Haaga-Helia
ammattikorkeakoulu Oy

Asiakasyrityksen tarpeiden määrittely ketterissä ohjelmistokehitysprojekteissa

Minna Tunkelo

Opinnäytetyö
Tietojenkäsittely
2018

Tiivistelmä

Haaga-Helia ammattikorkeakoulu Oy

Tekijä
Minna Tunkelo

Koulutusohjelma
Tietojenkäsittely

Opinnäytetyön nimi
Asiakasyrityksen tarpeiden määrittely ketterissä ohjelmistokehitysprojekteissa

Sivu- ja liitesivumäärä
33 + 2

Kommunikoinnin merkitys ohjelmistokehitysprojekteissa on projektin onnistumisen kannalta huomattava. Tämä opinnäytetyö tarjoaa ideoita asiakaskommunikointiin ketterissä ohjelmistokehitysprojekteissa ja tarjoaa katsauksen yleisimpiin ketteriin ohjelmistokehitysmenetelmiin. Työn tavoitteena oli selvittää, kuinka asiakasyritykseltä saadaan määrittelyt tarpeista riittävällä tasolla ketterää ohjelmistokehitystä varten.

Tässä opinnäytetyössä perehdytään, millaisia menetelmiä toimeksiantajayritys hyödyntää asiakasprojektissaan ja kuinka kommunikaatiota voisi parantaa. Opinnäytetyön toimeksiantajana toimii suomalainen ohjelmistokehitysyritys, ja tapauksena tutkitaan yrityksen asiakkuutta, jossa kehitetään olemassa olevaa web-pohjaista ohjelmistotuotetta.

Tämä opinnäytetyö toteutettiin tapaustutkimuksena ja työ koostuu teoreettisesta kirjallisuuskatsauksesta sekä empiirisestä tutkimusosuudesta. Materiaalia hankittiin yksilöhaastatteluiden ja kirjallisuuskatsauksen avulla. Tutkimuksen aikana paljastui useita kehityskohtia asiakkuuteen liittyvissä käytännöissä, jotka liittyvät ketteriin menetelmiin, vaatimusmäärittelyihin ja kommunikointiin sekä kokonaisuudessaan riskienhallintaan.

Asiasanat
ketterä ohjelmistokehitys, kommunikaatiotavat, ohjelmistokehitys, asiakasviestintä

Sisällys

1 Johdanto

Erilaiset ohjelmistoprojektit ovat merkittävässä roolissa yritysten kilpailukyvyn ylläpitämisessä ja kehittämisessä. Alati muuttuvassa maailmassa markkinoille pääseminen vaatii joustavaa ja tehokasta menetelmää ohjelmiston kehittämiseen sekä nopeaa reagointia asiakkaan toiveisiin.

Opinnäyte tarjoaa ideoita asiakaskommunikointiin ketterissä ohjelmistokehitysprojekteissa ja tarjoaa katsauksen yleisimpiin ketteriin ohjelmistokehitysmenetelmiin. Työn tavoitteena oli selvittää, kuinka asiakasyritykseltä saadaan määrittelyt tarpeista riittävällä tasolla ketterää ohjelmistokehitystä varten. Tässä opinnäytetyössä perehdytään, millaisia menetelmiä suomalainen ohjelmistokehitysyritys hyödyntää asiakasprojekteissaan ja kuinka kommunikaatiota voisi parantaa. Aihetta käsitellään käytännön tasolla kyseisen ohjelmistokehitysyrityksen toteuttaman asiakasprojektin näkökulmasta.

Tutkimusongelmaan pyritään saamaan vastauksia seuraavien tutkimuskysymyksien avulla:

- Miten vaatimusmäärittelyt tulisi toteuttaa ketterässä ohjelmistokehityksessä?
- Millaisia haasteita on toimittajan ja asiakkaan välisessä kommunikaatiossa?
- Miten voitaisiin parantaa toimittajan ja asiakkaan välistä kommunikaatiota?

Tämä opinnäytetyö on rajattu ketterien menetelmien osalta koskemaan ainoastaan Scrumia, Kanbania ja Extreme-ohjelmointia. Opinnäytetyö koskee asiakaskommunikointia, joten ulkopuolelle on rajattu ohjelmistokehitystiimin sisäiseen kommunikaatioon liittyvät asiat. Tässä opinnäytetyössä asiakas on ohjelmistotuotteen tilaaja. Loppukäyttäjät ovat ohjelmistotuotteen käyttäjiä.

Tämä opinnäytetyö on tapaustutkimus ja koostuu teoreettisesta kirjallisuuskatsauksesta sekä empiirisestä tutkimusosuudesta. Teoriaosuudessa tutustutaan aiheeseen olemassa olevien tutkimuksien ja kirjallisuuden avulla.

Tietoperusta on koottu 2010-luvun teoksista, jotka käsittelevät ketteriä ohjelmistokehitysprojekteja sekä muutamista vanhemmista teoksista, jotka sisältävät sellaisia teorioita, jotka ovat edelleen voimassa. Uudempaa tietoa on haettu internetistä löytyvistä tutkimuksista ja tieteellisistä artikkeleista.

Työn empiirinen osa käsittelee suomalaisen ohjelmistokehitysyrityksen asiakkuutta, jossa kehitetään olemassa olevaa ohjelmistotuotetta eteenpäin. Tähän työhön on haastateltu

ohjelmistokehitystiimin vetäjää sekä asiakkaan edustajaa. Opinnäytetyön tekijä työskentelee kyseisessä projektissa ohjelmistokehitystiimissä kehittäjänä. Näiden haastatteluiden pohjalta olen selvittänyt, millaisia käytäntöjä tässä kehitystyössä käytetään ja miten asiakaskommunikointi on toteutettu tässä projektissa.

Haastattelut toteutettiin yksilöhaastatteluina ennakkoon sovittuna ajankohtana. Haastateltavat eivät saaneet tutustua ennakkoon haastattelukysymyksiin (liite 1 ja liite 2). Haastattelut nauhoitettiin ääninauhurilla ja ne litteroitiin haastatteluiden jälkeen.

2 Moderni ketterä ohjelmistokehitysprojekti

Kirjallisuudessa projekti määritellään monin eri tavoin. Mäntyneva (2016) määrittelee projektin ainutkertaiseksi kokonaisuudeksi, joka sisältä rajauksia laajuuteen, kustannuksiin ja aikaan. Ainutkertaisuuden määritelmänä on se, ettei vastaavaa kokonaisuutta ole toteutettu aikaisemmin (Mäntyneva 2016, 13).

Projektille voidaan määritellä kolme kulmakiveä: sisältö, aikataulu ja kustannukset. Projektin alkuvaiheessa olisi hyvä valita tärkein kulmakivi edellä mainituista. Mikäli projekti ajautuisi tilanteeseen, jossa epäonnistumisen uhka kasvaisi merkittävästi, voidaan pitäytyä tärkeimmässä kulmakivessä ja joustaa muiden osalta. (Juvonen 2018, 29.)

On myös olemassa tilanteita, jolloin työn tilaaja ei tunnista tilattua työtä projektiksi vaan ennemminkin työkokonaisuudeksi. Usein tällaisen ajattelumallin taustalla on se, että kokonaisuus halutaan toteuttaa ilman byrokratiaa, joka usein liitetään projekteihin. Yksi syy projektimallin välttelylle voi olla ketteryyden toteuttamisen hankaluus. Tällaisessa tapauksessa olisi syytä saada työn tilaaja ymmärtämään, ettei ennuste tällaiselle projektille mairitteleva. (Juvonen 2018, 27.)

2.1 Modernin ohjelmistokehitysprojektin roolit

Modernissa ohjelmistokehitysprojektissa projektiryhmän roolitus vaihtelee valitun ketterän menetelmän mukaisesti. Ohjelmistokehitysprojekteissa on tyypillistä, että asiakkaalla ja toimittajalla on omat projektipäälliköt, jotka toimivat avainhenkilöinä projektissa. Näiden projektipäällikköjen näkökulmat ovat hyvin erilaiset. Asiakkaan projektipäällikkö vastaa omalle organisaatiolle ohjelmiston käyttöönotosta, kun toimittajan projektipäällikkö vastaa projektin päivittäisen työn ohjaamisesta. (Juvonen 2018, 37.)

Ominaista ketterille ohjelmistokehitys menetelmille on, että asiakas osallistuu ohjelmiston toteutukseen läpi kehityskaaren aktiivisesti ja jatkuvasti. On asiakkaan vastuulla tarjota kehittäjille tietoa, määrittää tärkeysjärjestys kehitettäville ominaisuuksille ja tehdä liiketoiminnalliset päätökset ohjelmiston vaatimuksista. (Bakalova & Daneva 2011, 1).

2.2 Ohjelmistokehitysprojektin elinkaari

Ohjelmiston elinkaarella tarkoitetaan aikaa, joka kuluu ohjelmiston kehittämisen aloittamisesta aina sen poistamiseen käytöstä.

Murch & Kosonen (2002) määrittelee ohjelmistokehityksen kuuteen eri vaiheeseen, jotka ovat projektisuunnittelu, analysointi, suunnittelu, rakennus, testaus ja käyttöönotto. Näiden vaiheiden lisäksi on kolme lisävaihetta, jotka suoritetaan edeltävien vaiheiden kanssa samanaikaisesti. Nämä vaiheet ovat testauksen suunnittelu ja valmistelu, koulutuksen suunnittelu sekä käyttöönoton suunnittelu. (Murch & Kosonen 2002, 59.)

Projektin jakamisella vaiheisiin ja vaiheiden aikataulutuksella on monia etuja projektin lopputuloksen kannalta. Hyvällä valmistautumisella saadaan määriteltyä projektin valmistumiseen tarvittavat työt. Vaiheiden suunnittelu ja aikataulutus saa asiakkaan pitäytymään paremmin alkuperäisessä suunnitelmassa, ja vältytään lisäominaisuuksien tekemiseltä, jotka usein johtavat projektin myöhästymiseen. Vaihejaon avulla on myös mahdollista nähdä, kuinka projekti etenee ja pysyykö se aikataulussaan. (Phillips 2005, 137.)

Oikeellisen työmääräarvion tekeminen vaatii usein kysymyksiä vaatimuksista ja käyttäjätarinoista. Nämä kysymykset auttavat ymmärtämään työn kokonaisuudessaan, jotta tehtävä voidaan pilkkoa pieniksi paloiksi. Kun työmääräarvio on tiedossa, priorisoidaan kehitysjonon tikettejä. Työmääräarvion tekeminen ei saa olla liian eksaktia, sillä muutoksia on haastava ennakoida. Työmääräarvion puuttuminen tai virheellinen tulkitseminen johtaa helposti siihen, että tiimi venyy yli suorituskyvyn, jolloin luovuus, kyky mukautua ja hahmottaa kokonaisuuksia kuolee. (Radigan)

Kun projektiin tulee muutoksia tai lisäominaisuuksia, tulisi aina tehdä arvio muutoksesta ja sen vaikutuksista projektiin. Muutoksia tulee hallita, jotta ne eivät aiheuta projektin leviämistä. Lisäominaisuuden tai muutospyynnön kohdalla olisi tärkeä miettiä seuraavia asioita ja kirjata asiat ylös projektitiimin tietoon:

- muutoksen tarkempi kuvaus, perustelu tarpeellisuudelle
- arvio työmäärästä
- vaikutus projektin aikatauluun
- vaikutus projektin hintaan tai laskutukseen

(Lehtimäki 2006, 48-49.)

Toimivan järjestelmän edellytyksenä on jatkuva testaus ja koekäyttäjät. IT-projekteissa testaaminen tulisi aloittaa hyvissä ajoin suunnitelmallisesti (Murch & Kosonen 2002, 107). Projektin alkuvaiheessa kannattaa suunnitella testausstrategia, jossa nimetään projektin eri vaiheiden testaustarpeet ja kerrotaan niiden tavoitteista (Lehtimäki 2006, 170). Kun kyseessä on ketterä ohjelmistokehitys, on testaustarpeiden määrittely parempi tehdä aina kun isompi kokonaisuus määritellään. Projektille olisi hyvä määritellä kuitenkin testausstrategia, jota hyödynnetään läpi projektin.

2.3 Riskitekijät ohjelmistokehitysprojekteissa

Projekti on epäonnistunut, mikäli se on ylittänyt suunnitelmansa jossakin oleellisessa vaiheessa, esimerkiksi budjetin tai aikataulun ylittyminen. Hyvin harvoin projektin epäonnistumisen syynä on tekninen osaamattomuus, vaan ennen kaikkea epäonnistumisen taustalla on joko projektin suunnitteluun, aikataulutukseen, viestintään tai projektin johtamiseen liittyvät tekijät. Voi myös olla, että projektin epäonnistumisen taustalla on monta ongelmaa samanaikaisesti. (Borgström, A 2015, 23-24.)

Projektinhallinnan mukana seuraa myös riskienhallinta, jotka tyypillisesti ovat projektin johtajan vastuulla. Sommervillen (2006, 104–105) riskit voidaan jakaa kolmeen kategoriaan: projektin riskeihin, tuoteriskeihin ja liiketoiminnan riskeihin. Projektiriskit vaikuttavat projektin aikatauluun tai resursseihin. Tuoteriskit vaikuttavat ohjelmiston laatuun tai suorituskykyyn. Liiketoiminnan riskit ovat ohjelmistoa kehittävään organisaatioon liittyviä riskejä. Yhdessä kategoriassa syntyvä riski voi vaikuttaa kaikkiin kategorioihin.

Yleisesti ottaen ohjelmistokehitysprojektien suurimmaksi riskitekijäksi koetaan tutkimuksen mukaan tehoton kommunikointi. Sen jälkeen yleisimpiä riskitekijöitä ovat sosiokulttuurilliset eroavaisuudet, aikaero sekä kielimuuri. Viidennen sijan jakaa epämotivoitunut tiimi sekä maantieteellinen etäisyys. Näiden riskitekijöiden tunnistaminen on avainasemassa onnistuneen ja tehokkaan ohjelmistokehityksen toteuttamiseksi. (Ghafoor, Shah & Rashid 2017.)

Ohjelmistokehitysprojekteissa, joissa tiimi sijaitsee maantieteellisesti hajautettuna, on havaittu suurimpina haasteina viestinnän, toiminnan koordinoinnin ja hallinnan ongelmat. Nämä haasteet ovat syntyneet suurelta osin nimenomaisesti etäisyyden seurauksena. Onkin todettu, että viestintä on ohjelmistokehityksen keskeinen prosessi ja tutkimusten mukaan on näyttöä, että se on vahvasti kytköksissä hallinnoinnin ja koordinoinnin tehokkuuteen. (Fernando, Hall & Fitzpatrick 2011, 131.)

Riskitekijänä ohjelmistokehitysprojekteissa nähdään usein myös puutteellinen tai liian laaja vaatimusmärittely. Usein näissä tilanteissa asiakkaan ja kehittäjien välille jää väärinymmärrys, joka johtaa erilaisiin ongelmiin projektin edetessä. (Sommerville 2006, 106.)

Vaatimusmäärittelyjen perusteella annetaan työmääräarvioita, joiden tekeminen on usein ongelmallista (Sommerville 2006, 107). Onkin tärkeä ymmärtää, miksi aika-arvioita tehdään ja kuinka arvio tehdään. Arvion tekeminen ei ole kaikissa tilanteissa tarpeellista tai

edes hyödyllistä, mutta usein aika-arviota tarvitaan liiketoiminnan johdon päätöksentekoon (ThoughtWorks 2013).

2.4 Tutkimusyrityksen ohjelmistokehitys

Käsittelemme tässä opinnäytetyössä suomalaisen ohjelmistokehitysyrityksen ohjelmistokehitystyötä, jota työstetään suomalaiselle startup-yritykselle. Palvelu on tuotantovaiheessa oleva web-pohjainen laskutus- ja kirjanpito-ohjelmisto, jota kehitetään jatkuvasti eteenpäin. Yrityksen tarjoama palvelu tarjotaan Software as a Service (SaaS) muodossa, eli ohjelmisto hankitaan palveluna lisenssien sijaan. Loppukäyttäjät maksavat palvelusta käytön ja käyttäjämäärän mukaisesti.

Haastattelussa selvisi, että kyseisessä asiakasprojektissa ei ole tällä hetkellä projektimuotoista lähestymistapaa, mutta kehitystä tulisi kuitenkin kehitystiimin vetäjän näkökulmasta toteuttaa projektimaisemmin. Varsinaista syytä projektimallin puuttumiselle ei tunnistettu haastatteluissa. Tälle kyseiselle kehitystyölle ei tällä hetkellä tunnisteta varsinaista päättymispäivää vaan kehitystyötä jatketaan toistaiseksi.

Tässä kyseisessä kehitystyössä tiimi koostuu asiakkaan puolelta asiakkaan edustajasta (toimitusjohtaja) ja toimittajan puolelta kehitystiimin vetäjästä, joka osallistuu myös ohjelmiston kehittämiseen sekä vaihtelevasta määrästä ohjelmistokehittäjiä. Tämän opinnäytetyön kirjoittamisen aikana projektissa työskenteli kaksi täysipäiväistä ohjelmistokehittäjää ja yksi osa-aikainen ohjelmistokehittäjä. Osa ohjelmistokehittäjien työstä ostetaan palveluna Intiasta.

Asiakkaan puolelta kehitystyöhön osallistuu myös edustajan lisäksi tarpeen mukaan substanssiosaamista omaavia henkilöitä, jotka eivät ole aikaisemmin olleet mukana ohjelmistokehitysprojekteissa.

Asiakkaan edustaja kertoo, että kehitystiimi osallistuu kaikkeen asiakasyrityksen toimintaan ja kehitystiimi on osana asiakkaan operatiivista toimintaa. Tämä on hieman poikkeuksellinen lähtökohta asiakkaan edustajan mielestä, mutta tämän ohjelmiston rakentamiseen se on ollut hyvä valinta. Tällaisella toimintamallilla on pyritty siihen, että kehitystiimin ja operatiivisen tiimin tavoitteet ovat samat ja ohjelmiston haluttu suunta on kaikille selkeä. Näin kehittäjät ovat myös osa tiimiä ja pääsevät osaksi asiakasyrityksen arkea.

2.5 Pohdintaa tapausyrityksen projektimallista

Tässä kyseisessä kehitystyössä ei pystytty tunnistamaan kaikkia kolmea projektin kulmakiveä eli sisältöä, aikataulua ja kustannuksia. Näistä asioista yksikään ei ole ollut täysin selkeä kehitystyön alkaessa. Mielestäni tässä on hyvin havaittavissa Juvosen (2018, 27.) kuvaama tilanne, jossa tilattua työtä ei tunnisteta projektiksi vaan ennemminkin työkokonaisuudeksi.

Sinänsä projektimaisuuden puuttuminen ei toistaiseksi ole vaikuttanut negatiivisesti ohjelmiston kehittämiseen, mutta kehitystyön jakaminen jatkuvaan palveluun ja pienempiin kehitysprojekteihin mahdollistaisi läpinäkyvyyden kehitystyöhön. Samalla se mahdollistaisi tarkemman henkilöresursoinnin. Kehityskokonaisuuksien pilkkominen projekteihin, joilla on sisältö, aikataulu- ja kustannusarvio toisi oletettavasti asiakasyritykselle merkittäviä kustannussäästöjä ja mahdollisuuden vaikuttaa selkeämmin ominaisuuksien aikataulottamiseen sekä projektin onnistumista voitaisiin mitata monilla erilaisilla mittareilla.

Kehitystyön kannalta työhön oli valittu mukaan olennaiset henkilöt, joiden avulla kehitystyön onnistumiselle oli hyvät edellytykset. Asiakkaan puolelta valituilla henkilöillä oli hyvin rajattu tai olematon kokemus ohjelmistokehitysprojekteista ja erityisesti ketteristä menetelmistä. Jotta työn onnistumisen edellytykset olisivat mahdollisimman hyvät, suosittelisin asiakkaan henkilöstöä kouluttamista ohjelmistokehityksen keskeisien asioiden ymmärtämiseen sekä mahdollisesti käytettävään ketterään menetelmään.

Haastatteluiden pohjalta oli havaittavissa, että riskienhallintaa projektissa ei ollut juurikaan pohdittu eikä sitä varten ollut suunniteltu toimintamallia. Riskienhallinta projektissa toteutettiin käytännössä siten, että kun ongelma tuli vastaan, mietittiin toimintatapa ja ratkaistiin ongelma. Pidemmän päälle olisi syytä toteuttaa riskienhallintasuunnitelma.

3 Ketterät ohjelmistokehitysmenetelmät

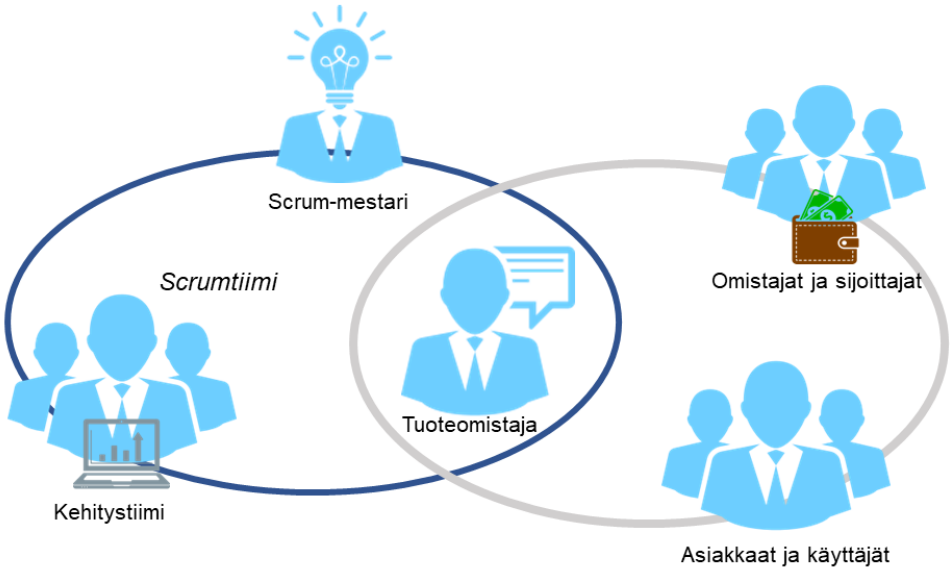
Lähtökohta ketterille menetelmille löytyy vuonna 2001 tehdystä ketterän ohjelmistokehityksen julistuksesta. Pohjana julistukselle on ajatus yhteisestä tekemisestä ja muiden auttaminen. Julistuksen mukaan ensisijaisesti tulisi keskittyä kanssakäymiseen ja yksilöihin menetelmien ja työkalujen sijaan sekä tekemään toimivaa ohjelmisto kattavan dokumentaation sijaan. Lisäksi tulisi enemmin olla valmis vastaamaan muutokseen, kun pitäytymään suunnitelmassa sekä tehdä asiakasyhteistyötä sopimusneuvottelujen sijaan. Julistuksen takana olevissa periaatteissa korostetaan muun muassa asiakkaan ja ohjelmistokehittäjien yhteistä työskentelyä sekä kasvokkain käytävää keskustelua. (Agile Manifesto, 2001.)

Ketterän ohjelmistokehityksen julistuksen myötä on syntynyt useita ketteriä menetelmiä ja niihin liittyviä työkaluja. Ketterien menetelmien avulla ohjelmistokehitys pystyy vastaamaan nopeasti muuttuvaan liiketoimintaympäristöön. Erilaisia projektinhallinnan viitekehyksiä on useita, mutta kaikille viitekehyksille on yhteistä mukautuvuus ja pyrkimys tuotteen arvon lisäämiseen. Ketteriä menetelmiä ovat muun muassa Scrum, Extreme-ohjelmointi (XP) ja Kanban. Yhdistävä tekijä näiden menetelmien välillä on muutoksiin sopeutuvuus sekä yhteistyön ja viestinnän merkityksen korostaminen. (Matharu, Mishra, Singh & Upadhyay 2015, 5.)

3.1 Scrum-viitekehys projektinhallintamenetelmänä

Scrum-viitekehyksen työskentely tapahtuu kehitysjaksoissa, joita kutsutaan sprinteiksi. Sprinttien kesto on muutamasta viikosta maksimissaan kuukauteen. Jokaisessa sprintissä on sama prosessi – sprintti suunnitellaan, päivittäisen työn tukena pidetään päivittäisiä palavereita, sprintin tulokset katselmoidaan ja lopuksi pidetään retrospektiivi, joka toimii eräänlaisena palautekeskusteluna. Jokaisen sprintin aikana vaatimusmääritysten mukaiset ominaisuudet kehitetään, testataan, integroidaan olemassa olevaan järjestelmään ja hyväksytään. Tuotoksena sprintistä syntyy ennakkoon määritetyt, toimivat ja julkaisuvalmiit ominaisuudet. Uusi sprintti alkaa heti edellisen päätyttyä. Sprinttejä toistetaan niin kauan, kunnes projekti on valmis. Projektin alussa ei välttämättä ole tiedossa, kuinka monta sprinttiä vaaditaan lopputuotteen valmiiksi saattamiseen. Projektille voidaan kuitenkin asettaa budjetti- tai aikaraja, joka määrää projektin keston. Tällöin lopputuote saatetaan niin valmiiksi, kuin mahdollista, keskittyen tärkeimpiin ominaisuuksiin. (Juvonen 2018, 11; Schrabber & Sutherland 2014, 7; Layton 2015, 99-100.)

Scrum-projektissa tiimiin kuuluu yleensä kehitystiimi, scrum-mestari sekä tuoteomistaja, joka määrittelee tuotteen vaatimukset ja priorisoi tehtäviä yhdessä tiimin kanssa. Yleensä tuoteomistaja edustaa omistajatahon mielipiteitä, ja tuo esiin tuotteen kehitystoiveita, jotka voivat tulla myös asiakkailta tai käyttäjiltä (Layton 2015, 36). Tuoteomistajalta ei vaadita ohjelmistoalan kokemusta (Juvonen 2018, 12). Tiimin riippuvuuksia on kuvattu alla olevassa kuvassa.



Kuva 1. Tiimin riippuvuudet kuvamuodossa. (Rubin 2012)

Yhdessä tiimi päättää sprinttien tavoitteet ja toteutettavat tehtävät. Scrum-mestarin tärkein tehtävä on vastata siitä, että kehitystiimillä on työrauha ja työ etenee suunnitellusti sekä tarvittaessa kommunikoi kehitystiimin ja tuoteomistajan kanssa muutoksista. Scrum-mestarin tehtävänä on myöskin valvoa, että menetelmää noudatetaan oikein. (Juvonen 2018, 11; Layton 2015, 37.)

Käytännön tasolla scrumia hyödynnetään siten, että tiedossa olevien vaatimukset, tarpeet ja toiveet kirjataan tuotteen kehitysjonoon, eli backlogiin. Kehitysjono on järjestetty lista kaikista toiminnoista, jonka tiedetään olevan tarpeen tuotteessa. Listan jokainen kohta sisältää vähintään kuvauksen ja työmääräarvion. Prioriteettia määritetään listan järjestyksellä. Kehitysjono elää projektin edetessä ja täydentyy aina, kun tietoa tulee lisää. Tyypillisesti tuoteomistajan tehtävänä on huolehtia kehitysjonon ylläpitämisestä. (Schrabber & Sutherland 2014, 12.)

Scrumissa pidetään tyypillisesti neljän tyyppisiä palavereja, jotka ovat:

- päivittäinen palaveri (daily scrum)
- sprintin suunnittelu (sprint planning)
- sprintin katselmus (sprint review)

- retrospektiivi (retrospective)

(Schrabber & Sutherland 2014, 7-11.)

Sprintin suunnittelupalaverin aikana suunnitellaan kaikki kyseisen sprintin aikana toteutettava työ, joka tukee asetettua tavoitetta. Kestoltaan sprintin suunnittelupalaveri saisi olla maksimissaan kaksi tuntia yhtä sprintin työviikkoa kohden (Layton 2015, 101). Kehitysjonon tiketeistä kerätään tikettejä sprintin kehitysjonoon, josta tiketit menevät ohjelmistokehittäjien toteutettavaksi. Lisäksi tarkastetaan, että vaatimukset ovat määritetty valituille tiketeille. Sprintin kehitysjonolle annetaan tikettien perusteella kokonaistyömäärä, jonka toteutumista seurataan päivittäin (Schrabber & Sutherland 2014, 13). Tuoteomistajalta vaaditaan hyväksyntä suunnitellun sprintin sisällölle, ennen toteutuksen aloittamista (Juvonen 2018, 12).

Sprintin aikana tiimin jäsenet valitsevat sprintin kehitysjonosta tehtäviä, ja kun tehtävä on suoritettu, se merkitään valmiiksi ja siirrytään seuraavaan tehtävään. Yksittäisen tehtävän maksimikokona pidetään kahta työpäivää (Juvonen 2018, 12). Päivittäisen työskentelyn tukena pidetään noin 15 minuutin mittaisia päiväpalavereita kehitystiimin kesken, jonka aikana on tarkoitus selvittää mitä on tehty edeltävänä päivänä, mitä seuraavaksi tehdään ja onko työn toteuttamiselle esteitä. Palaveriin osallistuu kehitystiimin jäsenet ja scrum-mestari, joka pyrkii ratkaisemaan mahdolliset esteet työn tekemiselle. (Opelt, Gloger, Pfarl & Mittermayr 2013, 18; Juvonen 2018, 13.)

Kun sprintti saadaan valmiiksi, esitellään tuotokset tuoteomistajalle sekä omistajille ja mahdollisille muille kiinnostuneille sprintin katselmuksessa. Katselmuksen kesto on maksimissaan tunti sprintin viikkoa kohden. Katselmuksen tarkoituksena on, että tuoteomistaja kertaa sprintin tavoitteen ja kuinka tavoite saavutettiin sekä summaa suoritetut kehitysjonon tiketit. Tässä tilaisuudessa tuoteomistaja saa organisaatiolta arvokasta palautetta ohjelmistokehityksen suunnasta ja sen oikeellisuudesta. Lisäksi kehitystiimillä on mahdollisuus kertoa toteutettujen ominaisuuksien sisällöstä. (Layton 2015, 121-122.)

Sprintin katselmuksen ja uuden sprintin suunnittelun välillä pidetään retrospektiivi eli jälkitarkastelu, jonka aikana tarkastellaan kriittisesti edeltävää sprinttiä ja parannetaan prosesseja seuraavaa varten. Jälkitarkastelun ohjeellinen pituus on 45 minuuttia. Tämän tilaisuuden tarkoituksena on antaa mahdollisuus koko tiimille osoittaa sprintin aikana hyvin sujuneet asiat sekä mahdolliset kehityskohdat tulevia sprinttejä silmällä pitäen. Mikäli kehityskohtia löytyy, tulee tehdä toimintasuunnitelma näiden kohtien korjaamiseen. (Layton 2015, 125-126.)

3.2 Extreme-ohjelmointi (XP)

Luonteenomaista Extreme-ohjelmoinnille on asiakkaan intensiivinen osallistuminen kehitysprosessiin sekä nopea reagoimiskyky lyhyiden iteraatioiden ja säännöllisyyden ansiosta. Asiakkaalle saadaan toimitettua pienissä erissä testattua ja toimiva ohjelmistoa. Menetelmä perustuu jatkuvaan asiakaspalautteeseen. Asiakkaan sitoutuminen tapahtuu projektin jatkuvalla seuraamisella, jolloin asiakas on tietoinen ohjelmistokehityksen edistymisestä ja pystyy puuttumaan mahdollisiin ongelmiin tai epäkohtiin hyvissä ajoin. (Matharu ym. 2015, 3.)

Extreme-ohjelmoinnissa vaatimusmäärittelyt esitetään käyttäjätarinoina, jotka asiakas kirjoittaa. Käyttäjätarinoiden käyttäminen vaatimusmäärittelyjen tekemisessä on hyvin laajalle levinnyt käytäntö ketterissä ympäristöissä (Garbajosa, Wang & Aguiar, 2018, 4). Käyttäjätarinoita tehdessä ei varauduta sellaisiin vaatimuksiin, jotka eivät ole tiedossa. Näin järjestelmä saadaan tietoisesti pidettyä tiiviinä. Käyttäjätarinoiden kirjoittamisen jälkeen kehitysryhmä lukee sekä antaa palautetta käyttäjätarinoista. Tämän palautteen perusteella niitä muokataan paremmiksi. Käyttäjätarinat pilkotaan kehittäjien toimesta pieniksi tehtäviksi, jotka asiakas järjestää tärkeysjärjestykseen. Käyttäjätarinoiden pohjalta tehdään ominaisuudesta yksinkertaisin mahdollinen toteutus, johon voidaan lisätä toimintoja, kun asiakas näkee sen tarpeelliseksi. Asiakkaan vastuulla on hyväksyä käyttäjätarinan perusteella toimitettu ominaisuus. (Garbajosa, Wang & Aguiar, 2018, 7).

Suunnittelupeliksi kutsutaan määrittelyä, jossa suunnitellaan iteraatio. Asiakas esittää tilaisuudessa vaatimuksia järjestelmälle ja samanaikaisesti priorisoi niitä. Kehitysryhmä antaa vaatimusten tietojen pohjalta työaika-arvion sekä kustannusarvion. Näiden tietojen valossa asiakas päättää, mitä ominaisuuksia seuraavassa iteraatiossa toteutetaan. Suunnittelupelin tarkoituksena on ainoastaan suunnitella seuraavan julkaisun sisältö, eikä tehdä pitkän ajan suunnitelmaa. (Beck & Gamma 2000, 86.)

Testitapausten kirjoittaminen tehdään kehittäjän toimesta jo ennen kuin riviäkään ohjelmakoodia on kirjoitettu. Testaaminen on kiinteä ja olennainen osa tässä testilähtöisessä kehittämistavassa. Tyypillisesti nämä testitapaukset integroidaan automatisoituun testausjärjestelmään, joiden onnistuminen määrittää ohjelmointivaiheissa etenemisen. Kun testit onnistuvat, voidaan siirtyä seuraavaan tapaukseen. Mikäli testit palauttavat virheen, korjataan virhe ja ajetaan testejä niin kauan, kunnes virheitä ei enää synny. (Matharu ym. 2015, 3.)

Muihin ketteriin kehitysmenetelmiin verrattuna tässä menetelmässä ainutlaatuista on pariohjelmointi, missä kehittäjät toimivat pareina. Tämä toimintamalli parantaa viestintää ja vähentää työaikaa- ja taakkaa. Pariohjelmoinnissa toinen kirjoittaa ohjelmakoodi toisen seuratessa ja havainnoidessa mahdollisia virheitä koodissa tai logiikassa. Rooleja vaihdetaan aina säännöllisesti. Kehittäjien välillä on jatkuva keskustelu esimerkiksi toteutusvaihtoehdoista. Pariohjelmointi mahdollistaa sen, että kuka tahansa kehittäjistä pystyy ainakin teoreettisella tasolla muokkaamaan mitä tahansa ohjelmiston osaa. Sen vuoksi tulee olla koodistandardi, jossa määritellään ohjelmointityyliä visuaalisesti sekä käytettävien muuttujien, metodien ja luokkien nimeämistyyli. Nämä koodistandardit sovitaan esimerkiksi projektikohtaisesti. (Matharu ym. 2015, 3.)

3.3 Kanban

Kanbanin tarkoituksena on visualisoida tehtävät näkyviksi ja vähentää työn alla olevien tehtävien määrää (Matharu ym. 2015, 3). Näillä toimilla saadaan toimitettua asiakkaalle jatkuvasti uusia ominaisuuksia, ja asiakas saa lisäarvoa (Leopold & Kaltenecker 2015, 18). Tiimi keskenään päättää seuraavat työn alle otettavat tehtävät, eikä asiakkaalla ole vaikutusmahdollisuutta tehtävän valinnassa. Olennaista on, että kaikki tehtävälistan tehtävät ovat sellaisia, että ne tuottavat asiakkaalle lisäarvoa. Tällä eliminoidaan ylituotanto ja vähennetään hukkaan mennyttä työtä ja aikaa. (Matharu ym. 2015, 3.)

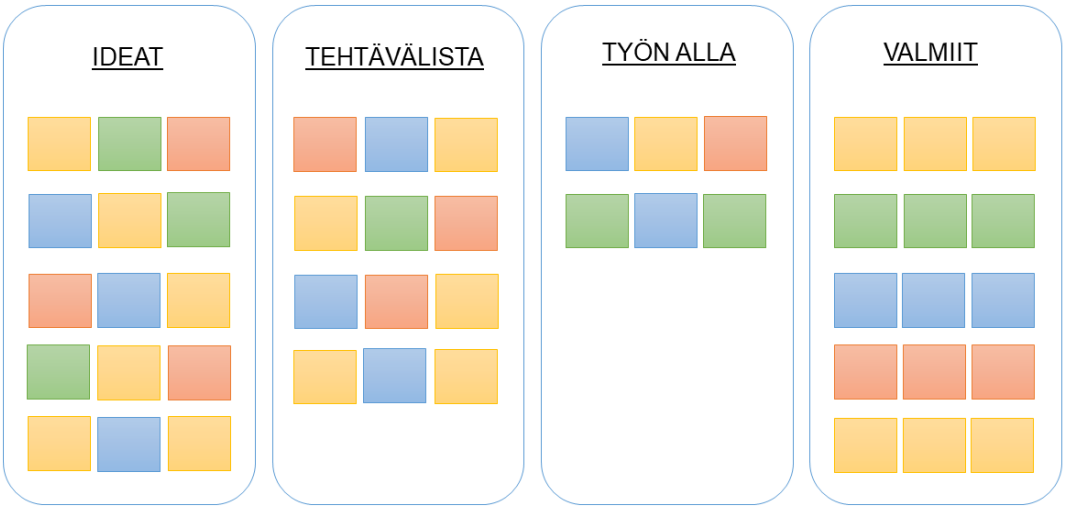
Kanban-prosessimallin keskeisten käytäntöjen tulkinta vaihtelee lähteen mukaan. Yleisimmin tunnistetaan seuraavat viisi käytäntöä:

1. Visualisoi työn kulku
2. Rajoita työn alla olevien tehtävien määrää
3. Tee työn kulku sujuvaksi
4. Luo selkeät prosessit
5. Paranna yhteistyömenetelmiä ja malleja. (Cole & Scotcher 2015, 71-72.)

Leopold ym. mukaan Kanbaniin kuuluu vielä yksi käytäntö, joka on palautteenantamiskäytännöt (Leopold & Kaltenecker 2015, 18).

Ensimmäisenä käytäntönä tulee visualisoida työn kulku. Visualisoinnin voi toteuttaa fyysisellä taululla, joka pystytetään työskentelytilaan tai ohjelmistolla. Visualisointi tehdään työn vaiheen mukaisesti alkaen tehtävistä, jotka tulisi tehdä, päättyen tehtäviin, jotka ovat tehty. Useimmilla on tässä välissä vain yksi välivaihe ja se on työn alla olevat tehtävät. Jotkut suosivat työn alla olevien tehtävien jakamista pienempiin palasiin, kuten esimerkiksi suunnitteluun, kehittämiseen, testaamiseen ja käyttöönottoon. Usein tauluun laitetaan myös sarake ideoille, joiden toteuttamista ei ole syystä tai toisesta päätetty. Taulun tulisi mahtua sellaiseen tilaan, että sen pystyy näkemään kokonaisuudessaan kerralla. Kuvassa 2 on esimerkki Kanban-taulusta, joka sisältää neljä saraketta: ideat, tehtävälista,

työn alla olevat tehtävät ja valmiit. Sarakkeissa olevat värikkäät laput kuvaavat yksittäistä tehtävää. (Cole & Scotcher 2015, 71-74.)



Kuva 2. Esimerkki Kanban-taulusta.

Tehtävälistalta tulisi löytyä ainoastaan sellaisia tehtäviä, jotka on ajateltu loppuun asti ja joiden tekemisen aloittamiseen tarvitaan ainoastaan päätös siitä, kuka tekee työn ja milloin työn tekeminen aloitetaan. Työn alla oleviin tehtäviin kuuluu tehtävät, jotka on määritetty tietylle tekijälle tai tekijöille ja sen tekeminen edistyy aktiivisesti. Valmiisiin tehtäviin kuuluvat tehtävät, jotka ovat kokonaisuudessaan valmiit. (Cole & Scotcher 2015, 76.)

Toisena käytäntönä on rajoittaa työn alla olevien tehtävien määrää, sillä liian monen asian samanaikainen tekeminen ei tuo parempaa lopputulosta. Työmääräraja suhteutetaan tiimin jäsenten lukumäärään, mutta siihen ei ole olemassa yksiselitteistä laskukaavaa. Hyvin tyypillisesti raja on kolme tehtävää yksilöä kohden tai tiimin koko kerrottuna kahdella. Työmääräraja kuvaa maksimimäärää työn alla oleville tehtäville yksilöä kohden. Jos raja on asetettu kolmeen tehtävään, ei työn alla olevissa tehtävissä saa olla yhdellä henkilöllä kolmea tehtävää enempää. Mikäli uusi tehtävä halutaan saada työn alle, tulee jonkin tehtävän siirtyä seuraavaan sarakkeeseen. (Cole & Scotcher 2015, 82.)

Kolmantena käytäntönä on saada työnkulku mahdollisimman nopeaksi ja sujuvaksi aina tehtävälistalta työn valmiiksi saattamiseen. Tällöin prosessi toimii optimaalisella tehokkuudella ja luo maksimaalista arvoa liiketoiminnalle nopealla aikavälillä. Tämän prosessin tulisi olla toistettava ja johdonmukainen. Usein prosessi hioutuu luonnollisesti, sillä työmäärärajoitus pakottaa tiimin perehtymään prosessin ongelmiin silloin, kun prosessissa on ongelmia ja korjaamaan ne (Cole & Scotcher 2015, 72-82).

Neljäntenä käytäntönä on luoda yksiselitteinen selvitys, kuinka työn toteutus tapahtuu, jotta prosessia voidaan arvioida puolueettomasti. Yhteisen ymmärryksen avulla on helpompi keskustella ongelmista puolueettomasti ja löytää yhteisymmärrys tilanteen kehittämiseen. On tärkeää, että jokaisella taulun sarakkeella on luonnollisen selkeät säännöt, joiden läpäisemisen jälkeen tehtävän voi siirtää seuraavaan sarakkeeseen. (Cole & Scotcher 2015, 72.)

Viidentenä käytäntönä on parantaa yhteistyömenetelmiä aina kun mahdollista. Työnkulun ollessa keskeisenä asiana tiimin päivittäisessä työssä, syntyy ideoita, kuinka työnkulkua voitaisiin parantaa entisestään. Työmäärärajan asettaminen korostaa prosessin ongelmia, jotka estävät sujuvan etenemisen. Kun ongelmat ovat tiedossa, on tiimin helppo löytää ratkaisut niihin ja yhteistyö paranee. (Cole & Scotcher 2015, 72.)

Mikäli ketterien menetelmien käyttö ei ole tuttua, on Kanbanin avulla helppo nähdä mitä ketterät menetelmät tarjoavat ja miten organisaatio suhtautuu ketterään menetelmään. Kanbania pystyy hyödyntämään yksittäinen tekijä tai kokonainen tiimi ja se toimii hyvin laajoissa projekteissa. (Cole & Scotcher 2015, 84, 69.) Kanban soveltuu hyvin ylläpitovaiheessa oleviin projekteihin (Juvonen 2018, 14).

3.4 Käytettävät menetelmät tutkimusyrityksessä

Tässä kehitystyössä ei ole projektimaista lähestymistapaa, ja ylipäätänsä käytössä olevat menetelmät ovat hajanaisia, eikä selkeitä toimintatapoja ole vielä löydetty. Syynä tähän on se, että asiakas on alkuvaiheessa oleva startup-yritys.

Sekä kehitystiimin vetäjän että asiakkaan edustajan haastatteluissa kävi ilmi, että tällainen hajanaisempi kehitysmalli sopii startup-yritykselle. Kehitystyötä tehdään tilanteen mukaisesti, ketteriä menetelmiä mukaillen noudattamatta kuitenkaan selkeästi mitään yksittäistä kehitysmallia. Startup-kulttuuriin kuuluu kokeilla, millainen toimintamalli tuo parhaita tuloksia, ja tässä tapauksessa erilaisten kokeiluiden pohjalta on tehty muutoksia ohjelmistoon ja koko ohjelmistokehityksen painopisteisiin. Asiakkaan edustaja näkee, että perinteiset ohjelmistokehitysmallit eivät olisi sopineet tähän projektiin, sillä alkuvaiheessa ohjelmiston vaatimukset eivät ole olleet selkeästi tiedossa. Näin ollen on ollut hyvä, että määrittelyjä on voitu tehdä sen mukaan, kun ymmärrys ohjelmiston tarpeista on lisääntynyt.

Asiakkaan edustaja kertoo, että ohjelmiston kehittämisessä on kokeiltu sprinttejä, mutta niistä on sittemmin luovuttu. Kehitystiimin vetäjä perustelee valintaa siten, että on helpompi tehdä ohjelmistoon uusia ominaisuuksia, kun aikatauluttaa isompia sprinttejä,

joissa on pienempiä paloja näistä ominaisuuksista. Lisäksi ominaisuuksien määrittelyt muuttuvat nopeasti, kun jotain isompaa osaa kokonaisuudesta ei ole osattu ajatella riittävän pitkälle suunnitteluvaiheessa. Sprinteistä luopumisen jälkeen kehitys on saatu joustavammaksi, kun ominaisuudet tehdään yhdessä tuoteomistajan kanssa yksi ominaisuus kerrallaan. Kehitystiimin vetäjä näkee, että sprintteihin palaaminen ei ole poissuljettua, mutta tällä hetkellä projekti etenee joustavammin eteenpäin valitulla strategialla.

Asiakkaan edustajan mukaan suunnitelmien ylläpito on ollut haastavaa ja ylipäätänsä sopivaa tapaa ylemmän tason suunnitelmien tekoon ei ole löytynyt, jonka vuoksi on kokeiltu monia eri tapoja. Ylemmän tason suunnitelmia on kokeiltu tehdä erilaisiin taulukoihin avulla. Tähän on kuitenkin panostettu huonosti ja suunnitelmien ylläpito on jäänyt, sillä ei ole löydetty toimivaa ja helposti ylläpidettävää ratkaisua. Ylemmän tason kehityssuunnitelmissa on ollut aikataulutuksia eri osakokonaisuuksille, ja nämä aikataulutukset on tehty jollakin tasolla yhteistyössä ohjelmistokehitystiimin kanssa. Aikataulut ovat nojanneet pitkälti kehitystiimin antamiin aika-arvioihin, sillä kehitystiimillä on asiakkaan mielestä paras osaaminen työmäärien arviointiin.

Tarkempia suunnitelmia on ylläpidetty projektinhallintatyökalu Jiran kehitysjonossa, mutta se on asiakkaan näkökulmasta hankala seurata. Jirassa tieto on paloiteltu liian pieniin osakokonaisuuksiin ja kokonaiskuvaa on tämän vuoksi vaikea seurata. Sieltä on asiakkaan edustajan mukaan vaikea löytää tietoa, joka olisi oikea-aikaista ja kiinnostavaa.

3.5 Tapausyrityksen ketterän menetelmän pohdinta

Kirjallisuuskatsauksen pohjalta perehdyin syvemmin kolmeen eri ketterään menetelmään, Scrumiin, Extreme-ohjelmointiin ja Kanbaniin. Kaikkiin kirjallisuuskatselmukseen valittuihin menetelmiin heijastui ketterän ohjelmistokehityksen julistuksen arvot ja periaatteet, joita ovat viestinnän tärkeys, asiakaslähtöisyys ja toimiva ohjelmisto. Vaikka menetelmät ovat luonteeltaan hyvin erityyppisiä, löytyy niistä yhteneväisyyksiä. Esimerkiksi kaikissa menetelmissä korostetaan viestinnän merkitystä suhteessa onnistuneeseen lopputulokseen.

Scum-menetelmän perusperiaatteet määrittelevät eniten esimerkiksi projektin jäsenten rooleja, kun taas Kanban eikä Extreme-ohjelmointi määrittele rooleja lainkaan. Kanban näkee projektin koordinoinnin koko tiimin tehtäväksi, kun taas scum-menetelmässä projektin koordinointiin on nimetty Scrum-mestari ja Extreme-ohjelmoinnissa ohjaaja.

Haastattelujen pohjalta ilmeni, ettei projektissa ole käytössä mitään tiettyä ketterää menetelmää. Huomasin kuitenkin, että projektin toimintatavoissa oli monia yhtäläisyyksiä näihin

kolmeen ketterään menetelmään, jotka tässä opinnäytetyössä on esitelty. Projektin toteutustapana oli aikaisemmin käytetty sprinttejä, jotka ovat selkeä Scrum-menetelmän ominaisuus. Näiltä ajoilta projektiin on varmasti jäänyt toimintatapoja Scrum-mallista, kuten esimerkiksi päivittäiset palaverit kehitystiimin kesken ja tapa kutsua asiakkaan edustajaa tuoteomistajaksi.

Oman kokemukseni ja haastatteluissa korostui esimerkkiprojektin merkittävät strategiset suunnanvaihdot ja sitä myöten määrittelyjen nopeat muutokset. Lisäksi haastattelujen perusteella korostui, että on tärkeää saada jatkuvasti uusia ominaisuuksia tuotantoon. Näin saadaan nähdä loppukäyttäjien reaktioita, jolloin ominaisuuksia pystytään jatkokehittämään oikeaan suuntaan. Näiden toiveiden ja kokemuksien perusteella scum ei ole menetelmänä ideaali tähän projektiin. Kanban ja extreme-ohjelmointi suhtautuu muutoksiin vielä joustavammin, ja lisäksi molemmat toteuttavat jatkuvan toimituksen periaatetta. Lisäksi haastattelujen aikana tuli ilmi, että asiakkaalle on hankalaa hahmottaa kehityksen tilaa.

Näiden tietojen valossa suosittelisin tähän projektiin ketteräksi menetelmäksi Kanbania, sillä se mahdollistaa projektin työnkulun visualisoinnin, rajaa työn alla olevien tehtävien määrää ja uusia ominaisuuksia saadaan toimitettua tuotantoon jatkuvasti. Toisaalta näkisin, että extreme-ohjelmointi voisi yhtä hyvin sopia tähän projektiin, sillä menetelmänä sisältää koodausstandardeja ja kehitystapa on hyvin testilähtöinen. Kuitenkin pohdintani tuloksena totean, että Kanbanin mahdollistama visualisointi tehtäville helpottaisi tässä projektissa asiakkaan näkymää projektin tilasta. Näin ollen asiakkaan kokonaiskuva kehityksen tilanteesta parantuisi ja kehittämistä pystyttäisiin toteuttamaan suunnitelmallisemmin.

4 Vaatimusmäärittely

Vaatimukset määrittelevät mitä ohjelmiston tulee tehdä. Vaatimuksilla kuvataan ohjelmiston varsinaiset palvelut tai toiminnot, joita järjestelmän tulisi tarjota sekä rajoitteet ohjelmiston toiminnalle. Vaatimukset kerätään erilaisilla tekniikoilla sidosryhmiltä heidän esittämiensä tarpeiden ja toiveiden täyttämiseksi. Vaatimusmäärittely toimii ikään kuin siltana loppukäyttäjien, asiakkaan ja muiden sidosryhmien tarpeiden ja tietotekniikan tarjoamien mahdollisuuksien välillä (Westfall 2006, 1; Jin 2018.)

Perinteisemmässä lähestymistavassa vaatimusmäärittely tehdään pääsääntöisesti projektin alussa, ja ne dokumentoidaan. Näihin dokumentoituihin vaatimuksiin ei haluta tehtävän muutoksia. Perinteinen vaatimusmäärittely pyrkii löytämään kaikki vaatimukset ennen kehitystyön aloittamista. Sen sijaan ketterässä projektissa vaatimusmäärittely on jatkuva prosessi, ja vaatimukset muuttuvat projektin edetessä. Ketterässä vaatimusmäärittelyssä jatkuva kommunikointi asiakkaan ja kehittäjien välillä on tärkeässä roolissa. (Ramesh, Cao & Baskerville 2010, 451; Jin 2018.)

4.1 Ketterä vaatimusmäärittely

Ketterässä kehityksessä vaatimukset eivät ole ennalta määritettyjä vaan ne syntyvät kehitysprosessin aikana. Aivan projektin alkuvaiheessa tehdään ylemmän tason vaatimusanalyysi hankkeesta. Tällä kehitystiimi hankkii ylemmän tason käsityksen sovelluksen kriittisistä ominaisuuksista yksityiskohtaisten vaatimusten sijasta. Näiden ylemmän tason vaatimusten ei ole tarkoitus olla täydellisiä eikä tarkoitus ole myöskään kattaa koko ohjelmistoa. Sen sijaan ne toimivat lähtökohtana suunnittelulle. Kun tuotteesta tai palvelusta tiedetään enemmän, lisätään lisää ominaisuuksia ja käyttäjätarinoita. (Ramesh ym. 2010, 457-458.)

Ketterä vaatimusten määrittely jatkuu jokaisessa kehityskierroksessa. Jokaisen sprintin alussa asiakas käy kehitystiimin kanssa läpi yksityiskohtaisesti toteutettavat ominaisuudet. Usein vaatimusanalyysissä käydään läpi myös tarkempi käyttöliittymämuotoilu. Keskustelun lopputuloksena syntyy tarkasti määritetyt vaatimukset, alustava suunnitelma ja joskus myös tarkempi toteutussuunnitelma. (Ramesh ym. 2010, 457-458.)

Vaatimusmäärittelyjen jälkeen priorisointi on välttämätöntä, etenkin kun hankkeella on rajallinen määrä budjettia, henkilöitä tai aikaa. Vaatimukset voidaan määritellä olennaisiksi, hyödyllisiksi ja toivotuiksi ominaisuuksiksi. Kehittämisen aikana asiakkaan sekä kehittäjän

ymmärrys hankkeesta parantuu ja uusia vaatimuksia lisätään tai olemassa olevia vaatimuksia muutetaan. Jotta prioriteetit pysyisivät ajan tasalla, priorisointi toistuu koko kehitysprosessin ajan. (Ramesh ym. 2010, 458-459.)

4.2 Vaatimusmäärittelyjen dokumentointi ketterässä ohjelmistokehityksessä

Ketterän ohjelmistokehityksen vaatimusmäärittelyt ovat usein vapaamuotoisia käyttäjätarinoita sekä testitapauksia, joita priorisoidaan. Useimmissa menetelmissä vaatimukset määritellään asiakkaan kanssa yhdessä, jotta vaatimukset tunnistetaan ja priorisoidaan. On tärkeää, että vaatimus sisältää kuvauksen halutusta toiminnasta ja mahdollisista rajoitteista. Asiakas ei välttämättä osaa itse jäsentää vaatimuksia riittävän hyvin, joten on tärkeää, että ohjelmistokehittäjä pystyy jäsentämään vaatimukset selkeiksi esimerkeiksi, kuten prototyypeiksi. Kaikki vaatimukset kerätään kehitysjonoon tiketeiksi, johon aina lisätään uusia tikettejä, jotka sisältävät määrittelyn. Tarpeen vaatiessa vaihdetaan järjestystä prioriteettitilanteen muuttuessa. (Ramesh ym. 2010, 451.)

Ketterässä ohjelmistokehityksessä vaatimusmäärittelyjen dokumentointi on korvattu pääsääntöisesti asiakkaan ja kehitystiimin välisillä keskusteluilla. Dokumentoinnin tukena käytetään vapaamuotoisia käyttäjätarinoita, jotka määrittelevät korkeatasoiset vaatimukset. Käyttäjätarinat ovat usein lyhyitä ja melko abstrakteja kuvauksia, jotka auttavat tarkemmissa keskusteluissa. Kehitystiimin on helppo hyödyntää käyttäjätarinoita suuntaa-antavissa työaika-arvioissa ja kustannusarvioissa. Tarkemmat määrittelyt keskustellaan asiakkaan kanssa ennen toteutuksen aloittamista ja sen aikana. Tämän käytännön tehokkuus riippuu hyvin voimakkaasti asiakkaan ja kehitystiimin välisestä kommunikaatiosta. (Ramesh ym. 2010, 455-457.)

Yksi nopeimmista tavoista täyttää vaatimukset, on tehdä priorisoitu ominaisuuksien luettelo. Sen sijaan, että käytettäisiin muodollisia vaatimuksia, moniin hankkeisiin käytetään prototyyppiä, jotta asiakkaan kanssa kommunikointi on helpompaa. Prototyyppi lisää selkeyttää ja lisää asiakkaan ymmärrystä halutun ominaisuuden lopputuloksesta. Prototyyppien avulla voidaan esittää todellisen ohjelmiston toimintoja, käyttäytymistä ja validoida vaatimuksia. Tämä mahdollistaa sen, että uusia vaatimuksia voidaan lisätä jo kehitettyyn toiminnallisuuteen. Uusi versio edellisestä prototyypistä tai toimivasta ohjelmasta lisää asiakkaan käsitystä lopullisesta ohjelmistosta ja mahdollistaa henkilökohtaisten mielipiteiden ja näkemysten esittämisen ohjelmasta. (Ramesh ym., 457-458.)

Kun prototyyppiä käydään läpi asiakkaan kanssa, saadaan palautetta, jonka avulla vaatimukset saadaan validoitua ja tarkennettua riittävälle tasolle. Myös jo tehdyn ohjelmiston

toimintoa voi käyttää prototyyppinä, jolloin toimintoa muokataan vaadittujen ominaisuuksien kokeilemista varten. Mikäli ominaisuudella on kiire markkinoille, saadaan prototyypin avulla luotua yksinkertainen versio, joka voidaan siirtää nopeasti tuotantoon. (Ramesh ym. 2010, 460-461.)

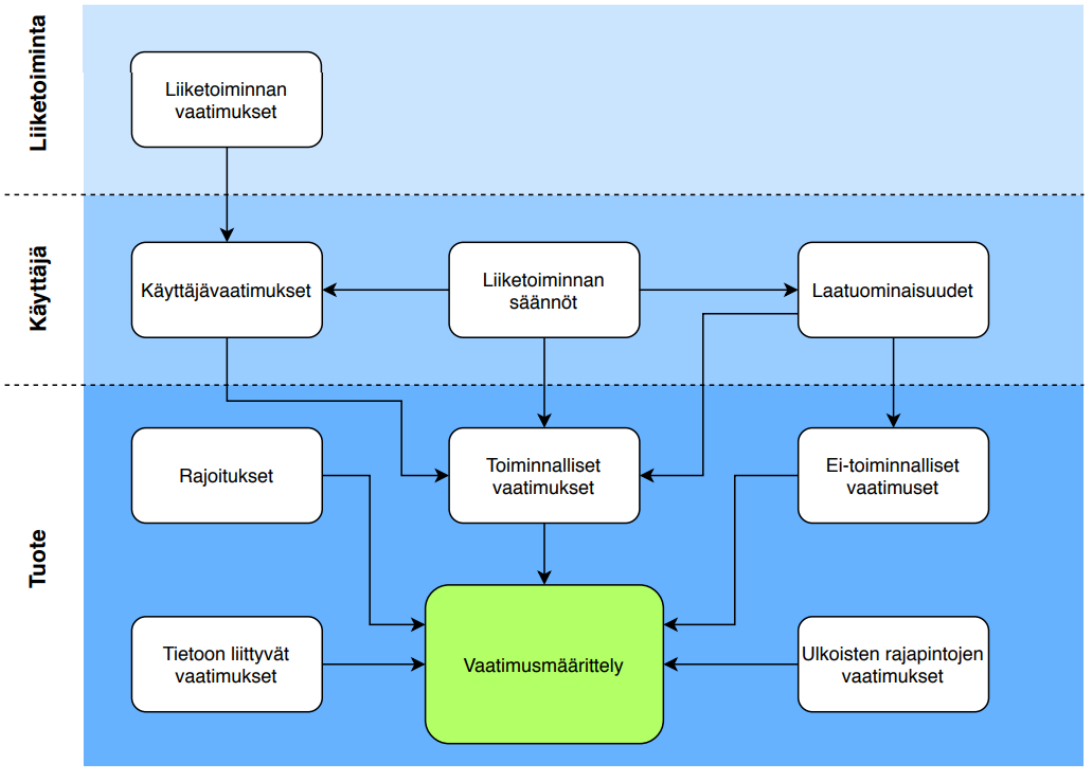
4.3 Asiakkaan rooli vaatimusmäärittelyissä

Kun ohjelmiston määrityksiä tehdään ketterästi, riippuu käytännön tehokkuus voimakkaasti asiakkaan ja kehitystiimin välisestä kommunikaatiosta. Asiakkaan tulee olla yhteistyökykyinen, tietoinen kehitettävästä ominaisuudesta ja ennen kaikkea käytettävissä, jotta hän pystyy tarjoamaan kehittämiselle tarvittavat vaatimukset. Mikäli asiakas ei pysty olemaan kehittämisessä vahvasti mukana, tällä lähestymistavalla on suuria riskejä, kuten lopputuloksena syntyvät riittämättömät tai väärät vaatimukset. (Ramesh ym. 2010, 455-457.)

Yleensä ketterän vaatimusmäärittelyn toimiva toteuttaminen edellyttää sen, että asiakkaat ja kehittäjät työskentelevät lähekkäin, mielellään samoissa tiloissa. Mikäli kehittäjät ja asiakas ei työskentele lähekkäin, voidaan vaihtoehtoisesti pitää päivittäin lyhyt palaveri tiimin ja asiakkaan välillä. Tällä tavoin saadaan lisättyä vuorovaikutusta asiakkaan kanssa. Palaveriin voi osallistua kaikki tiimin jäsenet tai joskus vain projektipäällikkö, tekninen johtaja tai kehittäjät, joilla on kysymyksiä asiakkaalle. (Ramesh ym. 2010, 457.)

4.4 Vaatimusmäärittelyjen tasot ja tyypit

Vaatimuksia on eri tasoisia ja eri tyyppisiä. Alla olevassa kuvassa on määritelty Westfallin (2006, 2-3) dokumentin mukaisesti, millaista tietoa on kerättävä, määriteltävä ja analysoitava, kun ohjelmistolle tehdään vaatimusmäärittelyjä.



Kuva 3. Vaatimusten tasot ja tyypit. (mukaillen Westfall 2006, 2.)

Liiketoiminnan vaatimukset kuvaavat miksi ohjelmistoa kehitetään ja määrittelevät ongelman, jonka ohjelmisto ratkaisee. Taustalta löytyy useimmiten asiakkaan tai organisaation havaitsema liiketoiminnan tavoite, joka vaatii ohjelmiston kehittämistä. Tyypillisesti yhden liiketoiminnallisen vaatimuksen täyttämiseen tarvitaan useita käyttäjätason vaatimuksia. (Westfall 2006, 1-2.)

Käyttäjävaatimukset kuvaavat ohjelmiston toimintoja käyttäjän näkökulmasta. Nämä vaatimukset kuvaavat, mitä ohjelmiston on toteutettava täyttääkseen liiketoiminnallisen vaatimuksen. Esimerkiksi liiketoiminnallinen vaatimus voisi olla mahdollistaa ohjelmiston kuukausimaksujen maksu. Käyttäjävaatimuksia olisi mahdollistaa turvallinen maksutapa ja kuitin saaminen maksutapahtumasta. (Westfall 2006, 2.)

Tuotevaatimukset kuvaavat toiminnallisia vaatimuksia, joita ohjelmiston toimintoihin on rakennettava, jotta käyttäjä voi suorittaa tietyn toiminnon ja näin ollen täyttää liiketoiminnallisen tarpeen. Käyttäjätason vaatimuksen täyttämiseen voidaan tarvita useita toiminnallisen tason vaatimuksia. Maksutapahtuman ollessa kyseessä tämän tason vaatimuksia olisi esimerkiksi havaita syötetyt kortin tiedot ja annettujen tietojen validointi. (Westfall 2006, 2; Jin 2018.)

Liiketoiminnan säännöt ovat vaatimuksia, jotka syntyvät tietyistä käytännöistä, standardeista, säännöistä, lakipykälistä ja määräyksistä, jotka määrittävät kuinka käyttäjän tulee toimia liiketoiminnallisesti. Tämän vuoksi nämä määritykset luetaan käyttäjätason vaatimuksiksi. (Westfall 2006, 2.)

Käyttäjätason laatuominaisuudet ovat ei-toiminnallisia vaatimuksia, jotka kuvaavat ohjelmiston laatua. Näitä ominaisuuksia ovat luotettavuus, saatavuus, turvallisuus, ylläpidettävyys, siirrettävyys, käytettävyys ja muut ominaisuudet. Laatuominaisuus voi myös olla tuotetason toiminnallinen vaatimus, joka määrittelee mitkä toiminnot tulee olla olemassa ei-toiminnallisen ominaisuuden täyttämiseksi. Se voi myös yhtä hyvin kääntyä tuotetason ei-toiminnalliseksi vaatimukseksi, joka määrittelee ominaisuudet, jotka ohjelmistolla on oltava ominaisuuden täyttämiseksi. (Westfall 2006, 2-3.)

Toiminnallisille vaatimuksille määritetään rajoitukset ja reunaehdot ei-toiminnallisten vaatimuksien avulla. Ei-toiminnalliset vaatimukset kertovat, mitä ehtoja ohjelmiston on täytettävä, jotta toiminnalliset vaatimukset voidaan toteuttaa. Tällaisia ehtoja ovat esimerkiksi ohjelmiston koko, nopeus, käytettävyys, sitkeys, siirrettävyys ja luotettavuus. Useimmiten ei-toiminnallinen vaatimus esitetään täsmällisenä lukuarvona, kuten esimerkiksi ohjelmiston on kestettävä 25 tuhannen saman aikaisen käyttäjän kuormitus. (Westfall 2006, 1-3; Jin 2018.)

Toiminnalliset vaatimukset kuvaavat millaisia tehtäviä käyttäjien tulee pystyä tekemään toteuttaakseen liiketoiminnan vaatimukset. Lisäksi ne määrittelevät kuinka ohjelmisto reagoi annetuissa tilanteissa ja miten käyttäjän syötteet käsitellään. Toiminnallinen vaatimus voi myös kertoa mitä ohjelmiston ei tule tehdä. (Westfall 2006, 1-3.)

Rajoitukset määrittävät mahdolliset rajoitukset, joita ohjelmiston toimittaja voi tehdä ohjelmiston suunnittelussa ja kehittämisessä. Nämä määritykset liittyvät esimerkiksi järjestelmän suorituskykyyn. (Westfall 2006, 3.)

Ulkoisten rajapintojen vaatimukset määrittävät tietovirran vaatimukset yhteisten rajapintojen kautta laitteille, käyttäjille ja muille ohjelmistosovelluksille kehitettävän ohjelmistotuotteen ulkopuolella. (Westfall 2006, 3.)

Tietoon liittyvät vaatimukset määrittävät tietyt tietotyypit tai tietorakenteet, joita tarvitaan kehitettävään ohjelmistoon. Tällainen vaatimus voisi olla esimerkiksi ohjelmiston kautta maksettujen tilausten kuitit halutulta aikaväliltä. (Westfall 2006, 3.)

4.5 Tutkimusyrityksen vaatimusmäärittelyt

Esimerkkitapauksessa ohjelmiston määrittelyjä toteuttaa asiakkaan toimitusjohtaja, jolla ei ole aikaisempaa merkittävää kokemusta vastaavan laajuisesta ohjelmistokehityksestä. Määrittelyjä tulee lisäksi asiakasyrityksen omistajilta sekä yksittäisiltä työntekijöiltä, joilla on substanssiosaamista määrittelyn kohteena olevaan ominaisuuteen, mutta ei varsinaista osaamista ohjelmistokehityksestä. Määrittelyt ovat usein eri tasoisia ja projektin vetäminen asiakasyrityksen puolelta on epäselvää ohjelmistokehitystiimin näkökulmasta.

Kehitysideat päätyvät yleensä asiakkaan edustajan työpöydälle. Tässä vaiheessa asiakkaan edustaja pohtii, onko idea potentiaalinen ja kannattaako toimintoa lähteä kehittämään eteenpäin. Yleensä kehitystiimin vetäjä otetaan jo tässä vaiheessa mukaan keskusteluun, jotta saadaan aikaiseksi avoin keskustelu ja yhdessä päätetään, lähdetäänkö toimintoa kehittämään. Jo tässä vaiheessa kehitystiimiltä tulee alustava aika- ja resurssiarvio toiminnon toteuttamisesta ja päätetään kuinka nopealla aikataululla toimintoa, lähdetään viemään kehitysputkessa eteenpäin.

Kun toiminnon kehittämisestä on päästy yhteisymmärrykseen, luodaan projektinhallintasovellus Jirassa ylläpidettävään kehitysjonoon tiketti, ja sen kehitys tulee vastaan ennemmin tai myöhemmin, riippuen asiakkaan edustajan kanssa sovitusta aikataulusta. Isoista ja selkeistä kokonaisuuksista tehdään työaika-arvio kehitystiimin toimesta, mutta pienemmät korjaukset ja muokkaukset tehdään ilman erillistä työaika-arviota.

Haastattelussa kävi ilmi, että jotkin kehitysideat ovat tulleet kehittäjille suoraa omistajilta, jolloin keskustelu toiminnon kehittämisen vaatimista aika- ja henkilöresursseista on jäänyt vähemmälle ja lisäksi vaatimusmäärittelyt ovat jääneet pintapuolisiksi. Asiakkaan edustaja ei kuitenkaan osaa haastattelussa ottaa kantaa siihen, että onko lopputuloksessa näkynyt vaikutuksia poikkeavasta toimintamallista.

Kehitystyön etenemisestä ei tällä hetkellä seurata varsinaisilla mittareilla. Työn edistymistä seurataan lähinnä tavoitteiden mukaisesti eli mitä on sovittu kehitettäväksi ja mitä on saatu aikaiseksi. Varsinaista ohjelmiston koodia tarkkaillaan laadullisesti kehitystiimin vetäjän toimesta päivittäin. Kehitystyön etenemisestä kerrotaan asiakkaalle viikkopalaverissa, ja tämän lisäksi satunnaisesti joko kehitystiimin vetäjän tai ohjelmistokehittäjän toimesta. Kehitystiimin vetäjän mielestä etenemisestä tulisi keskustella asiakkaan kanssa useammin ja monipuolisemmin.

Toteutetun kehitystyön testaamiseen osallistuu pääsääntöisesti toiminnallisuuden toteuttanut ohjelmistokehittäjä. Testaaminen toteutetaan asiakkaan toiveiden mukaisesti, mutta mitään yhteistä tapaa testitapausten kirjoittamiseen ei ole olemassa. Ohjelmistokehittäjä luo testitapaukset itse ja testaa lähinnä vain sen ominaisuuden, joka on toteutettu eikä ota huomioon mahdollisia riippuvuuksia vanhoihin ominaisuuksiin.

4.6 Pohdinta tapausyrityksen vaatimusmäärittelyistä

Haastatteluissa ei käy ilmi mitään selkeää kysymyspatteristoa, jonka avulla vaatimusmäärittelyjä tehtäisiin. En myöskään pystynyt havaitsemaan, että vaatimuksia luokiteltaisiin tai että asiakkaan edustaja tai kehitystiimi luokittelisi vaatimuksia esimerkiksi Westfallin esittämällä jaolla. Molemmissa haastatteluissa kävi kuitenkin ilmi, että projektin kokonaissuunnitelmallisuus on heikolla tasolla.

Vaatimusmäärittelyjä asiakkaan puolelta toteuttaa tässä projektissa ihmiset, joilla ei ole osaamista ohjelmistokehityksen parista ja ketterät menetelmät sekä vaatimusmäärittely on heille selkeästi vierasta. Tällaisessa tilanteessa olisi varmasti paikallaan kouluttaa asiakkaan henkilöstöä ymmärtämään ketterän ohjelmistokehityksen perusperiaatteita ja lisäksi voisi toteuttaa vaatimusmäärittelyn tueksi geneerisen tarkastuslistan. Tarkastuslistan avulla asiakas voisi käydä läpi, mitä ohjelmistokehitys yleisesti ottaen tarvitsee tietoonsa toiminnon kehittämistä varten.

Ohjelmistokehittäjän työn kannalta olisi tärkeää, että kehitysjonoon kirjattuihin tiketteihin olisi tehty selkeät vaatimusmäärittelyt jo ennen kuin tiketti päätyy päivän työlistalle. Tämä nopeuttaisi ohjelmointityön tekemistä ja vaatimusten perusteella tietäisi varmemmin, että tehty toiminto vastaa asiakkaan toivetta. Vaatimusmäärittelyjen puutteellisuus hidastaa ohjelmistokehittäjän työtä ja pahimmillaan aiheuttaa sen, ettei kehitetty ominaisuus vastaa asiakkaan tarpeeseen. Ominaisuuden korjaaminen voi vaatia merkittävän määrän lisätyötä ennen kuin ominaisuus voidaan hyväksyä tuotantoon vietäväksi asiakkaan hyväksynnän jälkeen.

Jotta toteutettujen toimintojen ulkoasu vastaisi asiakkaan näkemystä, olisi hyvä luoda graafinen ohjeistus koko tiimin käyttöön. Tällaisen ohjeistuksen avulla asiakas pystyisi kuvailemaan haluamaansa lopputulosta paremmin ja asiakkaan tyytyväisyys nousisi.

5 Kommunikointi ketterässä ohjelmistokehitysprojektissa

Viestintä on ihmisten välistä toimintaa eli vuorovaikutusta. Se on prosessi, jossa vaihdetaan sanomia ja tuotetaan merkitystä. Viestintä muodostuu sanallisesta ja sanattomasta viestinnästä. Sanatonta viestintää on ilmeet, eleet, katseet, asennot ja niin edelleen. Sanallista viestintää on sanoista rakentuva puhe sekä kirjoitus. Viestintä voi olla tiedostamatonta tai tiedostettua. Lukija arvioi viestintää kolmen asian perusteella: mikä väline on valittu viestin välitykseen, millä tavoilla puhuttelemme ja millainen ulkoasu tekstillä on. (Lohtaja & Kaihovirta-Rapo 2012, 11.)

Tehokkaalla viestinnällä voidaan saavuttaa avoin tiedon leviäminen, kokemusten jakaminen, uudenlaisia työtapoja ja tietysti projektin tulosten saavuttaminen. Lisäksi tehokas viestintä parantaa työssäjaksamista. (Häkkinen 2014, 14.)

Yksi suurimmista haasteista ketterissä ohjelmistokehitysprojekteissa on viestintä. Sitä pidetään tärkeänä elementtinä erityisesti projekteissa, joissa tiimi on maantieteellisesti hajautettu, jotta pystytään jakamaan tietämystä tiimin jäsenten välillä ja ymmärtämään asiakkaan vaatimuksia. (Kaur & Sharma, 2014, 41.)

5.1 Asiakaskommunikointi ketterässä ohjelmistokehitysprojektissa

Ketterät kehitysmenetelmät nojaavat aktiiviseen tiedon saantiin asiakkaalta, kun taas perinteisemmissä menetelmissä asiakaskommunikointia tehdään tarpeen mukaan (Korkala, Pikkarainen & Conboy 2010, 44). Ketterien menetelmien tavoitteena on parantaa viestintää niin kehitystiimin kuin asiakkaan suuntaan. Viestinnän rooli ketterässä kehityksessä on merkittävä. Jatkuva viestintä tiimin ja sidosryhmien välillä parantaa tiimityötä ja mahdollistaa tiimin välisen asioiden ja ratkaisujen jakamisen. Maantieteellisesti hajautetuissa tiimeissä viestintämenetelminä käytetään usein sähköposteja, pikaviestejä ja puhelinsoittoja. Samassa tilassa työskentelevät pystyvät myös keskustelemaan kasvotusten, jolloin viestintä on optimaalista. (Arora & Goel 2012, 394; Kaur & Sharma, 2014, 40; Korkala 2014, 70.)

Tyypillisesti projektipäällikkö on avainasemassa viestinnän kanssa. Projektipäällikön tehtävä on luoda yhteisymmärrys projektitiimin jäsenten ja asiakkaan välille sekä avata tarpeelliset kanavat, jotta viestintä on toimivaa. Projektipäällikön työn kannalta on tärkeää, että projektin jäsenet ovat sitoutuneita projektiin ja sitoutuminen syntyy kommunikoinnin kautta (Häkkinen 2014, 12). Ketterää menetelmää hyödyntävässä projektissa ei välttämättä ole erillistä projektipäällikköä, joten viestinnästä on pidettävä kollektiivista vastuuta.

Ketterät menetelmät auttavat parantamaan viestintää kehitystiimin välillä, mutta laajemman ympäristön viestintään (kuten asiakasviestintään) ne eivät tarjoa riittäviä työkaluja (Korkala 2014, 60). Ohjelmistokehityshankkeessa asiakkaan rooli on välttämätön, ja luo tiettyjä vaatimuksia asiakkaalle. Korkala (2014, 61) listaa muutamia asiakkaan velvollisuuksia ohjelmistokehitysprojekteissa ja niitä ovat seuraavat viisi:

1. Asiakkaan velvollisuutena on ymmärtää loppukäyttäjiä ja ylläpitää säännöllistä yhteyttä heihin, sekä tasapainottaa mahdolliset ristiriidat
2. Selkeyttää kehittäjille ominaisuuspyyntöjä ja ymmärtää teknisiä huolenaiheita, joita kehittäjät voivat kohdata
3. Määrittää toiminnalliset testit käyttäjätarinoille ja tarkistaa, että testit on suoritettu oikein
4. Osallistuminen kehitysjaksojen suunnitteluun ja julkaisuun
5. Ylläpitää hyvää yhteyttä johtoportaaseen, selvittää projektin edistymisen tila sekä perustella kehitystiimin kanssa vietetty aika.

Tänä päivänä tekniikka on mahdollistanut maantieteellisesti hajautetun ohjelmistokehityksen, josta syntyy erilaisia etuja. Tällaisella ohjelmistokehityksellä saadaan hyötyjä, kuten esimerkiksi laadukkaampaa ohjelmistoa, nopeampia innovaatioita ja tuottavuuden parantuminen. Lisäksi pystytään hyödyntämään työvoimaa, jolla on erikoisosaamista jonkin tietyn asian saralla sekä toisaalta myöskin ohjelmistokehityksen kustannuksia saadaan vähennettyä siirtämällä työn toteutus maihin, joissa on alhaisempi työvoimakustannus. Kansainvälisen ohjelmistokehityksen hyödyntäminen tarkoittaa tyypillisesti sitä, että tiimit ovat monikansallisia ja tulevat erilaisista kulttuureista. Lisäksi työskentely voi tapahtua eri aikavyöhykkeellä. Hyödyistä huolimatta maantieteellisesti hajautettu ohjelmistokehitys sisältää haasteita, jotka liittyvät viestintään, kulttuurien eroihin, valvonnan puutteeseen. Nämä haasteet syntyvät ajallisesta, maantieteellisestä ja kulttuurillisista eroista. (Kaur & Sharma, 2014, 39).

Kun tiimi on globaali, tarvitaan vahvaa koordinointia, sillä ketterissä menetelmissä tiimin jäsenet ovat riippuvaisia toistensa tekemisestä. Koordinointi parantavat päivittäiset palaverit, vapaamuotoinen kasvokkain käytävä viestintä ja kehitysjaksojen suunnitelupalaverit. Näiden kokousten avulla minimoidaan ja vähennetään koordinointiongelmia, jotka aiheutuvat ajallisista ja maantieteellisestä etäisyydestä. Kun koordinointiongelmat saadaan minimoitua, saadaan parempia ja nopeampia tuloksia ohjelmistokehityksessä. (Kaur & Sharma, 2014, 40).

Epämuodollista ja mieluiten kasvotusten käytävää viestintää on pidetty tehokkaimpana viestintätapana ketterissä projekteissa. Maantieteellisesti hajautetuissa kehitysprojekteissa kasvotusten käytävän keskustelun toteuttaminen voi olla äärimmäisen vaikeaa. Jotta ketterän kehitysmenetelmän vuorovaikutteista viestintää pystyttäisiin hyödyntämään,

ratkaisuksi on koettu oikeanlaiset viestintävälineet. Viestintävälineiden on oltava projektiin sopivat, ja on syytä pitää huolta, että kaikilla tiimin jäsenillä on asianmukaiset viestintävälineet esimerkiksi videoneuvotteluihin. Hyvillä ja toimivilla viestintäkanavilla saadaan aikaiseksi tarvittava tehokas kommunikointi. (Korkala 2014, 72; Kaur & Sharma, 2014, 41.)

Useita lähteitä tutkiessani huomasin kommunikaation nousevan keskeiseksi piirteeksi ketterässä ohjelmistokehityksessä. Kirjallisuuden perusteella ei kuitenkaan ole pystytty toteamaan, että ketterät menetelmät itsessään parantaisivat viestintää. On kuitenkin todettu, että viestintä on helpompaa tiimeille, jotka toimivat samoissa tiloissa sekä monessa lähteessä korostetaan erityisesti kasvokkaisen vuorovaikutuksen merkitystä. (Hummel, Rosenkranz & Holten 2013, 349.)

5.2 Hyvät käytännöt asiakaskommunikoinnissa ketterässä ohjelmistokehitysprojektissa

Viestinnän sisältöön vaikuttaa moni asia. Kommunikaatio vaatii yhteiset pelisäännöt ainakin palaverien ja aineistojen suhteen. On tärkeää miettiä, mitkä asiat ovat tärkeitä tai hyvä tietää nimenomaisessa yhteydessä sekä tunnistaa mahdolliset asiat, jotka voivat rasittaa toistuvaa viestintää. (Elo 2013, 21.)

Tutkimuksen mukaan virheiden ja väärinymmärrysten määrä kasvaa, mikäli projektin asiakaskommunikaation välineet eivät ole riittävän intensiivisiä ja informatiivisia, jonka vuoksi niiden merkitystä on syytä korostaa. Kun viestintä on intensiivistä, vähentää se väärinkäsitysten määrää, lisää työn tehokkuutta ja tarjoaa parempia oppimismahdollisuuksia. Ketterässä ohjelmistokehityksessä riski on kuitenkin pienempi, sillä iteraatiot ovat lyhyitä. (Korkala, Abrahamsson, Kyllönen 2006.)

Olisi suositeltavaa tehdä päätös viestintäkanavista projektikohtaisesti. Ketterän ohjelmistokehitysprojektin asiakaskommunikointiin on useita erilaisia kommunikaatiovälineitä, jotka toimivat hyvin. Kasvokkain käytävä kommunikointi on tyypillisin kommunikointiapa. Se on osoitettu tehokkaaksi, joten sitä tulisi hyödyntää aina kun on mahdollista. Seuraavaksi tehokkain viestinnän väline on videopuhelut. Sitä tulisi käyttää tilanteissa, joissa tarvitaan tehokasta asiakasviestintää ja palautetta, eikä asiakas ole samoissa tiloissa kasvokkain käytävää keskustelua varten. Videopuheluiden käytössä on havaittu niin negatiivisia kuin positiivisia puolia. Niissä on vaikea selventää ideoita ja usein päätöksen aikaansaamiseksi kuluu enemmän aikaa, verrattuna kasvokkain käytyyn keskusteluun. Toisaalta on todettu, että videopuheluissa tulee parempia päätöksiä, keskusteluissa on laajempia näkökulmia,

parempaa analyysia ja videopuhelu mahdollistaa avoimemman keskustelun. Äänipuheluilla ei pysty välittämään eleitä, mutta se mahdollistaa välittömän palautteen. Äänipuheluilla on hyödyllistä sopia esimerkiksi aikatauluista, mutta määrittelyihin tai tarkempaan keskusteluun se ei välttämättä ole toimiva väline. Kun keskusteltava aihe on tuttu, voidaan viestinnässä käyttää myös sähköpostia. (Korkala ym. 2010, 45.)

5.3 Tutkimusyrityksen kommunikointitavat

Asiakkaan koko tiimi, eli toimitusjohtaja, asiakaspalvelu ja markkinointi sekä kirjapitäjä istuvat samassa tiimissä yhden ohjelmistokehittäjän kanssa. Kehitystiimin vetäjä työskentelee pääsääntöisesti etänä toisessa kaupungissa Suomessa, mutta työskentelee satunnaisesti myös asiakkaan kanssa samoissa tiloissa. Lisäksi kehitystiimiin kuuluu offshore-tiimi, joka työskentelee Intiasta käsin.

Työskentelykielenä tässä projektissa käytetään suomea ja englantia. Pääsääntöisenä viestintäkanavana projektissa käytetään Slack-pikaviestintäsovellusta. Sen avulla pystyy soittamaan ääni- ja videopuheluita, jakamaan ruudun kaikille osanottajille sekä jakamaan tiedostoja sekä on mahdollista keskustella erilaisilla yksityisillä ja julkisilla kanavilla sekä yksityisesti yksittäisen käyttäjän kanssa.

Tässä projektissa Slackiin on yhdistetty projektin tehtävienhallintaohjelmisto Jira sekä Git-versionhallinnan graafinen käyttöliittymä Github. Lisäksi ohjelmistosta tulevat virheilmoitukset ohjataan myös Slackiin. Satunnaisesti viestintäkanavana käytetään myös Skype For Business -ohjelmistoa. Skypeä hyödynnetään pääasiassa kehitystiimin päivittäisiin palavereihin ja tilanteissa, kun ensisijainen viestintäkanava on syystä tai toisesta saavuttamattomissa.

Projektin dokumentteja säilytetään Dropboxissa sekä Jiran Confluensessa. Yhteisten pelisääntöjen puuttuessa on epäselvää, mihin sijaintiin dokumentit tulisi tallentaa ja mistä pitäisi etsiä tietynlaisia dokumentteja.

Tapauksen parissa työskentelevät kokivat, että viestintäkanavat ovat tällaisenaan riittävät, mutta niiden käyttämisessä on parantamisen varaa. Parantamisen varaa löytyy esimerkiksi yhtenäisistä toimintatavoista, joita ei varsinaisesti ole kirjattu mihinkään. Lisäksi epäselvyyksiä on esimerkiksi dokumenttien säilömispaikkaan ja jakeluun liittyvissä kysymyksissä.

Asiakkaan ja toimittajan välillä keskustelu käydään pääsääntöisesti suomeksi, mutta kehitystiimin päivittäisenä työkielenä on englanti. Ohjelmiston koodi, kommentit ja kaikki koodiin liittyvä dokumentaatio sekä kehitysjonon tiketit kirjoitetaan englanniksi. Työskentelyn tukena kehitystiimi pitää päivittäin englanninkielisen palaverin, jossa käydään läpi työnalla olevat tiketit sekä mahdolliset tekemisen esteet.

Asiakaspalaverit pidetään suomeksi. Asiakkaan kanssa ei kommunikoida säännöllisesti sovittujen palaverien puitteissa, vaan palavereita pidetään enemmänkin tarpeen mukaan. Asiakkaan edustaja on kuitenkin tavoitettavissa pikaviestintäsovelluksien välityksellä normaalin työpäivän aikana tarvittaessa. Sekä asiakkaan että toimittajan näkemyksen mukaan säännöllisistä, sovituista palavereista olisi hyötyä kehitystyön edistymisen kannalta. Asiakkaan edustaja kokee, että kasvotusten keskustelu on paras tapa asioiden hoitamiseen, mutta yhteisten pelisääntöjen puuttuessa sovitut asiat eivät välttämättä toteudu.

Asiakkaan edustajan haastattelussa kävi ilmi, että kehitystöiden etenemisestä ei ilmoiteta asiakkaalle säännöllisesti ja on nähtävissä, ettei toimittaja välttämättä koe tarpeelliseksi ilmoittaa etenemisestä kaikissa tapauksissa.

5.4 Pohdinta tapausyrityksen kommunikointitavoista

Tässä projektissa viestintäkanavana toimi hyvin yksiselitteisesti pikaviestintäsovellus Slack. Sen toimintavarmuus on hyvä ja se mahdollistaa hyvin laajan kommunikoinnin valitulle jakelulle. Haastattelussa kävi ilmi, ettei asiakkaan kanssa kommunikoida säännöllisesti sovittujen palaverien puitteissa vaan niitä järjestetään tarpeen mukaan. Sekä asiakkaan että toimittajan näkökulmasta tilanne ei ole ihanteellinen. Asiakkaan ja kehitystiimin vetäjän välillä olisi hyvä pitää säännöllinen palaveri esimerkiksi viikon loppupuolella. Sijoittaisin palaverin viikon loppupuolelle, sillä työviikon aikana edistetyt asiat ovat tuoreessa muistissa ja toisaalta seuraavaan viikkoon on helppo orientoitua.

Pikaviestintäsovelluksen lisäksi korostaisin kasvokkain käytävän keskustelun merkitystä, sillä sen merkitys on korostunut tämän opinnäytetyön tietoperustassa useamman kerran. Kasvokkain tapahtuvalle viestinnälle on kuitenkin sovittava tietynlaiset pelisäännöt, jotta niistä saadaan täysi hyöty irti. Tälle projektille voisi sopia ennakkoon tarkasti määritetty palaverin agenda (esimerkiksi tietty osakokonaisuus). Palaverista olisi hyvä kirjoittaa yksinkertainen muistio, jotta se on saatavilla jälkikäteen ja sovittujen asioiden etenemistä pystytään konkreettisesti seuraamaan seuraavassa palaverissa.

Projektiin liittyvien asiakirjojen hallintaa varten olisi hyvä sopia yksi paikka, johon sekä asiakkaalla että toimittajalla on pääsy. Asiakirjojen säilömistä olennaista on saavutettavuus, jakelu ja säilyvyys. Pilvipalvelut ovat hyviä paikkoja säilöä asioita, mutta asiakirjojen säilyttämisestä olisi hyvä olla olemassa kirjallinen ohje, jotta projektin parissa harvemmin työskentelevät löytäisivät asiakirjat helposti ja toimintatapa olisi yhtenäinen.

Asiakkaan edustajan haastattelussa kävi ilmi, että projektinhallintasovelluksen seuraaminen on työlästä, sillä tieto on paloiteltu liian pieniin osakokonaisuuksiin ja kokonaiskuvaa on vaikea seurata. Projektinhallintasovelluksesta on vaikea löytää tietoa, joka olisi oikea-aikaista ja kiinnostavaa. Onnistuneen kehitystyön edellytyksenä on asiakkaan pitäminen ajan tasalla projektin vaiheista. Tämän vuoksi on huolestuttavaa, että asiakas kokee, ettei projektinhallintasovelluksen tieto ole oikea-aikaista ja kiinnostavaa. Ratkaisuna tähän asiaan näkisin projektinhallintasovelluksen käyttökoulutuksen. Koulutuksen avulla asiakkaalle voitaisiin opettaa sovelluksen käyttöä ja sen aikana voitaisiin myös luoda omia näkymiä, josta asiakas saisi tarvitsemaansa tietoa aina tarvittaessa.

Haastatteluissa ei tullut ilmi, että kehitystiimin käyttämä terminologia olisi vaikeasti ymmärrettävää. Mikäli asiakkaan puolelle tulee uusia työntekijöitä, joille ohjelmistokehitys on aiheena vieras, on kuitenkin syytä pitää mielessä, ettei käytettävä terminologia ole itsestään selvää henkilölle, joka ei ole ollut tekemisissä ohjelmistokehitysalan kanssa.

Kaiken kaikkiaan projektin kommunikointiin olisi syytä tehdä yhteisiä pelisääntöjä myös riskienhallinnan kannalta. Tällä hetkellä kaikki projektin käytännöt ovat avainhenkilöiden päänsisäistä tietoa eikä varsinaisia ohjeistuksia ole olemassa. Tässä on olemassa merkittävä riski esimerkiksi tapauksessa, jossa projektiryhmän kokoonpano vaihtuisi äkillisesti merkittävästi ilman mahdollisuutta perehdyttämiseen.

6 Yhteenveto

Kuten tietoperustasta käy ilmi, pidetään kommunikaatiota tärkeänä osatekijänä onnistuneessa ohjelmistokehitysprojektissa. Tässä tapaustutkimuksessa asiakkaan ja toimittajan välisen kommunikaation haasteeksi muodostui suunnittelemattomuus. Asiakkaan ja toimittajan välillä ei lähtökohtaisesti pidetty yhteyttä säännöllisesti sovittujen palaverien tiimoilta, vaikka kehitystiimin välillä toimintamalli oli itsestään selvä ja toimiva.

Opinnäytetyön otanta koskee vain yhtä projektia ja siksi tämän hyödynnettävyys on rajattavissa. Tästä työstä on kuitenkin hyötyä sellaisenaan toimeksiannon antaneelle yritykselle. Lisäksi uskon, että mukaillen tästä on hyötyä myös muille ohjelmistokehitysalan yrityksille, joilla on haasteita asiakaskommunikaatiossa.

Työ toteutettiin tapaustutkimuksena, ja menetelmäksi valitsin haastattelut. Näin jälkikäteen voin todeta, että haastattelut olisi kannattanut toteuttaa heti opinnäytetyöprojektin alkuvaiheessa ennen tietoperustaan tutustumista. Näin ollen tietoperustan näkökulmaa olisi ollut selkeämpi. Haastatteluiden toteuttaminen venyi hieman ajateltua myöhemmäksi, sillä opinnäytetyö toteutettiin haastavaan loma-aikaan. Haastattelin kahta henkilöä, jotka olivat avainasemassa projektissa ja lisäksi toimin itse projektin kehitystiimin jäsenenä. Haastatteluun olisi voinut ottaa mukaan vielä yhden asiakkaan operatiivisissa tehtävissä työskentelevän henkilön, jotta näkemyksiä olisi tullut enemmän.

Pääosin pyrin käyttämään työssä lähteinä 2010-luvun teoksia. Yhdistettynä tämä rajaus ja työn aihe, osoittautui hyvien ja ajankohtaisten lähteiden löytäminen hankalaksi. Asiakkaan ja toimittajan välisen kommunikaation näkökulmasta kirjoitettuja teoksia, jotka käsittelevät aihetta nimenomaan ohjelmistokehityksen näkökulmasta, oli hyvin rajatusti saatavilla. Onnistuin kuitenkin pitämään lähteet sidottuna ohjelmistokehitykseen ja lopputuloksena löysin lähteitä riittävän kattavasti.

Suunnitelmaan verrattuna aikataulu venyi joitakin viikkoja ja siihen pääsyynä opinnäytetyön toteuttaminen kesäloma-aikana, jonka takia haastattelujen toteuttaminen venyi ja lisäksi työn toteutus tapahtui päivätyön ohessa. Päivätyön vaihteleva työkuorma vaikutti paikoitellen työn edistymiseen.

Lähteet

Agile Manifesto. 2001. Manifesto for Agile Software Development. Luettavissa: http://agilemanifesto.org. Luettu: 10.5.2018.

Bakalova, Z & Daneva, M. 2011. A comparative case study on clients participation in a 'traditional' and in an agile software company. Luettavissa: https://ris.utwente.nl/ws/files/5501401/Valoir_Camera_Ready.pdf. Luettu: 1.9.2018.

Cole, R & Scotcher, E. 2015. Brilliant Agile project management: a practical guide to using Agile, Scrum and Kanban. Pearson.

Elo, M. 2014. Viestintä IT-projekteissa: merkitys, hallinta ja esteet. Luettavissa: https://jyx.jyu.fi/dspace/bitstream/handle/123456789/44842/URN%3aNBN%3afi%3ajyu-201412103472.pdf?sequence=1. Luettu: 1.9.2018.

Fernando, B., Hall, T. & Fitzpatrick, A. 2011. The Impact of Media Selection on Stakeholder Communication in Agile Global Software Development: A Preliminary Industrial Case Study. ACM.

Garbajosa, J., Wang, X. & Aguiar, A. 2018. Agile Processes in Software Engineering and Extreme Programming. Springer International Publishing.

Ghafoor, F., Shah, I. & Rashid, N. 2017. Issues in Adopting Agile Methodologies in Global and Local Software Development: A Systematic Literature Review Protocol with Preliminary Results. Luettavissa: https://pdfs.semanticscholar.org/d43c/25df54d0d139f5ef2feaaa2879ca71f754a3.pdf. Luettu: 26.5.2018.

Hummel, M., Rosenkranz, C. & Holten, R. 2013. The role of communication in agile systems development. Luettavissa: https://doi.org/10.1007/s12599-013-0282-4. Luettu: 20.8.2018.

Häkkinen, T. 2014. Projektin kokoukset ja hyvä projektiviestintä projektin elinkaaren eri vaiheissa. Luettavissa: https://www.slideshare.net/tiinahakkinen148/projektiviestint-hkkinen. Luettu: 30.12.2017.

Jin, Z. 2018. Environment Modeling-Based Requirements Engineering for Software Intensive Systems. Morgan Kaufmann.

Juvonen, R. 2018. Ohjelmistoprojektin sudenkuopat ja miten ne vältetään. BoD - Books on Demand.

Korkala, M. 2014. Customer communication in distributed agile software development. Luettavissa: https://www.vtt.fi/inf/pdf/science/2015/S80.pdf. Luettu: 10.7.2018.

Korkala, M., Abrahamsson, P. & Kyllönen, P. 2006. A Case Study on the Impact of Customer Communication on Defects in Agile Software Development. Luettavissa: https://ieeexplore.ieee.org/document/1667565. Luettu: 20.8.2018.

Korkala, M., Pikkarainen, M. & Conboy, K. 2010. A case study of customer communication in globally distributed software product development. Luettavissa: https://dl.acm.org/citation.cfm?doid=1961258.1961269. Luettu: 20.8.2018.

Layton, M. 2015. Scrum for dummies. John Wiley & Sons.

Leopold, K & Kaltenecker, S. 2015. Kanban Change Leadership. Wiley.

Matharu, G., Mishra, A., Singh, H. & Upadhyay, P. 2015. Empirical Study of Agile Software Development Methodologies: A Comparative Analysis. Luettavissa: https://mycourses.aalto.fi/pluginfile.php/442685/mod_assign/intro/matharu.pdf. Luettu: 1.9.2018.

Murch, R. & Kosonen, J. 2002. IT-projektinhallinta. Edita Publishing Oy. Luettavissa: http://cna.mamk.fi/public/KosonenH/Projektinhallinta/IT_projektinhallinta/it-projektinhallinta_luku05.pdf. Luettu: 11.07.2018.

Mäntyneva, M. 2016. Hallittu projekti – Jäntevästä suunnittelusta menestykselliseen toteutukseen. Kauppakamari.

Opelt, A., Gloger, B., Pfarl, W. & Mittermavr, R. 2013. Agile Contracts: Creating and Managing Successful Projects with Scrum. Wiley.

Radigan, D. The secrets behind story points and agile estimation. Atlassian. Luettavissa: https://www.atlassian.com/agile/project-management/estimation. Luettu: 10.7.2018.

Ramesh, B., Cao, L. & Baskerville, R. 2010. Agile requirements engineering practices and challenges: an empirical study. Information Systems Journal.

Rubin, K. 2012. Product Owner Faces Two Directions. Agile Innovation Solutions Innolution LLC. Luettavissa: http://www.innolution.com/val/detail/product-owner-faces-two-directions. Luettu: 10.7.2018.

Schrabber, K. & Sutherland, J. 2014. Scrum Guide - Scrumin määritelmä ja pelisäännöt. Scrum.Org ja ScrumInc. Luettavissa: http://www.scrumguides.org/docs/scrum-guide/v1/Scrum-Guide-FI.pdf. Luettu 10.7.2018.

Sommerville, I. 2006. Software Engineering (Eighth edition). Luettavissa: https://doc.lagout.org/science/0_Computer%20Science/Software%20Engineering%2C%208th%20Edition.pdf. Luettu: 10.7.2018.

ThoughtWorks. 2013. Agile Project Management. Luettavissa: https://info.thoughtworks.com/rs/thoughtworks2/images/twebook-perspectives-estimation_1.pdf. Luettu: 10.7.2018.

Westfall, L. 2006. Software Requirements Engineering: What, Why, Who, When, and How. The Westfall Team. Luettavissa: http://www.westfallteam.com/Papers/The_Why_What_Who_When_and_How_Of_Software_Requirements.pdf. Luettu: 14.8.2018.

Liitteet

Liite 1. Haastattelukysymykset projektin kehitystiimin vetäjälle

1. Miten kuvailisit käytössänne olevaa kehitysmallia?
2. Onko käytössänne oleva malli toimiva, onko sinulla ideoita mallin parantamiseksi? Missä suurimmat ongelmat ovat, jos niitä on?
3. Toteutatteko ohjelmistokehitystä projektilähtöisesti?
4. Mistä kehityssuunnan vaihtuminen johtuu?
5. Onko teillä olemassa suunnitelma tulevista kehitysvaiheista?
6. Kuinka osavaiheet aikataulutetaan?
7. Millaiset henkilöt asiakkaan puolelta toteuttaa järjestelmänmäärittelyjä ja suunnittelua?
8. Koetko, että asiakkaan puolelta vaadittaisiin koulutusta?
9. Pidetäänkö säännöllisiä palaveria asiakkaan kanssa?
10. Pidetäänkö säännöllisiä palaveria kehitystiimin kanssa?
11. Miten kehitysideat siirtyvät kehitysputkeen?
12. Kuka ylläpitää kehitysjonoa?
13. Miten idean eteenpäinvieminen tapahtuu, kun se on hyväksytty kehitettäväksi?
14. Onko mahdollista, että kehitystyötä ei aloiteta, vaikka idea siirtyy kehitysputkeen?
15. Miten kehitysjonon tikettien muutoksiin kehitystiimissä ja asiakkaan puolelta suhtaudutaan ja voidaanko niihin tehdä muutoksia?
16. Aikataulutetaanko hyväksyttyjä kehitystöitä, jos aikataulutetaan, miten ja kenen toimesta?
17. Miten kehitystyön etenemistä seurataan?
18. Miten kehitystöiden etenemisen tilanteesta ilmoitetaan asiakkaalle?
19. Testataanko asiakkaan puolesta järjestelmää esimerkiksi uusien kehitettyjen ominaisuuksien osalta? Jos testataan, miten testitapaukset luodaan, miten testaus toteutetaan ja kuinka usein?
20. Mitä tapahtuu, jos tuotantoon viennin yhteydessä havaitaan ongelmia?
21. Miten havaituista ongelmista informoidaan asiakasta?

Liite 2. Haastattelukysymykset projektin asiakkaan edustajalle

1. Miten kuvailisit käytössänne olevaa kehitysmallia?
2. Onko käytössänne oleva malli toimiva, onko sinulla ideoita mallin parantamiseksi? Missä suurimmat ongelmat ovat, jos niitä on?
3. Toteutatteko ohjelmistokehitystä projektilähtöisesti?
4. Onko teillä olemassa suunnitelma tulevista kehitysvaiheista?
5. Kuinka osavaiheet aikataulutetaan ja kenen toimesta?
6. Millaiset henkilöt teidän (asiakkaan) puolelta toteuttaa järjestelmänmäärittelyjä ja suunnittelua?
7. Miten kehitysideat siirtyvät kehitysputkeen?
8. Miten idean eteenpäinvieminen tapahtuu, kun se on hyväksytty kehitettäväksi?
9. Onko mahdollista, että kehitystyötä ei aloiteta, vaikka idea siirtyy kehitysputkeen?
10. Miten kehitysjonon tikettien muutoksiin kehitystiimissä ja asiakkaan puolelta suhtaudutaan ja voidaanko niihin tehdä muutoksia?
11. Aikataulutetaanko hyväksyttyjä kehitystöitä, jos aikataulutetaan, miten ja kenen toimesta?
12. Miten kehitystyön etenemistä seurataan?
13. Miten kehitystöiden etenemisen tilanteesta ilmoitetaan teille (asiakkaalle)?
14. Testataanko teidän (asiakkaan) puolesta järjestelmää esimerkiksi uusien kehitettyjen ominaisuuksien osalta? Jos testataan, miten testitapaukset luodaan, miten testaus toteutetaan ja kuinka usein?
15. Mitä tapahtuu, jos tuotantoon viennin yhteydessä havaitaan ongelmia?
16. Miten havaituista ongelmista informoidaan teille (asiakkaalle)?