Change (2nd Edition). Addison Wesley.

Bernstein, L. (2005). Taking Software Requirements Creation from Folklore to Analysis. ACM SIGSOFT Software Engineering Notes 30(5), 3-5.

Boehm, B. (2002). Get Ready For The Agile Methods, With Care. Computer 35(1), 64–69.

Cho, J. (2008). Issues and chalenges of agile software development with Scrum. Issues in Information Systems. VOL IX, No. 2, 188-195.

Chow, T., Cao, D.-B., (2007). A survey study of critical success factors in agile software projects, J. Syst. Software.

Cockburn, A., Highsmith, J. (2001). Agile software development: the people factor. IEEE Computer, 34 November, 131-133.

Cockburn, A, 2002. Agile Software Development. Boston, Addison-Wesley.

Daft, R., L., Lengel, R., J. (1986). Organizational Information Requirements, Media Richness and Structural Design. Manage. Sci. vol. 32, 554-571.

Davis, A.M. (1994). Fifteen principles of software engineering. IEEE Software 11(6), 94-96.

van Deursen, A., 2001. Customer Involvelment in Extreme Programming. XP2001 Workshop Report.

Dingsøyr, T., Dybå, T., Abrahamsson, P., 2008. A Preliminary Roadmap for Empirical Research on Agile Software Development. Teoksessa: Proceedings of Agile 2008. 2008: 83-94.

Dybå, T., Dingsøyr, T. 2008. Empirical studies of agile software development: A systematic review. Inform. Softw. Technol.

Dybå, T., Dingsøyr, T. (2009). What Do We Know about Agile Software Development?. IEEE Software 26(5), 6-9.

Fraser S., Martin A., Hussman D., Mats C., Poppendijck M., Rising L. The XP customer role. In XP 2004.

Highsmith, J. (2002). What Is Agile Software Development?. CrossTalk, The Journal of Defence Software Engineering, 15(10), 4-9.

Highsmith, J. 2004. Agile Project Management: Creating Innovative Products. Boston, Addison Wesley

Kabbedijk, J., Brinkkemper, S., Jansen, S. 2009. Customer Involvelment in Requirements Management: Lessons from Mass Market Software Development. Teoksessa: Proceedings of 17th IEEE International Requirements Engineering Conference. 2009: 281-286.

Kane M., Schwaber , K. 2002 . Scrum with XP [online]. Prentice Hall. [viitattu 10.5.2010]. Saatavilla osoitteessa: <URL: http://www.informit.com/articles/article.aspx?p=26057>

Kaulio, M., A., (1998). Customer, consumer and user involvelment in product development: A framework and a review of selected methods. Total Quality Management, 9, 141-149.

Keil M., Carmel E. (1995). Customer-Developer links in software development, Comm. Of the ACM 38(5), 33-44.

Korkala, M., Abrahamsson, P., Kyllönen, P., 2006. A Case Study on the Impact of Customer Communication on Defects in Agile Software Development. Teoksessa: Proceedings of AGILE'06 Conference. 2006: 76-88.

Koskela, J., Abrahamsson, P. 2004. On-site customer in an XP project: Empirical Results from a Case Study. Teoksessa: Proceedings of European Software Process Improvement Conference. 2004: 1-11.

Mann, C., Mauer, F. 2005. A Case Study on the Impact of Scrum on Overtime and Customer Satisfaction. Teoksessa: Proceedings of the Agile Development Conference. 2005: 70-79.

Martin, A., Biddle, R., Noble, J. 2004. The XP Customer Role in Practice: Three Studies. Teoksessa: Proceedings of Agile Development Conference. 2004: 42-54.

Martin, A., Biddle, R., Noble, J. 2009. The XP Customer Team: A Grounded Theory. Teoksessa: Proceedings of Agile Conference. 2009: 57-64.

McCauley, R. (2001). Agile Development Methods Poised to Upset Status Quo. SIGCSE Bulletin 33(4): 14–15.

Paasivaara, M., Durasiewicz, S., Lassenius, C. 2009. Using Scrum in Distributed Agile Environment: A Multiple Case Study. Teoksessa: Fourth IEEE International Conference on Global Software Engineering. 2009: 195-204.

Salo, O., Abrahamsson, P. (2008). Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum. IET Softw., 2(1), 58-64.

Schwaber, K. 2004. Project Management wiht Scrum. Redmont, Washington, USA: Microsoft Press.

Schwaber, K., Beeble, M. 2001. Agile Software Development with Scrum. Upper Saddler River, New Jersey, USA: Prentice Hall.

Sillitti A., Succi G., 2005. Requirements Engineering for Agile Methods. Teoksessa: Engineering and Managing Software Requirements. 2005: 309-326.

Standish Group. 2002. CHAOS Chronicles Version 3.0. [online], Standish Group. [viitattu 11.5.2010]. Saatavilla osoitteessa: <URL:http://www.standishgroup.com/chaos/intro2.php>.

Svenbrant, H. 2008. Customer value in agile projects. Pro gradu-tutkielma, Faculty of Engineering, Lund University.

Turner R., Boehm B. (2003). People factors in software management: lessons from paring agile and plan-driven methods. Cross Talk, The Journal of Defense Software Engineering 16(12), 4-8.