VAASAN YLIOPISTO
TEKNILLINEN TIEDEKUNTA
TIETOTEKNIIKKA

Henri Nyholm

TAPAUSTUTKIMUS: MITEN IT-ORGANISAATIO HYÖDYNTÄÄ KETTERÄN KEHITYKSEN ARVOJA JA PERIAATTEITA OHJELMISTOPROJEKTEISSA?

Tietotekniikan
pro gradu -tutkielma

VAASA 2016

SISÄLLYSLUETTELO sivu

1 JOHDANTO 6
1.1 Tutkimuksen tavoite ja rajaus 7
1.2 Tutkimusaineisto –ja menetelmät 8
1.3 Tutkielman rakenne 9

2 OHJELMISTOPROJEKTI 10
2.1 Elinkaari 10
2.1.1 Elinkaarimalli ja elinkaaren vaiheet 11
2.1.2 Vesiputousmalli 13
2.1.3 Iteratiivinen malli 14
2.2 Prosessit 14

3 KETTERÄ OHJELMISTOKEHITYS 19
3.1 Ketteryyden määrittely ja ideologia 19
3.2 The Agile Alliance ja Agile Manifesto 21
3.3 Ketterän ja perinteisen ohjelmistokehityksen eroavaisuudet 23
3.4 Ketteryys ja projektihallinta 29

4 KETTERIÄ MENETELMIÄ 33
4.1 Extreme Programming (XP) 33
4.1.1 Arvot ja käytännöt 34
4.1.2 Säännöt 36
4.1.3 Roolit ja tiimit 37
4.1.4 Elinkaari 39
4.2 Scrum 41
4.2.1 Empiirinen prosessienhallinta 41
4.2.2 Scrum-roolit 42
4.2.3 Scrum-viitekehys 43
4.3 Kanban 46
4.3.1 Viitekehys 47
4.3.2 Kanban-taulu 48

5 TUTKIMUSASETELMA 50
5.1 Aineistonkeruumenetelmät 50
5.1.1 Kysely 50
5.1.2 Teemahaastattelut 51
5.2 Tutkimusaineiston analysointi 52

6 TUTKIMUSAINEISTO JA -TULOKSET 54
6.1 Projektien perustiedot 54
6.2 Esikyselyn tulokset 56
6.3 Teemahaastattelun tulokset 58
6.3.1 Yksilöt ja kanssakäyminen 58
6.3.2 Toimiva ohjelmisto 60
6.3.3 Asiakasyhteistyö 62
6.3.4 Muutokseen vastaaminen 66
6.4 Yhteenveto tutkimustuloksista 69

7 JOHTOPÄÄTÖKSET 74

LÄHDELUETTELO 75

LIITTEET 79

LIITE 1. Esikysely 79

LIITE 2. Teemahaastattelukysymykset 80

KUVALUETTELO **sivu**

Kuvio 1. Vesiputousmallin vaiheet (Royce 1970) 13
Kuvio 2. Prosessilajit (IEEE Std 15288 2008: 12). 15
Kuvio 3. Organisaation prosessit (ISO/IEC 15288 2005:60.) 16
Kuvio 4. Extreme Programming -projektin sisältö kuvattuna yleisellä tasolla (Schuh 2005: 21) 34
Kuvio 5. Extreme Programming -elinkaaren vaiheet mukailtuna (Abrahamsson ym. 2002:19) 39
Kuvio 6. Scrum -viitekehys (Rubin 2013:17) 44
Kuvio 7. Esimerkki Kanban -taulusta (VersionOne 2015). 48

TAULUKKOLUETTELO **sivu**

Taulukko 1. Ohjelmiston elinkaaren vaiheet (ISO/IEC 15288 2005:57.) 12
Taulukko 2. Ketterän ja perinteisen lähestymistavan merkittävimmät eroavaisuudet osa-alueittain. (Awad 2005.) 25
Taulukko 3. Ketterän ja perinteisen lähestymistavan eroavaisuudet periaatteittain. (Aitken & Ilango 2013.) 27
Taulukko 4. Tärkeimmät menestystekijät ketterässä ohjelmistokehitysprojektissa ominaisuuksineen (Chow & Cao 2007). 32
Taulukko 5. Esikyselyn teemat 51
Taulukko 6. Teemahaastattelun teemat 51
Taulukko 7. Projektien perustiedot 55
Taulukko 8. Esikyselyn teemoittain saadut vastauskeskiarvot projektikohtaisesti 56
Taulukko 9. Yksilöt ja kanssakäyminen tutkimuskohteena toimineissa projekteissa 58
Taulukko 10. Toimiva ohjelmisto tutkimuskohteena toimineissa projekteissa 61
Taulukko 11. Asiakasyhteistyö tutkimuskohteena toimineissa projekteissa 63
Taulukko 12. Muutokseen vastaaminen tutkimuskohteena toimineissa projekteissa 66

VAASAN YLIOPISTO
Teknillinen tiedekunta
Tekijä: Henri Nyholm
Tutkielman nimi: Tapaustutkimus: Miten IT -organisaatio hyödyntää ketterän kehityksen arvoja ja periaatteita ohjelmistoprojekteissa?
Ohjaajan nimi: Laura Lappalainen
Tutkinto: Kauppatieteiden maisteri
Oppiaine: Tietotekniikka
Opintojen aloitusvuosi: 2013
Tutkielman valmistumisvuosi: 2016 **Sivumäärä**: 82

TIIVISTELMÄ:

Tämän pro gradu -tutkielman tutkimusaiheena on ketterä ohjelmistokehitys sekä ketterän kehityksen arvot ja periaatteet. Tutkielmassa luodaan lukijalle selkeä käsitys ketterän ohjelmistokehityksen taustasta sekä ketteryyden periaatteista ja arvoista. Tutkielman tavoitteena on selvittää mitä hyötyjä organisaatio voi saavuttaa ketterillä toimintatavoilla ja toisaalta, voiko ketteryys muodostua haitaksi tai uhaksi organisaatiolle. Tätä tarkastellaan erityisesti tutkimuskohteena olevan yrityksen tapauksessa. Tutkielmassa selvitetään mikä tai mitkä periaatteet ketterässä kehityksessä vaativat edelleen organisaatiotasolla lisähuomiota ja kuinka näitä periaatteita sekä niihin liittyviä käytäntöjä olisi mahdollista kehittää.

Tutkielman teoreettinen viitekehys on muodostettu aiemmasta tutkimusaineistosta induktiivisena kirjallisuuskatsauksena. Teoriakokonaisuus johdetaan ohjelmistokehityksen alan kirjallisuudesta sekä tieteellisistä artikkeleista. Keskeisimpiä tutkimuksessa hyödynnettyjä lähteitä ovat *the Agile Alliancen* julkaisut sekä Robert C. Martinin teos *Agile software development: principles, patterns and practices*, joihin perustuu ketterän kehityksen arvot ja periaatteet. Tutkimuksen empiirinen osa on toteutettu teorioita testaavana kolmen (3) casen tapaustutkimuksena. Pyrkimyksenä on 1) löytää yhtäläisyyksiä onnistuneista projekteista ja 2) pystyä tuottamaan ennakoidusta syystä vastakkaisia tuloksia.

Tutkimustuloksilla voidaan selkeästi osoittaa, kuinka kohteena toimiva IT-organisaatio hyödyntää *the Agile Manifeston* mukaisia ketterän kehityksen periaatteita ja arvoja. Tutkimuksen tuloksena saadaan aikaan kuvaus ketterän kehityksen myötä organisaation saavuttamista hyödyistä sekä mahdollisista kehityskohteista tai haasteista. Tutkimustulosten perusteella voidaan vahvistaa, että ketterän ohjelmistokehitysprojektin onnistumisen tason ja projektissa hyödynnettyjen ketterien periaatteiden ja arvojen välillä on riippuvuus.

AVAINSANAT: ketterä kehitys, ketterät menetelmät, ohjelmistokehitysprojekti, projektihallinta

UNIVERSITY OF VAASA
Faculty of technology
Author: Henri Nyholm
Topic of the Master's Thesis: Tapaustutkimus: Miten IT -organisaatio hyödyntää ketterän kehityksen arvoja ja periaatteita ohjelmistoprojekteissa?
Instructor: Laura Lappalainen
Degree: Master of Science, Economics and Business Administration
Major: Computer Science
Year of entering the University: 2013
Year of completing the Thesis: 2016 **Pages**: 82

ABSTRACT:

Subject of this Master's thesis is agile software development and agile values and principles. Theoretical part of the study is aiming to form clear picture of the background of agile software development and values and principles behind it. The aim of this study is to find out what kind of benefits an organization can reach by using agile values and principles and also to find out if those can militate against organization. This issue will be treated especially in the case of the subject organization. This thesis aims to find out which agile principles require more attention at organization level and how those principles could be developed.

The theoretical framework of the study has been formed of earlier publications and studies as an inductive literary review. The theoretical framework is based on literature of the discipline of software development and scientific researches. Publications of *the Agile Alliance* and *Agile software development: principles, patterns and practices* by Robert C. Martin form the most important references of the study. The empirical part of the study has been conducted as a case study that is based on three cases. The aim was 1) to find out similarities of successful projects and 2) to be able to find out oppositions of projects.

The results of the study show how the subject organization takes advantage of the agile values and principles mentioned in the Agile Manifesto. As a result, this study forms a picture of benefits achieved and possible targets of development in the subject organization based on using agile software development. The results of the study confirm that there is relation between successful projects and the level of agility used in the projects.

KEYWORDS: agile development, agile methodologies, software development project, project management

1 JOHDANTO

Projekti on tapa saavuttaa jokin päämäärä selkeästi suunniteltuna hankkeena. Perinteisesti projekteja voidaan löytää arjen yksinkertaisimmistakin ilmiöistä – esimerkiksi yksittäisen kotitalouden tietystä arvoltaan suuremmasta hankinnasta tai vaikkapa oppilaiden peruskouluesitelmästä. Tekniikan alalla ja erityisesti IT -organisaatioissa tuotteita ja palveluita rakennetaan projektiluontoisesti. Projekti ei toki ole vain tämän tieteen ja teollisuuden alan käyttämä tapa kehittää tuotteita. IT-projektit ovat usein muiden palvelualojen projekteihin verrattuina aikataulullisesti hyvin laajoja ja saattavat olla kestoltaan merkittävän pitkiä, jopa useita vuosia. Digitalisaation murroksen myötä juuri IT –palvelualan projektit ovat hyvin seurattuja ja jokapäiväinen osa lähes kaikkien toimialojen uudistumista sekä liiketoiminnan kehittymistä digitaalisempaan suuntaan. IT –palvelualalla toimivat organisaatiot pyrkivät parantamaan asiakkaidensa kilpailukykyä kehittämällä näiden liiketoimintaa digitaalisin ratkaisuin. Projekteilla pyritään saamaan aikaan mahdollisimman suuri liiketoiminnallinen hyöty sekä asiakas- että toimittajaorganisaation näkökulmasta. Ominaista projekteille on käytössä olevat resurssit, olemassa olevat riskit, projektiin kuuluvat henkilöt tai tahot, mutta ennen kaikkea se, miten projektit kokonaisuutena toteutetaan ja kuinka projektin eri vaiheita hallitaan.

Ohjelmistotuotannossa on viimeisten kahden vuosikymmenen ajan tehty muutosta kohti uudenlaista projektihallintaa – ketterää ohjelmistokehitystä. Ketterällä ohjelmistokehityksellä ja ketterillä menetelmillä pyritään luomaan uusi tapa toimia raskaampien vaihejakomallien, kuten vesiputousmallin sijaan. Muutos kohti ketterää kehitystä perustuu alan asiantuntijoiden näkemyksiin siitä, miten ohjelmistokehitystä voitaisiin tehdä paremmin ja tehokkaammin keinoin päätavoitteena minimoida riskejä sekä toimia mahdollisimman kustannustehokkaasti. Ketterät menetelmät ovat levinneet laajalle vuoden 2001 ketterän ohjelmistokehityksen julistuksen, the Agile Manifeston esiintulon jälkeen, pyrkien vastaamaan edellä mainittujen perinteisten menetelmien raskaaseen rakenteeseen. Ketterässä kirjallisuudessa ketterillä menetelmillä viitataan ketterän allianssin Agile Alliancen asiantuntijoiden kehittämiin ja jalostamiin menetelmiin, kuten Scrumiin tai eXtreme Programmingiin. Menetelmäkohtaisista yksityiskohdista huolimatta näillä useilla eri menetelmillä on paljon yhteistä, kuten lyhyisiin iteraatioihin perustuvat elinkaaret, nopea ja suora palaute asiakkailta sekä jatkuva oppiminen.

Tässä tutkielmassa esitetään tapaustutkimuksena erään IT -organisaation tapa toteuttaa ketterää kehitystä sekä kuvataan ketterän ohjelmistokehityksen ominaispiirteet sekä käytetyimmät menetelmät. Tutkielmassa projektihallintaa tarkastellaan nykyaikaisesta näkökulmasta sekä luodaan käsitys siitä, että ketterä kehitys perustuu yksinkertaisten asiakokonaisuuksien oikeanlaiseen hallintaan ja sitä on mahdollista hyödyntää lopulta lähes missä tahansa projektissa.

1.1 Tutkimuksen tavoite ja rajaus

Tämän pro gradu -tutkielman teoreettinen viitekehys luo lukijalleen selkeän käsityksen ketterän ohjelmistokehityksen taustasta sekä ketteryyden periaatteista ja arvoista. Siinä kuvataan näiden periaatteiden ja arvojen ilmentyminen sekä merkitys ketterän ohjelmistoprojektin aikana. Tutkielman empiirisessä tutkimusosiossa selvitetään, kuinka IT -ratkaisuja –ja palveluita tarjoava organisaatio hyödyntää ketterän ohjelmistokehityksen arvoja ja periaatteita ohjelmistoprojekteissa käytännössä. Mikäli kohdeorganisaatio ei hyödynnä projekteissaan mitään tiettyjä tai tiettyä yleisesti käytettyä ohjelmistonkehitysmenetelmää (kuten *Scrum* tai *eXtreme Programming*), määritellään sen käyttämä menetelmä ja selvitetään sen suhde muihin yleisesti käytettyihin menetelmiin.

Tutkielman tavoitteena on selvittää mitä hyötyjä tutkimuksen kohteena oleva organisaatio voi saavuttaa ketterillä toimintatavoilla ja toisaalta, voiko ketteryys muodostua haitaksi tai uhaksi organisaatiolle. Tutkielmassa saadaan selville mikä tai mitkä periaatteet ketterässä kehityksessä vaativat edelleen organisaatiotasolla lisähuomiota ja kuinka näitä periaatteita sekä niihin liittyviä käytäntöjä olisi mahdollista kehittää.

Tutkielman tutkimuskysymys on:

Kuinka IT -ratkaisuja –ja palveluita tuottava organisaatio hyödyntää ketterän ohjelmistokehityksen arvoja ja periaatteita?

Tutkimuskysymys voidaan jakaa seuraaviin alakysymyksiin:

- Käytetäänkö organisaatiossa yleisesti käytettyjä ketteriä menetelmiä? Jos ei, onko sillä jokin oma menetelmä ja mikä on sen suhde yleisiin kehitysmenetelmiin?

- Miten organisaatio hyötyy ketteristä arvoista ja periaatteista?
- Onko projektien onnistuneisuuden ja niissä hyödynnettyjen ketterien arvojen ja periaatteiden välillä riippuvuutta?

1.2 Tutkimusaineisto ja -menetelmät

Tässä pro gradu -tutkielmassa on käytetty kahta eri menetelmää tutkimusaineiston keräämiseen. Esikyselyn avulla kerätään ennakkotietoa tutkimuskohteina olevista projekteista ja teemahaastattelulla tarkennetaan kyselyssä esiintyviä näkökulmia. Tutkimusaineisto perustuu kolmeen tutkimuskohteena olleeseen ohjelmistokehitysprojektiin. Alla on esitelty käytössä olleet aineistonkeruumenetelmät, jotka ovat kysely ja teemahaastattelu.

Ensimmäinen tutkielman tiedonkeruutekniikoista on kysely. Kysely on tutkijan itsensä tapa kerätä aineistoa. Se on keskeinen menetelmä survey-tutkimukselle, mutta käytännöllinen myös yhdistettynä kvalitatiiviseen tutkimusmenetelmään. Tässä tutkimuksessa kysely toimii tukiaineistona ja tuo siten lisäarvoa juuri yhdistettynä kvalitatiiviseen aineistonkeruumenetelmään. Kysely voidaan toteuttaa monin erilaisin keinoin ja usealla eri tavalla. Kyselyä valittaessa tai käytettävää aineistonkeruumenetelmää perustellessa on tärkeää pohtia, milloin koehenkilöiden on saatava toimia vapaasti ja milloin on järkevämpää käyttää ennalta määrättyä strukturoitua tekniikkaa. Kyselylomakkeen huolellinen laatiminen – tutkimusaiheen valitsemisen ohella – voi olla hyvinkin merkittävässä asemassa tutkimuksen onnistumista ajatellen. Kysely voidaan perustaa avoimiin kysymyksiin, monivalintakysymyksiin tai asteikkoihin eli skaaloihin. Tässä tutkielmassa valittu asteikkoihin perustuva kysymystyyppi perustuu ajatukseen, että vastaaja antaa näkemyksensä siitä, kuinka voimakkaasti on samaa tai eri mieltä esitetystä väittämästä. (Järvinen & Järvinen 2011: 147.)

Haastattelutekniikaksi voidaan valita tutkielmasta riippuen avoin, puolistrukturoitu tai strukturoitu haastattelu riippuen siitä, onko kysymysten vastausvaihtoehdot ennalta määritellyt. Tässä tutkielmassa tekniikaksi on valittu avoin haastattelu, jossa tutkimusteemat ohjaavat haastattelua. Tässä tutkimuksessa teemat on hyvin selkeästi johdettavissa teoreettisesta viitekehyksestä, joten teemoihin perustuva haastattelu on toimivin vaihtoehto. Haastattelun kohteiksi on valittu kohdeorganisaatiosta kolme edustajaa, jotka ovat toimineet johtotehtävässä ketterässä ohjelmistoprojektissa. Haastattelussa

saadaan avoimien kysymysten avulla selville mahdollisimman kattavasti organisaatiossa hyödynnettyjä ketteriä toimintatapoja. Haastattelussa tutkija keskustelee haastattelun kohteena olevan henkilön kanssa, joka edustaa haastattelussa tietolähdettä. Haastattelun etuna on juuri elävä vuorovaikutustilanne, jolloin tutkijan on mahdollista haastattelutilanteessa tarkentaa kysymyksiään lisäkysymyksillä ja siten saada lisää uutta tietoa. Haastattelussa on erityisen tärkeää, että tutkija kykenee asettumaan haastateltavan asemaan ja katsomaan tilannetta tämän silmin. (Järvinen & Järvinen 2011:.)

1.3 Tutkielman rakenne

Tutkielman toisessa luvussa esitellään tutkielman lähtökohta – ohjelmistoprojekti sekä sen elinkaari ja elinkaaren sisältö. Tutkielman kolmannessa luvussa esitellään tutkimusaiheen perusta eli ketterä ohjelmistokehitys ja luodaan lukijalle käsitys ketteryyden lähtökohdista sekä taustalla merkittävästi vaikuttavasta Agile Alliancesta. Neljännessä luvussa esitellään tutkielman läpiviennin kannalta tärkeässä roolissa olevia ketteriä menetelmiä ja niiden perusominaisuuksia. Tutkielman viides luku sisältää kuvauksen tutkimusympäristönä toimineesta kohdeorganisaatiosta sekä tutkielmassa käytetyistä tutkimusmenetelmistä. Kuudes luku käsittää itse tutkimusaineiston sekä tutkimustulokset. Seitsemännessä ja viimeisessä luvussa kuvataan tutkimuksen yhteenveto sekä esitetään mahdollisia jatkotutkimuskohteita.

2 OHJELMISTOPROJEKTI

Ohjelmistotuotteita rakennettaessa ja teknisiä ratkaisuja tarjotessa hyödynnetään usein erilaisia projektimalleja, joita perinteisesti ajatellaan elinkaarina. Ohjelmistoprojektin elinkaari alkaa tilannekartoituksella ja asiakkaan intressien määrittämisellä ja päättyy tilanteeseen, jossa ohjelmistotuotteen voidaan todeta olevan valmis siirrettäväksi syrjään ja mahdollisesti korvattavaksi uudella, nykyaikaisemmalla ja kehittyneemmällä tuotteella. Tällainen projekti tai elinkaari koostuu tietyistä vaiheista, jotka sisältävät usein tiettyjä prosesseja. Elinkaari määritellään tarkemmin alaluvussa 2.1.. Tekniikan alalla hyödynnetään yleisesti standardeja, jotka toimivat kansainvälisinä ohjeistuksina ja suuntaviivoina yhdenmukaisuuden takaamiseksi. Tässä luvussa kuvataan tekniikan alalla hyödynnettävän standardin IEEE Std 15288™ pohjalta ohjelmiston elinkaarta ja siihen sisältyvät prosessit sekä kuvataan perinteisimmin käytettyjä elinkaarimalleja, vesiputousmallia ja iteratiivista mallia. Vesiputousmalli valittiin esiteltäväksi, koska sitä pidetään perinteisenä elinkaarimallina ohjelmistokehityksessä ja iteratiivinen malli edustaa niin sanottua uutta, kevytrakenteisempaa ajattelua, joka on merkittävä osa ketterää kehitystä.

2.1 Elinkaari

Ohjelmistotuotanto perustuu tuotettavien tuotteiden tai palveluiden elinkaariin. Ohjelmistotuotteen ajatellaan syntyvän ja sen elinkaaren jatkuvan niin kauan, kun se on toimintaympäristönsä käytössä. Ohjelmistojen elinkaarta on kuvattu alalle ominaisesti standardein. ISO/IEC 15228 -standardi kuvaa ohjelmiston elinkaaren, siihen kuuluvat prosessit ja sen toimintaympäristön tarkemmin. Tämän kansainvälisen standardin mukaan kohteena olevaa tuotetta tai palvelua ajatellaan keinotekoisena järjestelmänä, joka on luotu tuottamaan palveluita määrätyssä ympäristössä käyttäjien ja sidosryhmien hyödynnettäväksi. Tällaiset järjestelmät voivat olla määriteltyjä toimimaan yksittäisten tai useampien laitteiden, ohjelmistojen, ihmisten, prosessien, proseduurien, olosuhteiden tai luonnollisten olioiden kanssa. (IEEE Std 15288™ 2008: 52.) Tässä luvussa esitellään elinkaaren peruslähtökohdat sekä alalla perinteisimmin esiintyvät elinkaarimallit.

2.1.1 Elinkaarimalli ja elinkaaren vaiheet

Jokaisella edellä kuvatulla järjestelmällä on siis oma elinkaarensa, jota voidaan luonnehtia käyttäen abstraktia toiminnallista mallia, joka kattaa järjestelmän tarpeen, toteuttamisen, hyödynnettävyyden, kehitettävyyden ja hävittämisen. Tällainen järjestelmä kehittyy elinkaarensa aikana ihmisistä koostuvien organisaatioiden hallinnoimien ja toteuttamien prosessien tulosten myötä. Nämä prosessit sekä niiden seuraukset, suhteet ja vaiheet määrittävät järjestelmän elinkaarimallin yksityiskohdat. Standardissa on määritetty joukko nimettyjä prosesseja, joiden pohjalta on mahdollista mallintaa järjestelmän elinkaarta. (IEEE Std 15288™ 2008: 10.)

Elinkaaret vaihtelevat järjestelmän luonteen, tarkoituksen, käytön ja vallitsevien olosuhteiden mukaan. Huolimatta rajattoman laajasta elinkaarten kirjosta, on olemassa tietyt, välttämättömät vaiheet, jotka löytyvät jokaisesta kokonaisvaltaisesta järjestelmäelinkaaresta. Kukin vaihe suunnitellaan ja toteutetaan elinkaaren aikana harkiten sekä niillä on selkeä tarkoitus ja panos koko elinkaaressa. Vaiheet edustavat järjestelmän elinkaaren pääperiodeita ja ne koskevat järjestelmäkuvauksen tai itse järjestelmän tilaa. Elinkaarivaiheet kuvaavat järjestelmän edistymistä, kehitystä ja sen saavuttamia virstanpylväitä elinkaaren aikana. Organisaation luodessa tai hyödyntäessä kohteena olevaa järjestelmää, se käyttää ns. päätösportteja (*engl. Decision gates*), jotka määrittävät seuraavaksi toteutettavat toimet järjestelmän elinkaarella. Päätösporttien avulla voidaan hallita elinkaaren varrelle olennaisesti kuuluvia riskejä ja epätietoisuutta liittyen kustannuksiin, aikatauluun ja toimivuuteen. Elinkaaren päävaiheet luovat perustan näille ensisijaisille päätösporteille. Näin ollen elinkaarivaiheiden avulla yritysjohdolla on korkeantason näkemys ja kontrolli toteutettavasta projektista tai teknisestä prosessista. (IEEE Std 15288™ 2008: 10-11.)

Taulukossa 1 on esitelty yleisimmin käytetty esimerkki elinkaarivaiheista. Taulukossa on myös kuvattu kunkin vaiheen lähtökohtainen tarkoitus sekä mahdolliset päätösvaihtoehdot riskien- ja saavutustenhallintaan elinkaaren etenemisen aikana. Organisaatiot soveltavat vaiheita ja niiden järjestystä eri tavoin kyetäkseen vastaamaan käytössä oleviin liiketoiminta- ja riskinhallintastrategioihin. Vaiheiden samanaikainen ja eri järjestyksissä tapahtuva toteuttaminen voi johtaa ominaisuuksiltaan hyvin erilaisiin elinkaariin. Vaiheittain järjestyksessä toteutettavat tavat ovat yleisesti käytettyjä, mutta vaihtoehtoisesti sopiva näiden risteymä voidaan kehittää. Standardissa viitataan

tällaisen kehityksen riippuvan useista tekijöistä, kuten vallitsevasta liiketoimintakontekstista, järjestelmän luonteesta ja monimutkaisuudesta, vaatimusten stabiiliudesta, teknologisista mahdollisuuksista, järjestelmän suorituskykyyn kohdistuvista vaatimuksista eri vaiheissa sekä käytössä olevista resursseista ja budjetista. (ISO/IEC 15288 2005: 57.)

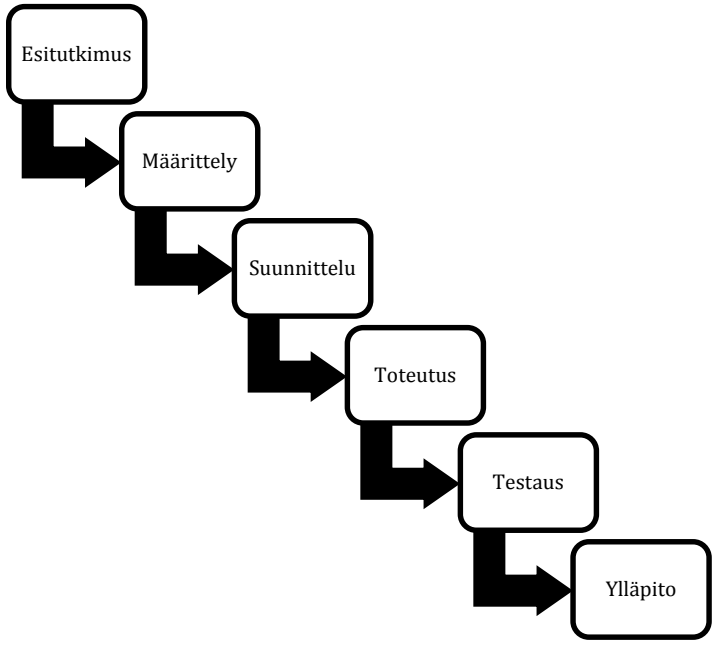
ELINKAARIVAIHE	TARKOITUS	PÄÄTÖSPORTIT
KONSEPTI	• Identifioi sidosryhmien tarpeet • Tutki konsepteja • Esitä toteuttamiskelpoisia ratkaisuja	• Toteuta seuraava vaihe • Jatka tässä vaiheessa • Palaa edeltävään vaiheeseen • Jäädytä projekti • Päätä projekti
KEHITYS	• Jalosta järjestelmän vaatimukset • Laadi ratkaisun kuvaus • Rakenna järjestelmä • Verifioi ja validoi	
TUOTANTO	• Toteuta järjestelmät • Tutki ja testaa	
HYÖDYNTÄMINEN	• Hoida järjestelmää tyydyttääksesi asiakkaan tarpeet	
TUKI	• Ylläpidä järjestelmän suorituskykyä	
VETÄYTYMINEN	• Järjestelmän varastointi, arkistointi tai hävittäminen	

Taulukko 1. Ohjelmiston elinkaaren vaiheet (ISO/IEC 15288 2005:57.)

Jokainen vaihe tulee ottaa huomioon läpi järjestelmän elinkaaren ja niitä tulee harkita kaikissa muissakin elinkaaren vaiheissa. Jokainen järjestelmän elementti vaikuttaa kokonaisuuteen, joten eri vaiheiden ja järjestelmän osien tulee voida toimia yhtenä, eheänä kokonaisuutena läpi elinkaaren. Tällainen elinkaarivaiheiden ja toiminnallisten vaikuttajien välinen synergia on ehdotonta onnistuneiden projektitoimintojen saavuttamiseksi. Projektitiimin jäsenten välinen kommunikaatio sekä erilaisten toimijoiden ja organisaatioiden vastuuntuntoisuus eri elinkaaren vaiheissa johtavat johdonmukaiseen sekä yhtenäiseen elinkaarikokonaisuuteen. (IEEE Std 15288™ 2008: 10-11.)

2.1.2 Vesiputousmalli

Ensimmäisen ohjelmistokehitysprojekteissa käytetyn elinkaarimallin, vesiputousmallin esitteli nykyisessä muodossaan Winston W. Royce (1970). Vesiputousmalli on yksinkertainen, vaiheesta seuraavaan etenevä elinkaarimalli. Vesiputousmalli perustuu nimensä mukaisesti siihen, että jokainen mallin vaihe suoritetaan vain kerran ja loppuun asti, kunnes siirrytään elinkaaren seuraavaan vaiheeseen – vesiputouksessa ei liikuta ylöspäin.



Kuvio 1. Vesiputousmallin vaiheet (Royce 1970)

Vesiputousmallin esitutkimusvaiheessa asetetaan asiakkaan näkemysten pohjalta yleiset vaatimukset koko toteutettavalle ohjelmistolle. Määrittelyvaiheessa tarkennetaan esitutkimusvaiheen tuloksena syntyneitä yleisiä vaatimuksia, laaditaan ohjelmistolle toiminnalliset ja tekniset vaatimukset sekä ei-toiminnalliset vaatimukset ja rajoitukset. Määrittelyn tuloksen syntyy dokumentti, joka tunnetaan toiminnallisena määrittelynä tai vaatimusmäärittelydokumenttina. Suunnitteluvaiheessa määritetään se, miten ohjelmisto lopulta rakennetaan. Tuloksena syntyy tekninen dokumentaatio, joka käsittää muiden muassa ohjelmiston arkkitehtuurin sekä moduulisuunnittelun. Toteutusvai-

heessa tehdään itse koodaustyö ja toteutetaan halutunlainen ohjelmisto. Testausvaiheessa tuotettu ohjelmisto testataan vaiheissa ja pyritään löytämään tuotetusta koodista mahdolliset virheet. Käyttöönottovaihe perustuu ohjelmiston julkaisemiseen asiakkaan käyttöön ja ylläpitovaihe ohjelmiston ylläpitotyöhön, kuten myöhemmin löytyvien virheiden korjaamiseen, ohjelmiston muutoksiin sekä laajentamiseen. (Haikala & Mikkonen 2011).

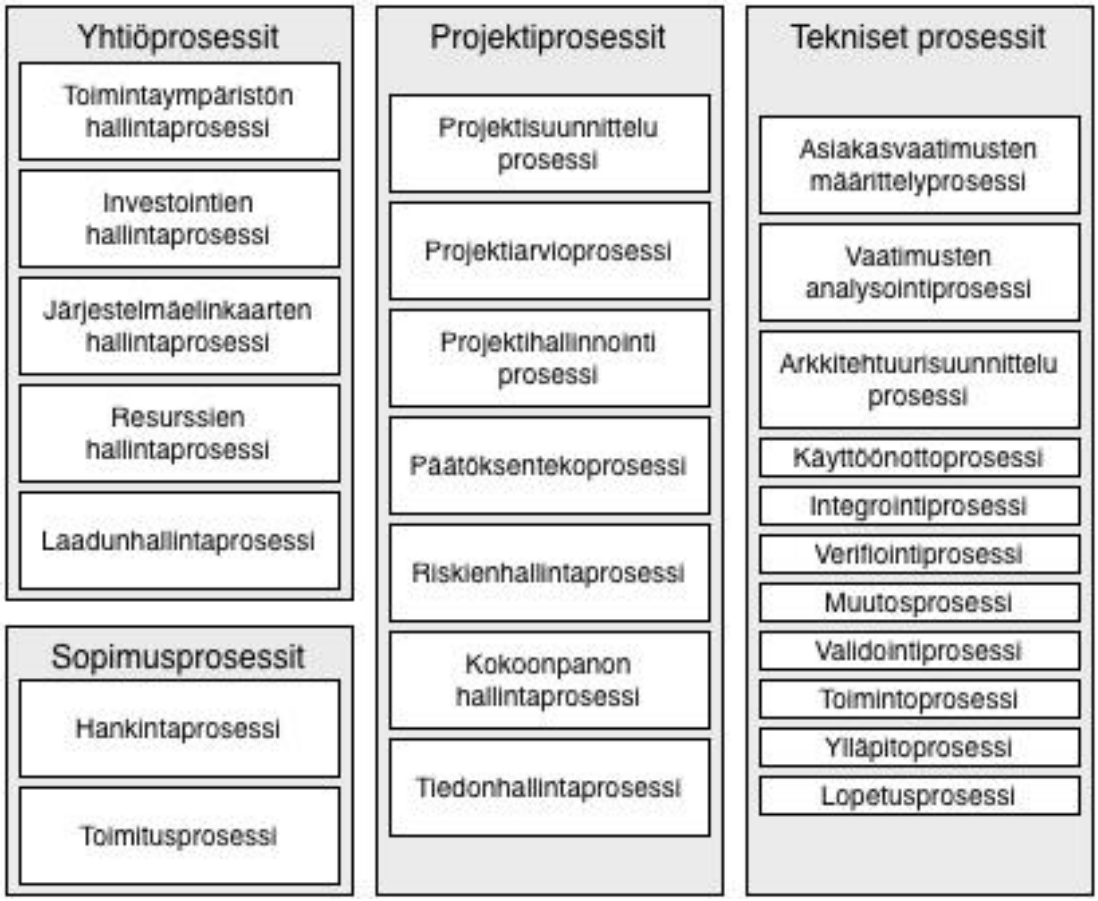
2.1.3 Iteratiivinen malli

Iteratiivinen elinkaarimalli on vuosikymmeniä kehittynyt ideologia, jossa ohjelmiston tuottaminen perustuu perättäisiin sykleihin eli iteraatioihin. Edellä kuvattu vesiputousmalli on vahvasti iteratiivisen mallin taustalla – kaikki vaiheet suoritetaan miniprojektin tavoin iteraatiossa läpi. Kiinteiden määrittelyjen sijaan iteratiivisessa mallissa kehitystyö alkaa määrittelemällä ja suunnittelemalla vain osa ohjelmistosta. Kunkin iteraation jälkeen saadaan uusi versio ohjelmistosta. Iteratiivisen mallin perusajatuksena ovat juuri ajalliset vaatimukset – kullekin iteraatiolle määritetään aikataulu, johon mennessä kyseinen iteraatio tulee päätökseen. Iteraation aikana tehdään aiemmin suunniteltu työ ja iteraation päätyttyä pyritään saamaan ulkoisista lähteistä, kuten asiakkaalta kehitysehdotuksia ja esimerkiksi uusia vaatimuksia, jotka toteutetaan seuraavassa iteraatiossa. Iteratiivisen mallin etuna on se, että kyetään tuottamaan toimiva malli ohjelmistosta hyvin aikaisessa vaiheessa projektia, jolloin toiminnallisten ja rakenteellisten heikkouksien löytäminen on mahdollista jo projektin alussa. (Basili & Larman 2003: 2).

2.2 Prosessit

Ohjelmistoprojekti ja sen myötä ohjelmiston elinkaari kattaa poikkeuksetta tietyn määrän prosesseja, joilla kullakin on fokus jossain tietyssä projektin osassa tai elinkaaren vaiheessa. Tyypillisesti organisaatiot hajauttavat sisäiset liikkeenjohdolliset vastuualueensa ja toimintonsa. Yhdessä nämä alueet kuitenkin vaikuttavat organisaation liiketoiminnalliseen suorituskykyyn. IEEE Std 15288™-standardissa (2008: 59) organisaation prosesseja on käsitelty kolmen vastuualueen näkökulmasta: koko yhtiön kattavat prosessit, projekteja koskevat prosessit ja tekniset prosessit. Kunkin kategorian prosessit yhtenäisenä kokonaisuutena vaikuttavat tehokkaaseen järjestelmien luomiseen ja käyttämiseen, ja ovat sitä kautta merkittävä tekijä organisaation tavoitteiden

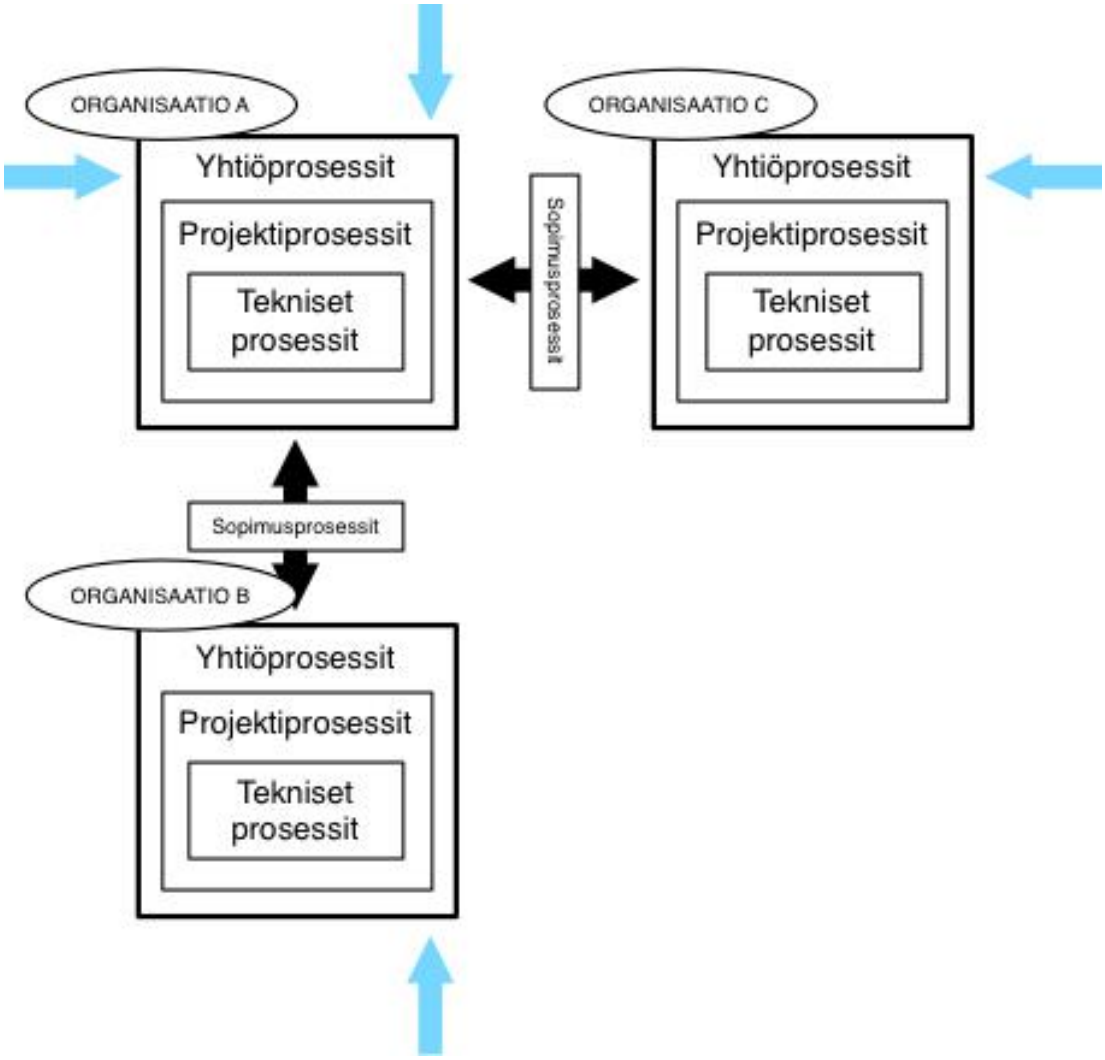
saavuttamisen kannalta. Lisäksi eri organisaatiot ja organisaatioiden eri vastuualueet yleisesti luovat suhteita ja tiedostavat vastuitaan tekemällä sopimuksia. Sopimuksilla pyritään yhtenäistämään ja eheyttämään eri vastuualueilla tehtyjä päätöksiä tavoitellen yhteistä, yhdenmukaista liiketoiminnallista tarkoitusta. Kuviossa 2 on esitetty prosessilajit ja niiden ominaisuudet.



Kuvio 2. Prosessilajit (IEEE Std 15288 2008: 12).

Seuraavissa kappaleissa esitellään kunkin edellä kuvatun prosessilajin pääpiirteet. *Sopimusprosessit* ovat merkittävä tekijä, koska ohjelmistotuotannossa ja tietoteknisessä liiketoiminnassa organisaatiot ovat järjestelmille sekä toimittajia eli palveluntarjoajia että kuluttajia eli asiakkaita ja he käyvät kauppaa tuotteilla sekä palveluilla. Tehtyään sopimuksen, on kaksi organisaatiota tilanteessa, jossa organisaatio, hankkijan asemassa, teettää toisella, toimittajana toimivalla organisaatiolla tuotteita tai palveluita.

Yleisesti organisaatiot toimivat samanaikaisesti tai peräkkäin sekä järjestelmiä hankkivana että tuottavana osapuolena (Kuvio 3). Kuviossa pystysuunnassa esitettyjen organisaatioiden A ja B voidaan ajatella edustavan toimitusketjuun kuuluvia organisaatioita, jotka käyvät kauppaa ollessaan samassa vaiheessa sillä hetkellä työn alla olevan projektinsa elinkaarella.



Kuvio 3. Organisaation prosessit (ISO/IEC 15288 2005:60.)

Samanaikaisesti vaakasuunnassa kuvattu suhde organisaatioiden A ja C välillä esittää tilannetta, jossa organisaatiot ovat perättäisissä vaiheissa elinkaarellaan. Kuviossa 3 esitetään tilannetta, jolla halutaan selvittää, että organisaatiolla on usein sopimuksia

eri yhteistyöorganisaatioiden kanssa samanaikaisesti – jotkut organisaatiot edustavat samassa elinkaaren vaiheessa toimivia, jotkut taas vaiheen edellä tai jäljessä olevia.(IEEE Std 15288™ 2008: 12.)

Yhtiötä kokonaisuutena koskevat eli *yhtiöprosessit* pyrkivät IEEE- standardin 15288™ (2008: 13) mukaan takaamaan, että organisaation hallinnollisten tahojen tarpeet ja odotukset kohtaavat. Tällaiset koko yhtiön kattavat prosessit koskevat yleisesti strategista johtoa, organisaation liiketoiminnan parantamista, resurssien ja varallisuuden järjestelyjä sekä kehittämistä tai riskienhallintaa. Vastuu tällaisista prosesseista on luonnollisesti organisaation korkeimmalla tasolla. Nämä prosessit ovat useissa organisaatioissa liiketoiminnallisen linjan taustalla ja selkeyttävät voiton tavoittelun motiiveja. (IEEE Std 15288™ 2008: 13.)

Projektiprosessit koskevat yritysjohdon allokoimien resurssien ja varallisuuden hallintaa ja niiden kohdentamista organisaation tekemiin sopimuksiin sekä sopimusten täyttämiseen. Prosessit koskevat projektien hallintaa, erityisesti kulujen, aikataulujen ja tavoitteiden suunnittelua sekä toimintojen tarkastamista suunnitelmien ja suorituskyvyn kriteerien mukaisuuden varmistamiseksi. Juuri projektiprosesseilla pyritään tunnistamaan toiminnot, joilla voidaan ehkäistä elinkaarella etenemisen hidastumista. (IEEE Std 15288™ 2008: 13.)

Tekniset prosessit ilmenevät teknisinä toimintoina koko elinkaaren ajan. Niiden tarkoituksena on muuntaa sidosryhmien tarpeet ensin tuotteeksi ja sitten, tuotetta soveltamalla, tuottaa ylläpidettävä palvelu tarpeiden mukaisesti, joka saavuttaa asiakastyytyväisyyden. (IEEE Std 15288™ 2008: 13.)

Standardin IEEE 15288™ (2008: 13) mukaan järjestelmän elinkaaren aikana mikä tahansa prosesseista voidaan toteuttaa vaatimusten mukaan missä tahansa vaiheessa, eikä niiden käytölle ole määrättyä järjestystä. Prosessien toteuttamisen järjestykseen ja ajankohtaan vaikuttaa usein sosiaaliset, kaupalliset, organisatoriset ja tekniset näkökulmat, jotka kaikki voivat vaihdella ja elää merkittävästi järjestelmän elinkaaren aikana. Järjestelmän elinkaari projektikokonaisuutena on monimutkainen prosessien muodostama systeemi, jolla voi olla normaalisti rinnakkaisia, iteratiivisia, toistuvia ja ajasta riippuvia ominaispiirteitä eli vaiheita voidaan tarvittaessa toistaa tai iteroida vallitsevien tarpeiden mukaan. Esimerkiksi prosessien rinnakkaista käyttöä voi ilmetä

useammissa projekteissa, joissa esimerkiksi suunnitteluun ja valmisteluun liittyviä toimia toteutetaan samanaikaisesti ja eri projektien välillä, kun pyritään tuottamaan järjestelmän osia useampiin projekteihin hyödynnettäväksi kustannustehokkuutta korostaen. Prosessien iteratiivinen toteuttaminen –esimerkiksi jonkin prosessin toistuva toteuttaminen tai hierarkkisella tasolla samaan aikaan toteutettava joukko prosesseja– on tärkeässä asemassa jatkuvan prosessitehokkuuden edistyksen kannalta. Esimerkiksi peräkkäisten verifiointi –ja integraatiotoimintojen välinen vuorovaikutus voi parantaa projektin etenemisen yhdenmukaisuutta merkittävästi. (IEEE Std 15288™ 2008: 13.)

Prosessien toistuva käyttäminen eli saman prosessin toistuva toteuttaminen tai prosessijoukon toteuttaminen perättäisissä elinkaaren vaiheissa on esitetyn IEEE Std 15288™ -standardin yksi avainkohdista. Prosessien tulokset informaatio–, artefakti– tai palvelutasolla toimivat lähtökohtina samoille prosesseille, joita toteutetaan ylemmällä tai alemmalla tasolla. Tällaisin toiminnoin voidaan alkuperäisen prosessin tulosta uudelleenkäsiteltynä kehittää ja muokata samaa prosessia toistaen, joka edelleen mahdollistaa työtulosten yhdenmukaisuuden järjestelmän kaikilla arkkitehtuurisilla tasoilla. Järjestelmiin vaikuttavien tekijöiden muuttuva luonne vaatii jatkuvaa tarkkailua liittyen toteutettaviin prosesseihin ja niiden toteutuksen oikeaan ajoitukseen. Tällaisia tekijöitä voivat olla muiden muassa operationaaliset ympäristömuutokset, uudet teknologiset mahdollisuudet, muuttuva järjestelmärakenne tai organisaatiota koskevat muutokset. Prosessien käyttö on siten hyvin dynaamista ja sen aiheuttavat useat ulkoiset tekijät, jotka vaikuttavat järjestelmän toteutukseen. Elinkaaren aikana toteutettavien prosessien suunnitteluun, toteutukseen ja hallintaan vaikuttavat siis tutkielmassa jo aiemmin kuvatut elinkaaren vaiheet, joille on myös määritelty tietty tarkoitus sekä rakenne. Erityisesti samankaltaisilla markkina- ja tuotesektoreilla voidaan yhdenmukaisilla ja helposti uusiokäytettävillä vaihe- ja prosessivalinnoilla rakentaa tarkoituksenmukainen ja tahokas elinkaarimalli mille tahansa järjestelmälle. (IEEE Std 15288™ 2008: 13.)

3 KETTERÄ OHJELMISTOKEHITYS

Tässä luvussa kuvataan tällä hetkellä yleisesti käytössä oleva ketteryyden ja ketterän ohjelmistokehityksen määritelmä. Luvussa selvitetään, mitä ketteryys on, miten se on syntynyt, miten ketterä kehitys poikkeaa perinteisestä kehityksestä ja miten ketteryyttä sovelletaan ohjelmistoprojektiympäristössä. Luvussa käsitellään tutkielman teoreettisen viitekehyksen kulmakivenä toimivat ketterän kehityksen arvot ja periaatteet. Alaluvussa 3.1 määritellään ketteryyden käsite ja ketteryyden syntyyn johtanut ideologia, alaluvussa 3.2 koko ketterän kehityksen peruskivenä toimiva, alan asiantuntijoiden muodostama Agile Alliance sekä heidän Agile Manifesto. Alaluku 3.3 luo lukijalle käsityksen siitä, miten ketterä ohjelmistokehitys eroaa perinteisestä ohjelmistokehityksen ajattelusta ja alaluvussa 3.4 esitellään käytännön projektityössä esiintyviä ketteriä piirteitä sekä kuvataan, millaista projektinhallintaa ketterästi toimiminen edellyttää käytännössä.

3.1 Ketteryyden määrittely ja ideologia

Ketterä tarkoittaa terminä 1) kykyä liikkua nopeasti ja helposti sekä 2) kykyä ajatella ja ymmärtää nopeasti (Oxford Dictionaries 2015). Ketteryyttä konseptina pidetään hyvin monipuolisena. Sitä ovat käyttäneet hyvin monet ihmiset, viitatessaan hyvin erilaisiin ilmiöihin. 1990-luvun alusta lähtien ketteryyttä on käytetty terminä laajalti liiketoiminnan alalla eri kentillä ja tieteenaloilla (Convoy 2009: 330). Cockburnin (2000: 149) mukaan ketterä viittaa tehokkaana sekä helposti liikuteltavana olemiseen. Ketteryys voidaan määritellä tarkoittamaan laatua tai kykyä olla nopealiikkeinen (*engl. quick moving*) sekä vikkelä (*engl. nimble*). Tietojärjestelmän kehittämisen *(Information Systems Development, ISD*) kontekstissa ketteryyden voi määritellä tietojärjestelmän kehitysorganisaation kyvyksi havainnoida ja vastata nopeasti teknisiin muutoksiin ja uusiin liiketoiminnallisiin mahdollisuuksiin (Lyytinen & Rose 2006: 183). Ketteryyden käsitteellistä taustaa tutkinut Kieran Convoy (2009: 341) määrittelee lopulta ketteryyden tarkoittamaan ohjelmistokehityksessä kehitysmenetelmän alituista valmiutta äkillisesti tai olennaisesti aiheuttaa muutosta. Hän viittaa tällä proaktiiviseen eli ennakoivaan tai reaktiiviseen eli ympäröivään ilmiöön reagoivaan muutoksen tukemiseen sekä muutoksesta oppimiseen edistäen asiakkaan kokemaa taloudellista, laadullista ja yksinkertaista lisäarvoa. Tämän kaiken tulisi tapahtua hyödyntäen menetelmän kollektiivisia komponentteja sekä sen suhdetta ympäristöönsä.

Ohjelmistokehityksessä ketteryydellä pyritään tarjoamaan vastine liiketoimintayhteisölle, joka vaatii nopeilla kehitysprosesseilla kevyempiä rakenteita. Äkillisesti kasvavan ja epävakaan Internet-ohjelmistoteollisuuden kuten myös kasvavan mobiilin sovellusympäristön tapauksissa tilanteeseen törmätään yhä useammin. (Abrahamsson ym. 2002: 9.)

Highsmith (2002: XXIII) kuvaa, että ailahtelevaisuus ja epävarmuus määrittävät yhä enemmän tämän päivän liiketoimintaa. Lahjakkaat, tekniset ihmiset haluavat työskennellä organisaatioissa, joissa heillä on enemmän valtaa sen suhteen kuinka he työskentelevät ja kuinka he vuorovaikuttavat kollegoidensa, asiakkaidensa ja johtonsa kanssa. Ongelmat, ihmiset ja ideat muuttuvat. Vaikka edelleen suunnitelmalähtöisillä kehitys- ja hallintatyyleillä on mahdollisuuksia ja jalansijaa, on kasvu ja kehitys ketteränä ja joustavana olemisen varassa. Ketterän kehityksen vaikutuksia ja ketterien menetelmien sopeuttamista tutkinut Peter Schuh (2005: 2) kuvaa, että ketterä kehitys on ohjelmiston rakentamismenetelmä, jossa luotetaan ihmisiin, tiedostaen muutos normina ja suosien alituista palautetta. Hänen mukaan ketterä lähestymistapa voidaan tiivistää seuraaviin periaatteisiin:

- Käyttökelpoisen ja asiakkaalle arvokkaan ohjelmiston toimittaminen on ohjelmistokehitysprojektin tärkein yksittäinen tavoite
- Projektitiimi toimii parhaiten, kun sillä on ”near-term”, toteuttamiskelpoiset ja tunnistettavissa olevat arvokkaat tavoitteet
- Asiakkaat ovat tyytyväisimmillään, kun he tuntevat kontrolloivansa sitä, mitä kehitetään ja näkevät säännöllisesti todellisia tuloksia
- Lyhyet ja säännölliset palautesilmukat ovat tarpeellisia sekä ohjelmistoprojektin mittaamisen että etenemisen ohjauksen kannalta
- Prosessien virtaviivaisuus sekä automatisointi (sekä hallinnan että ohjelmoinnin toimintojen) antaa ihmisille vapauden tehdä arvokkaampaa ja mielenkiintoisempaa työtä.

Ketteryydessä voidaan siis todeta olevan kyse ihmisten ja heidän välisen vuorovaikutuksen tärkeydestä sen sijaan, että kukin toimisi itsenäisesti. Lähtökohtaisesti tulisi ajatella, että toimivan tiimin muokkaaminen ympäristöön sopivaksi on ketteryydessä merkittävää – ei ympäristön muokkaaminen. Ketterässä kehityksessä ei ole pyrkimyk-

senä tuottaa yksityiskohtaista dokumentaatiota koskien järjestelmää ja koodia. Halutaan saada aikaan yksiselitteistä dokumentaatiota, joka kuvaa selkeästi järjestelmän sekä suunnitelmallisten päätösten perustelut. Pitkien, täsmällisten dokumenttien sijaan tiimin jäsenet halutaan osaksi projektia jatkuvalla läheisellä vuorovaikuttamisella tarkoituksena siirtää tietoa eteenpäin tiimin jäsenien keskuudessa. Ketterä ajattelu korostaa asiakasyhteistyötä ja toimittajan sekä asiakkaan välistä alituista vuorovaikutusta.

Ketterä kehitys sai alkunsa, kun 1990-luvun lopulla lukuisiin ohjelmistokehityksen menetelmiin alkoi kohdistua kasvavassa määrin julkisuutta. Kussakin näistä oli erilainen yhdistelmä vanhoja ideoita, uusia ideoita ja muunneltuja vanhoja ideoita. Kaikki nämä metodologiat kuitenkin painottivat läheistä yhteistyötä ohjelmoijien ja liiketoiminta-asiantuntijoiden välillä sekä kasvokkain tapahtuvaa kommunikaatiota kirjoitettua dokumentaatiota tehokkaampana. Metodologiat korostivat uuden järjestäytyneen liiketoiminnallisen arvon yleistä tuottamista, tiiviitä, itsestään ohjautuvia tiimejä, sekä keinoja, joilla koodin ja tiimin taitava toiminta järjestäytyisi parempaan suuntaan (Agile Alliance 2001). Ketterä ohjelmistokehitys nähdään vastavetona kömpelöille prosesseille, tehtävänään uudistaa tietokoneohjelmointi ohjelmistotuotannoksi ja muuntaa se helppohoitoiseksi, odotustenmukaiseksi, kuten mikä tahansa muu tekniikan tieteenala. Ketterä ajattelu on uudenlaista, koska sen takana seisovat ihmiset ovat kokemustensa perusteella todistaneet listaamansa ketterät käytännöt, soveltaneet niissä ihmislähtöisiä ydinarvoja sekä projektiympäristöjä ja osoittaneet todellisin sekä arvokkain keinoin, kuinka niiden avulla lähestytään parempaa ohjelmistojen rakentamista. (Schuh 2005: 2.)

3.2 The Agile Alliance ja Agile Manifesto

Perinteistä lähestymistapaa, joka perustuu vesiputousmallin kaltaiseen vaihejakoon, pidetään raskaana, monien osakokonaisuuksien summana (Awad 2005: 2). Loputtomasti laajuudeltaan kasvaneet, kehityksensä puolesta paikalleen pysähtyneet prosessit ohjelmistotiimeissä ajoivat vuonna 2001 ryhmän ohjelmistokehityksen alan asiantuntijoita yhteen hahmotellakseen arvot ja periaatteet, jotka sallisivat ohjelmistotiimien toimia nopeasti ja alan muutokseen vastaten. Nämä asiantuntijat kutsuivat ryhmäänsä nimellä *the Agile Alliance*. Ja seuraavien kuukausien aikana he loivat pohjan ketterälle ajattelulle – tuloksena syntyi *The Manifesto of the Agile Alliance* eli ketterän ohjelmis-

tokehityksen julistus, jota pidetään ketterän ajattelun peruskivenä ja joka tiivistää parhaiten arvot ketterän kehityksen takana. Julistuksen sisältö on esitelty alla. (The Agile Alliance 2001.)

Helmikuussa 2001, seitsemäntoista eri ketterien menetelmien edustajaa – agilistia – muodostivat ketterän allianssin – Agile Alliancen – edistääkseen näkemyksiään, joka johti ketterän ohjelmistokehityksen julistuksen – the Agile Manifeston – syntyyn. Monia ketteriä tekniikoita on käytetty jo ennen allianssin syntyä, muttei perustuen allianssin näistä monista tekniikoista kokonaisuutena muodostamaan toimivaan viitekehykseen. (Agile Alliance 2001.)

Agilistien kunnioittamat keskeiset arvot on esitetty seuraavassa (Agile Alliance 2001.):

”Löydämme parempia tapoja tehdä ohjelmistokehitystä, kun teemme sitä itse ja autamme muita siinä. Kokemuksemme perusteella arvostamme:

Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja
Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota
Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja
Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa”

The Agile Manifestossa julistetut 12 periaatetta ovat seuraavat (Agile Alliance 2001.):

- Tärkein tavoitteemme on tyydyttää asiakas toimittamalla tämän tarpeet täyttäviä versioita ohjelmistosta aikaisessa vaiheessa ja säännöllisesti.
- Otamme vastaan muuttuvat vaatimukset myös kehityksen myöhäisessä vaiheessa. Ketterät menetelmät hyödyntävät muutosta asiakkaan kilpailukyvyn edistämiseksi.
- Toimitamme versioita toimivasta ohjelmistosta säännöllisesti, parin viikon tai kuukauden välein, ja suosimme lyhyempää aikaväliä.
- Liiketoiminnan edustajien ja ohjelmistokehittäjien tulee työskennellä yhdessä päivittäin koko projektin ajan.
- Rakennamme projektit motivoituneiden yksilöiden ympärille. Annamme heille puitteet ja tuen, jonka he tarvitsevat ja luotamme siihen, että he saavat työn tehtyä.

- Tehokkain ja toimivin tapa tiedon välittämiseksi kehitystiimille ja tiimin jäsenten kesken on kasvokkain käytävä keskustelu.
- Toimiva ohjelmisto on edistymisen ensisijainen mittari.
- Ketterät menetelmät kannustavat kestävään toimintatapaan. Hankkeen omistajien, kehittäjien ja ohjelmiston käyttäjien tulisi pystyä ylläpitämään työtahtinsa hamaan tulevaisuuteen.
- Teknisen laadun ja ohjelmiston hyvän rakenteen jatkuva huomiointi edesauttaa ketteryyttä.
- Yksinkertaisuus – tekemättä jätettävän työn maksimointi – on oleellista.
- Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat syntyvät itseorganisoituvissa tiimeissä.
- Tiimi tarkastele säännöllisesti, kuinka parantaa tehokkuuttaan, ja mukauttaa toimintaansa sen mukaisesti.

Yksi agilisteista – Robert Martin (2002: 8) tiivistää ketterän ohjelmistokehityksen julistuksen arvot nimenomaan prosessien kykyyn tuottaa asiakkaalle lisäarvoa ja asiakastyytyväisyyden maksimointiin perustuen. Hänen mukaan jokaisen ohjelmistokehittäjän ja jokaisen kehitystiimin ammattimainen tavoite on toimittaa korkein mahdollinen määrä arvoa työntekijöilleen sekä asiakkailleen. Martin näkee prosessiajattelun hyödyntämisen huomattavan kasvun olevan ainakin osittain syyllinen aiempien projektien epäonnistumisiin sekä niiden heikkoon arvontuottamiskykyyn. Hänen mukaansa ketterän ajattelun periaatteet ja arvot ovat perustuneet juuri prosessiajattelun vastavedoksi – ne on luotu keinoksi auttaa ohjelmistotiimejä rikkomaan jatkuvaa projektilaajuuden kasvua ja keskittymään yksinkertaisiin tekniikoihin tavoitteiden saavuttamiseksi.

3.3 Ketterän ja perinteisen ohjelmistokehityksen eroavaisuudet

Tässä alaluvussa käydään läpi yleisen tason eroja perinteisen, vaihejakoon perustuvan lähestymistavan ja ketterän lähestymistavan välillä. Näiden lähestymistapojen välillä on paljon yhtäläisyyksiä, mutta erittäin oleellisia eroavaisuuksia. Merkittävimpiä eroavaisuuksia on käsitelty seuraavassa perustuen lähinnä Aitkenin ja Ilangon (2013), Wellsin, Dalcherin ja Smythin (2015) sekä Awadin (2005) tutkimuksiin.

Aitken ja Ilango (2013) mainitsevat perinteisen ja ketterän lähestymistavan välisten eroavaisuuksien liittyvän erilaisiin ominaisuuksiin, kuten käytettyjen mallien vaihtelevuuteen, mallien käyttötarkoitukseen sekä mallintamisen lähtökohtiin. Lähtökohtaisesti lähestymistapoja ei kuitenkaan nähdä täysin ristiriitaisina, eikä tulevaa nähdä pelkän ketterän lähestymistavan aikana. Awad (2005) nimittää perinteistä lähestymistapaa ja perinteisiä menetelmiä raskaansarjan menetelmiksi, jotka perustuvat kattavaan suunnittelutyöhön, yksityiskohtaiseen dokumentaatioon ja laajaan suunnitelmaan. Raskaansarjan menetelmille Awad kuvaa vaihtoehtoisen, kevyemmän menetelmän – ketteryyden. Ketterän lähestymistavan kulmakiviksi hän mainitsee lyhyet iteratiiviset syklit, ja ketterän tiimin jäsenten hiljaiseen tietoon nojautumisen dokumentaation sijasta. Alla olevassa taulukossa (Taulukko 2) on listattuna Awadin näkemys merkittävimmistä eroista perinteisen ja ketterän lähestymistavan välillä.

	KETTERÄ	PERINTEINEN
LÄHTÖKOHTA	Mukautuva	Ennaltamäärätty
ONNISTUMISEN MITTAUSPERUSTE	Liiketoiminnallinen arvo	Suunnitelmanmukaisuus
PROJEKTIKOKO	Pieni	Suuri
JOHTAMISTYYLI	Hajautettu	Autokraattinen
SUHTAUTUMINEN MUUTOKSEEN	Muutokseen mukautuva	Muutosvastainen
KULTTUURI	Johto on osa tiimiä	Johto selkeästi hierarkiassa ylempänä
DOKUMENTAATIO	Vähäistä	Raskasta
PAINOTUS	Ihmislähtöinen	Prosessiorientoitunut
SYKLIT	Useita	Yksittäinen
TOIMINTAYMPÄRISTÖ	Ennalta määrittelemätön/ Tutkiva	Ennalta määritelty
ENNAKKOSUUNNITTELU	Minimoitua	Korostettua
SIJOITETUN PÄÄOMAN TUOTTO	Projektin alusta alkaen	Projektin lopussa
TIIMIKOKO	Pieni	Suuri

Taulukko 2. Ketterän ja perinteisen lähestymistavan merkittävimmät eroavaisuudet osa-alueittain. (Awad 2005.)

Awadin (2005: 35) mukaan sekä raskailla että ketterillä menetelmillä on molemmilla vahvuuksia ja heikkouksia, mutta perimmäinen valinta näiden välillä tulisi perustua toteutettavan projektin kokoon, projektissa toimiviin ihmisiin sekä projektiin liittyviin riskeihin. Awad näkeekin juuri projektikoon olevan yksi rajoittavista tekijöistä ketteryyden projektikohtaista käytettävyyttä arvioidessa, koska perinteisen lähestymistavan hän kuvaa soveltuvan budjetin, projektitiimin koon ja keston mittapuulla suuriin projekteihin. Awad kuvaa ketteryyden yhdeksi lähtökohdaksi taas nimenomaan sen, että pienemmissä, ketterissä tiimeissä työskennellessä voidaan saavuttaa kaikkien ketterien arvojen mukaista etua. Hän lisää, että lähtökohtainen ratkaistava ongelma ketterissä projekteissa on myös kooltaan pienempi, jota on korostanut muiden muassa Alistair Cockburn, yksi alkuperäisistä agilisteista. Yhden käytetyimmän ketterän menetelmän, Scrumin, kehittäjä Ken Schwaber (2004: 8) ei näe ratkaistavan ongelman suuruuden

tai projektitiimin koon vähentävän toiminnan tehokkuutta. Hän korostaa itseorganisoituvaa tiimityötä ja nimenomaan tiimin kykyä tehdä ratkaisuja tilanteessa, jossa esimerkiksi tiimin suuruus nähdään ongelmaksi. Tällöin tiimin tulisi lähtökohtaisesti ohjata itse itsensä pienempiin osiin ja jatkaa projektityötä eteenpäin uudenlaisena. Myös Aitken ja Ilango (2013: 4758) nostavat merkittävänä eroavaisuutena perinteisen ja ketterän lähestymistavan välillä projektijohdon – perinteisessä ohjelmistoprojektissa on projektipäällikkö, kun taas ketterässä projektissa vastuu projektista on itse tiimillä ja sillä on valmentajan tai masterin avulla kyky toimia itsenäisesti. Kestollisesti perinteinen lähestymistapa on huomattavasti ketterää pidempi menetelmä johtuen erityisesti kunkin vaiheen dokumentoinnista, kun taas ketteryys perustuu lyhyisiin iteraatioihin ja samanaikaisiin prosesseihin. Aitken ja Ilango mainitsevat dokumentoinnin olevan perinteisissä menetelmissä vastine ketterässä mallissa syntyvälle koodille ja toki koodia toteutetaan myös perinteisessä lähestymistavassa, mutta dokumentointia tehdään huomattavasti enemmän. Alla olevassa taulukossa (Taulukko 3) on esitelty Aitkenin ja Ilangon näkemys eroavaisuuksista perinteisten sekä ketterien menetelmien välillä mukaillen ketteriä arvoja ja periaatteita. (Schwaber 2004: 8; Aitken & Ilango 2013: 4758.)

KETTERÄ LÄHESTYMISTAPA	PERINTEINEN LÄHESTYMISTAPA
Ihmis- ja yhteistyölähtöinen	Prosessi- ja työkalulähtöinen
Koodiin ja ohjelmointiin perustuva. Tuotettu toimiva ohjelmisto on onnistumisen mitta. Korostaa koodin ulkoasun ja selkeyden tärkeyttä.	Dokumentteihin perustuva. Jokaista toimintoa verrataan/ mitataan dokumentein tai toimituskelpoisuuden mukaan.
Asiakkaat ovat mukana jatkuvasti.	Asiakkaat eivät ole mukana jatkuvasti.
Vaatimusmuutokset mahdollisia myös projektin edetessä ja myöhemmin projektin aikana, mikä mahdollistaa myös ”suunnanmuutokset”.	Vaatimusmuutoksia ei suosita projektin etenemisen aikana. Suuntaa on vaikea muuttaa sopimusten syntymisen jälkeen.
Toimitetaan asiakkaalle jatkuvasti toimivaa ohjelmistoa.	Ei suosi ohjelmiston jatkuvaa toimitusta asiakkaalle.
Syklien maksimikesto 2 kuukautta, usein ohjelmaa toimitetaan asiakkaalle jopa alle 2 viikon välein.	Lyhyitä iteraatioita ei suosita.
Ohjelmisto toimitetaan parhaalla mahdollisella tavalla yhdessä toimivien tiimien toimesta.	Ei suosi itseorganisoituja tiimejä.
Vaiheita ei noudateta muodollisesti. Erillistä analysointi –tai suunnitteluvaihetta ei ole.	Ohjelmistokehityksen elinkaaren vaiheita noudatetaan järjestelmällisesti, analysointia ja suunnitelmallisuutta suositaan.
Ohjelmisto rakennetaan lyhyissä pätkissä, yksittäisinä ominaisuuksina tai ”tarinoina”, joita voidaan kehitellä vaiheittain.	Jokainen määrittelyvaiheessa määritetty vaatimus tulee olla toteutettu lopullisessa tuotteessa.
Kehittäjien ja asiakkaiden tai käyttäjien tulisi tavata säännöllisesti, jotta kehitys kestää läpi projektin.	Kehittäjät ja asiakkaat ovat tekemisissä vain projektin alussa ja lopussa.

Taulukko 3. Ketterän ja perinteisen lähestymistavan eroavaisuudet periaatteittain. (Aitken & Ilango 2013.)

Jo pelkästään Agile Manifeston (2001) sisältöä tarkkaillessa, voidaan huomata ihmisten olevan poikkeuksellisen merkittävässä asemassa ketterässä kehityksessä ja kehitysprojektiin kuuluvat ihmiset sekä näiden merkitys juuri erottaa ketterän ajattelun perinteisestä lähestymistavasta. Awad (2005: 37) korostaa ihmisissä erityisesti näiden taitoja sekä kokemusta ja näiden merkitystä ketterässä lähestymistavassa. Tiimin koostuessa ihmisistä, joilla on useista eri tehtävistä ja tilanteista kokemusta, on tiimin sisällä helppo antaa palautetta ja huomata mahdollisia virheitä jo toiminnan aikana. Toinen merkittävä ero on asiakkaan läsnäolo projektissa. Tämä aspekti on erittäin merkittävä osa ketterää ajattelua ja uutta perinteiseen nähden. Suurin etu saavutetaan sillä, että asiakas voi tarkkailla koko projektin ajan sen edistymistä ja jokaisen iteraation aikana on asiakkaan toiveisiin perustuen mahdollista muuttaa kehitysprojektin suuntaa tai tuoda siihen uusia piirteitä. Awadin mukaan organisaatiokulttuuri on myös merkittävässä asemassa valittaessa käytettävää menetelmää. Raskaampi, vaihejakoon perustuva perinteinen lähestymistapa saattaa olla parempi vaihtoehto organisaatioissa, joissa ympäristömuutokseen reagointi ei yksinkertaisesti ole mahdollista tai vaatii huomattavia ponnisteluja. Lukuisiin rajoitteisiin, sääntöihin ja käytäntöihin pohjautuva organisaatiomalli, joka ei suoranaisesti ole vastuussa muutoksesta tai jonka ei välittömästi ole pakko reagoida sitä ympäröivään muutokseen, ei toimi ketterässä lähestymistavassa. (Awad 2005: 37.)

Awadin mukaan suurimmat riskitekijät ohjelmistokehityksessä ovat projektien kriittisyyden ja muutoslähtöisyyden tasot. Nopeasti rakennettavat ja vähäisempää laadun varmistusta vaativat sovellukset ovat ketterään kehitykseen parhaiten sopivia. Raskaansarjan menetelmiä taas tulisi suosia projekteissa, joiden kohteena ovat hyvin kriittiset, käyttövarmat ja turvallisuuteen perustuvat järjestelmät, koska tällöin toteutetaan vaatimukset pitkään harkiten ja perusteellisesti ennen toteutukseen ryhtymistä. Muutoslähtöistä ajattelua korostavissa projekteissa on helppo valinta toteuttaa kohde ketterällä menetelmällä. Muutoksiin on huomattavasti helpompi reagoida asiakkaan ollessa läsnä ja iteraatioiden ollessa lyhyempiä. Awad tiivistää riskien huomioimisen siten, että mitä poikkeuksellisemmat projektin olosuhteet ovat, sitä selkeämpi on perinteisen tai ketterän lähestymistavan valinta käytettäväksi. (Awad 2005:37.)

3.4 Ketteryys ja projektihallinta

Edellä kuvatun perusteella voidaan selkeästi tulkita ketterän ohjelmistokehityksen eroavan perinteisestä menetelmästä. Ketterää menetelmää valitessa on syytä tuntea myös toteutettava projekti ja ketterän menetelmän valinta tulee olla perusteltua. Edeltäviä, perinteisen ja ketterän lähestymistavan eroja tukien, myös Peter Schuh (2005) listaa erot ketterän ja perinteisen lähestymistavan välillä sekä korostaa ketterän tavan vahvuuksina kehitystiimin kommunikaatiota ja vastuuta, läpi projektin jatkuvaa joustavaa suunnittelua, asiakkaan läsnäoloa, monipuolista joustavuutta prosessien ja työkalujen valinnassa.

Schuhin (2005: 10) mukaan ketterä tiimi vaatii tarkoituksenmukaisesti toimiakseen projektihallinnan toteutuksen niin, että se vastaa tiimin täydellisen toiminnan mahdollistaviin tarpeisiin ja tiimin intresseihin perustuen tuottaa säännöllisen palautteen mahdollistavia mekanismeja, jotka taas tukevat projektin etenemistä ja prosessien ketteryyttä. Projektijohdon tulee priorisoida juuri hallintalähtöisiä ketteriä käytäntöjä, kuten lyhyitä iteraatioita, tiheään tapahtuvia julkaisuja sekä tasaisen etenemistahdin ylläpitoa ja näiden hyödyntäminen perustuu projektijohdon myötävaikutukseen sekä yhteistyöhön. Käytettävästä menetelmästä riippuen tiimin vastuun ja hallinnan tarve vaihtelee, kuitenkin tiedostaen, että tiimin läsnäolo päätöksenteossa on ketterille menetelmille poikkeuksetta ominaista. Yleisellä tasolla Schuh kuvaa, että agilistit ovat yksimielisiä siitä, että voidakseen toimia ketterästi, projektijohdon on kuunneltava tiimiä ja vastattava sen tarpeisiin. Lyhyet iteraatiot ovat esimerkki edellä kuvatuista palautteen mahdollistavista mekanismeista, koska ne mahdollistavat kokonaisvaltaisen edistymisen seuraamisen, kun palautetta saadaan säännöllisesti ja lyhyin väliajoin. Tällöin tiimi saa selkeän kuvan siitä, toimivatko hyödynnetyt käytännöt, lakkaako jokin projektiin kuuluva käytäntö toimimasta sekä kykeneekö tiimi vastaamaan tehokkaimmin keinoin vaatimus –tai ympäristömuutoksiin.

Ketterässä ohjelmistokehityksessä asiakkaan läsnäolo on yksi merkittävimmistä arvoista. Schuhin (2005: 11) mukaan ketterässä projektissa asiakas on jokin tiimin helposti lähestyttävissä oleva taho –ei välttämättä yksittäinen henkilö, joka tuntee projektin, vaan esimerkiksi ryhmä asiantuntijoita, jolla on valta tehdä projektia koskevia päätöksiä ja joka kykenee puhumaan yksimielisesti edustamansa organisaation puolesta, jolle projektin lopputulos aiheuttaa etua. Asiakkaan halutaan olevan mukana ketterässä

projektissa sekä toiminnallisista että sananmukaisesti ketteristä syistä. Toiminnallisesti katsottuna on luonnollista, että projektin lopputuloksena syntyy parempi tuote, kun mukana on yhtäläisellä panoksella ja tavoitteilla varustettu sekä tietämyksen että intressin omaava osapuoli. Toisekseen ketterä projekti vaatii asiakkaan jatkuvaa läsnäoloa, jotta se voidaan toteuttaa ”juuri oikeaan tarpeeseen” -periaatteella *(engl. just-in-time*i. Tämä tarkoittaa sitä, että toimitetaan vain tarvittavia tuotteita niitä tarvitsevalle asiakkaalle, vasta silloin kun niitä tarvitaan. Tarkemmin tarkasteltuna tulee huomioida myös se, että projektin edetessä elinkaarellaan, tehdään jatkuvasti suunnitteluun ja vaatimuksiin liittyviä päätöksiä, jotka vaativat asiakkaan sitoumusta ja osallistumista. Asiakkaan läsnäolo on menetelmäkohtaista, mutta ketteryydessä asiakasyhteisyössä korostetaan asiakkaan kanssa päivittäin tapahtuvia neuvotteluita tai yksilöitä, jotka istuvat ja työskentelevät jatkuvasti tiimin kanssa samassa tilassa. Menetelmästä riippumatta on tavoite kuitenkin aina sama – helposti tavoitettava ja kommunikointia arvostava asiakas tarkoittaa, että vaatimuksia voidaan parannella ja jalostaa milloin tahansa. Ei kuitenkaan voida olettaa asiakkaan heti olevan osa tiimiä ja ketterää projektia, vaan perehdyttäminen ja osaksi tiimiä sekä ketterää lähestymistapaa hioutuminen vie aikansa. Lopulta tavoitteena on taata, että asiakas tuntee luonnollisesti olevansa läsnä uusien toimintojen määrittelyssä, kuvaamisessa ja suunnittelussa. (Schuh 2005:11.)

Ketterässä lähestymistavassa pyritään korostamaan ”vain välttämätön” -periaatetta. Tämä näkyy muiden muassa projektissa hyödynnettävien prosessien ja työkalujen valinnassa – projektitiimin tulee hyödyntää niitä prosesseja ja työkaluja, jotka tuottavat tarkoituksenmukaisesti tarkastuksia ja rakennetta pitäen niiden määrän rajallisena ja tavoiteltuna, ei yhtään laajempana. Tiimi voi esimerkiksi päättää, että tuotettua koodia tarkistetaan useita kertoja päivässä koko tiimin toimesta, eikä koodi leviä useita päiviä eri suuntiin useille yksittäisille työasemille jätettynä. Mikä tahansa prosessi tai työkalu, joka ei ole tiimin yhteisesti hyväksymä tai toimii vastoin ”vain välttämätön” -periaatetta, on tarpeetonta. Ketterässä lähestymistavassa on nimenomaan kyse siitä, että kaikki tämä tarpeeton halutaan ajoissa pyrkiä karsimaan projektin ulkopuolelle. Tästä tyypillinen esimerkki on dokumentaatio, josta ketterät toimijat pyrkivät hankkiutumaan eroon. (Schuh 2005: 11-12.)

Schuh (2005: 13) mainitsee, että mikä tahansa projekti voi hyötyä ketterästä kehityksestä. Tutkittaessa jotain ketteryyteen pyrkivää projektia, jonkin tietyn piirteen tai käytännön puuttuminen saattaa määrittää suunnan sille, mitä menetelmää tai menetelmiä

projektissa käytetään tai jonkin tietyn menetelmän valitseminen käytettäväksi tietyssä ketteryyteen pyrkivässä projektissa saattaa määrittää mitä piirteitä siinä tulisi esiintyä. Tällainen tilanne, jossa menetelmän valinta tai ketterien piirteiden määrittäminen on epäselvää, on haasteellinen, muttei estä projektin toteuttamista ketteränä. Tällöin tiimin tulee kyetä hyödyntämään ketteryyttä. Tällaisessa tilanteessa olevalla projektitiimillä on kaksi vaihtoehtoa. Ensinnäkin projektissa voidaan pyrkiä muuntamaan sen ympäristöä. Riippuen projektin yksityiskohdista tämä voi tarkoittaa kääntymistä sidosryhmien puoleen, keskustelun alulle panemista korkeamman tason johdon kanssa tai pyrkimystä muuttaa suhdetta asiakkaaseen. Mikäli nämä eivät tunnu toimivilta keinoilta, projekti tulee joka tapauksessa pitkällä aikavälillä katsottuna hyötymään ympäristön viemisestä kohti ketterämpää ajattelutapaa ja lähtökohtaa. Toinen vaihtoehto on ottaa käyttöön ketteriä käytäntöjä vaiheittain ja opportunistisemmasta lähtökohdasta. Jos projektiympäristöä ei kyetä muokkaamaan, ei sitä voi kutsua ketteräksi. Silti on mahdollista omaksua hiljalleen, määrätietoisesti ja harkiten ketteriä käytäntöjä.

Edellä kuvatussa pyritään selittämään sitä, että projekti hyötyy jopa yhden yksittäisen ketterän käytännön tai piirteen omaksumisesta. Tässä tulee huomioida, että tuon yhden yksittäisen käytännön tulee olla harkiten valittu, tiimin hyväksymä ja sen vaikuttamiselle ja noudattamiselle tulee antaa tarpeellinen määrä aikaa. Schuhin mukaan tällaisen käytännön omaksuminen ei ole yksiselitteistä ja se voi vaatia useamman yrityksen, mutta lopulta, kun tiimi antaa jalan sijaa uusille toimintatavoille ja omaksuu ne hiljalleen, huomaa se lopulta joustamattomien rajoitusten katoavan ympäristöstään. Ketterien käytäntöjen oikeanlainen omaksuminen ja ketteriin arvoihin pyrkiminen voi saada aikaan merkittäviä tuloksia ympäristöjen valikoimassa. Ei-ketterissä ympäristöissä toimiminen voi tehdä ketterien käytäntöjen omaksumisesta vaikeaa, vähentää projektitiimin tuottavuutta, uhata moraalia ja vaikeuttaa merkittävästi ketteryyden oppimista. Schuh kuitenkin kuvaa ketteryyden aina vievän projektin pitkällä aikavälillä parempaan suuntaan. (Schuh 2005: 13-14.)

Schuhin (2005) näkemykset edellä kuvaavat ketterän projektinhallinnan oikeanlaisen toteuttamisen edellyttämiä projektien sisältämiä käytäntöjä. Chow ja Cao (2007) ovat kyselytutkimuksellaan tunnistaneet ja koonneet yhteen kriittisimmät menestystekijät ketterissä ohjelmistokehitysprojekteissa. Heidän havaintonsa tukevat Agile Manifestossa esitettyjä ketteriä periaatteita: kolme merkittävintä menestystekijää ovat oikeanlainen toimitusstrategia, ketterien tekniikoiden sopiva hyödyntäminen sekä tiimin tai-

dollinen kyvykkyys ja nämä kaikki kolme muodostuvat pitkälti Agile Manifeston sisällöistä. Myös kolme muuta tekijää – hyvä sekä ketterä projektijohto, ketteryysystävällinen toimintaympäristö ja asiakkaan voimakas läsnäolo – mukailevat Agile Manifeston sisältöä. Taulukossa 4 on kuvattu kyseiset menestystekijät ominaisuuksineen. (Chow & Cao 2007: 968.)

Menestystekijä	**Ominaisuudet**
Toimitusstrategia	ohjelmistoa tulee toimittaa säännöllisesti tärkeimmät tuoteominaisuudet tulee toimittaa ensisijaisesti
Ketterät tekniikat	ohjelmiston rakenteen tulee olla yksinkertainen dokumentaatiota tulee tuottaa sopiva määrä
Tiimin kyvykkyys/taidot	tiimin tulee olla asiantunteva ja motivoitunut johdolla tulee olla tietämystä ketteryydestä johdon tulee olla toimintatavoiltaan sopeutuva tiimi tulee kouluttaa asianmukaisesti projektin alussa
Projektijohto	vaatimusmäärittely tulee olla ketterästi toteutettu projektihallinta tulee olla ketterää projektin etenemistä tulee seurata selkein mekanismein kommunikoinnin tulee olla voimakasta projektin tulee edetä tasaisesti ja esteittä
Toimintaympäristö	koko tiimin tulee toimia samassa lokaatiossa tiimin tulee olla itseorganisoitunut tiimin tulee olla kooltaan pieni tiimin tulee olla yksittäinen kokonaisuus
Asiakkaan läsnäolo	asiakassuhteen tulee olla luonteeltaan positiivinen asiakkaan tulee olla voimakkaasti läsnä ja vaikuttaa päätöksiin asiakkaalla tulee olla täysi päätösvalta

Taulukko 4. Tärkeimmät menestystekijät ketterässä ohjelmistokehitysprojektissa ominaisuuksineen (Chow & Cao 2007).

4 KETTERIÄ MENETELMIÄ

Ketterä lähestymistapa ohjelmistokehityksessä perustuu ketteriin menetelmiin, joita on viime vuosikymmeninä kehitetty ja niiden alkuperäisideoita jalostettu pidemmälle. Tänä päivänä käytössä on lukuisia erilaisia menetelmiä, joilla kuitenkin voidaan nähdä olevan hyvin paljon yhteisiä piirteitä. Tässä luvussa esitellään viimeisellä vuosikymmenellä yleisimmin käytössä olleet sekä tämän tutkielman kannalta merkittävimmät ketterät menetelmät: ketterän lähestymistavan esittelemisen kannalta merkittävä eXtreme Programming (VersionOne: 2015) sekä tutkimusaineistossa merkittävää roolia näyttelevät Scrum ja Kanban.

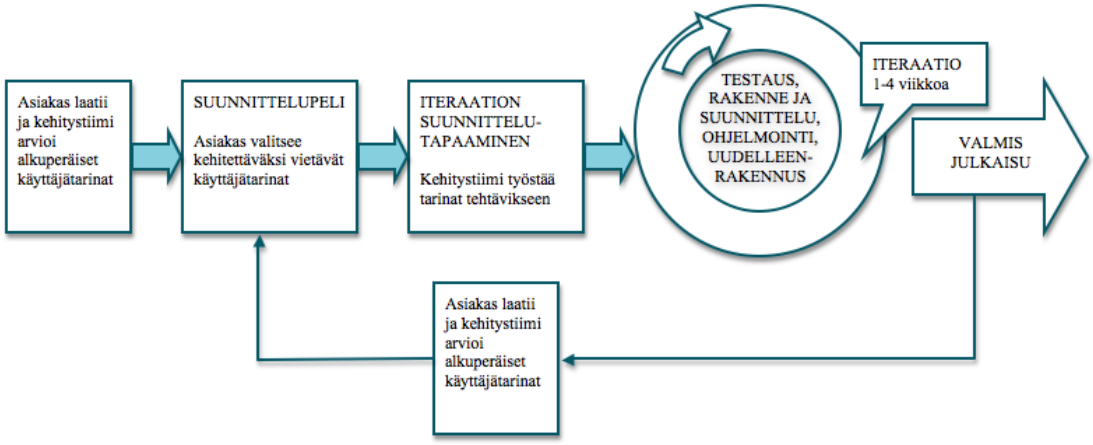
4.1 Extreme Programming (XP)

Extreme Programming – jatkossa XP (lyh. eXtreme programming) – on ketterä ohjelmistokehitysmenetelmä, jonka on kehittänyt Kent Beck. Kuten alaluvussa 4.2 esiteltävä Scrum, on XP:kin iteratiivinen menetelmä. XP:n perusideologia on hyvin yhteneväinen tutkielmassa aiemmin esitettyihin Agile manifeston arvoihin ja käytäntöihin. XP:ia on pidetty tunnetuimpana ja laajimmin käytettynä ketteränä menetelmänä. Nykyisin Scrum on menetelmistä käytetyin, mutta XP on yksi ensimmäisenä käyttöön omaksutuista menetelmistä (VersionOne 2015). Poikkeuksellisen XP:sta tekee se, että sitä on pidetty ainoana ketteränä menetelmänä, joka keskittyy ensisijaisesti ohjelmistokehityksen koodauspuoleen (Schuh 2005: 19). Robert C. Martinin (2002) näkemyksen mukaan XP on poikkeuksellinen käytäntöjensä vuoksi:

> ”Extreme Programming on ketteristä menetelmistä tunnetuin. Se koostuu joukosta yksinkertaisia, toisistaan riippuvaisia käytäntöjä. Nämä käytännöt toimivat yhdessä muodostaakseen kokonaisuuden, joka on sen yksittäisiä osia osioita eheämpi.” (Martin 2002: 11).

XP:in suosio perustuu sen painottamaan asiakastyytyväisyyteen – sen sijaan, että toimitettaisiin asiakkaalle kaikki mahdollinen yhdessä paketissa pitkällä tulevaisuudessa, toimitetaan XP:ssa tarvittu ohjelmisto tarvitunlaisena. XP:ssa pystytään vastaamaan muuttuviin asiakasvaatimuksiin myös ohjelmiston elinkaaren loppuvaiheessa. XP painottaa tiimityötä – johtajat, asiakkaat ja kehittäjät ovat kaikki tasavertaisia kumppaneita yhteistyössä toimivassa tiimissä. Kehitystiimi on XP:ssa jatkuvasti yhteydessä

asiakkaaseen ja se kommunikoi koko ajan keskenään. XP:ssa korostuu ohjelmistotuotteiden toimitus asiakkaalle niin aikaisin kuin mahdollista, jolloin sitä voidaan parannusehdotusten mukaan kehittää edelleen. XP koostuu pienemmistä osista, jotka yhdessä muodostavat eheän kokonaisuuden – taustalla ovat *XP-arvot –ja käytännöt*, jotka luovat perustan *XP-säännöille*. Kuviossa 4 on kuvattu XP- projektin sisältö yleisellä tasolla. (Wells 2013.)



Kuvio 4. Extreme Programming -projektin sisältö kuvattuna yleisellä tasolla (Schuh 2005: 21)

Seuraavissa alaluvuissa käydään läpi XP:n neljä perusarvoa ja niiden välisiä suhteita, XP:n sääntöjä, roolitusta XP -projekteissa sekä yksityiskohtaisemmin XP -projektin elinkaarta.

4.1.1 Arvot ja käytännöt

XP perustuu neljään perusarvoon. Ensimmäinen on kommunikaatio, jota XP:ssa halutaan korostaa kommunikaatiota vaativilla käytännöillä. Tällaisia käytäntöjä ovat muiden muassa yksikkötestaus, parikoodaaminen tai tehtävien arviointi. Tällaisissa tilanteissa ohjelmoijien, asiakkaiden ja johdon on kommunikoitava.

Toinen arvo on yksinkertaisuus, joka XP:ssa tarkoittaa tietynlaista varovaisuutta. Ajatellaan, että jonkin todella monimutkaisen toiminnon sijaan, jota ei mahdollisesti tulla koskaan käyttämään, on parempi tehdä jokin yksinkertainen toiminto, jota voidaan

tarvittaessa muuttaa tai viedä monimutkaisempaan suuntaan. Tästä myöhemmästä muutoksesta ollaan myös tarvittaessa valmiita maksamaan. Yksinkertaisuus kulkee XP:ssa käsi kädessä kommunikaation kanssa – mitä enemmän kommunikoit, sitä selkeämmin tiedät mitä tulee tehdä ja mitä ei. Mitä yksinkertaisempi järjestelmäsi on, sitä vähemmän se vaatii kommunikointia, joka taas johtaa relevantimpaan kommunikointiin, jolloin yksinkertaisuus voi ulottua jopa pienempään vaaditun henkilöstön määrään.

Kolmas arvo on palaute. Sitä saadaan edelleen arvostetun kommunikaation myötä asiakkaalta ja kollegoilta. Ohjelmoijat saavat XP:ssa palautetta tekemällä testausta ja arviointia välittömästi toimeksiannon saatuaan, jolloin asiakkaat ja muu tiimi voi niin pian kuin mahdollista kommentoida toteutusta.

Neljäs arvo on Extreme Programming -menetelmässä rohkeus. Rohkeus viittaa tässä kohtaa uskallukseen tuoda esiin myös monimutkaisempia ideoita, jotka saattavat riskeerata koko tuotannossa olevan projektin. Joskus ideat monimutkaisimmatkin ideat kuitenkin toimivat ja niistä syntyy menestys. Kommunikaation myötä on mahdollista esittää rohkeimmatkin ajatukset, Yksinkertaisuuden korostaminen tukee rohkeutta, koska yksinkertaisella pohjalla, tiimillä on varaa olla rohkeampi ja sen on helpompi kokeilla rohkeita asioita yksinkertaisemmassa järjestelmässä. Palaute taas takaa niin sanotun turvallisuudentunteen uusia asioita kokeillessa ja tiimi voi rohkeammin kokeilla asioita saadessaan niistä palautetta. XP:ssa kehitystiimin määrä rohkeasti vastata muuttuviin vaatimuksiin ja teknologiaan (Wells 2013). Beck (1999: 32-36) on esittänyt myös viidennen arvon, kunnioituksen. Kunnioitus muita tiimiläisiä ja itse projektia kohtaan on perusta kaikille muille neljälle ja XP ei toimi ilman kokonaisuuden sisäistä kunnioitusta.

Beck (1999: 47-48) listaa kaksitoista XP:lle ominaista käytäntöä selityksineen:

- *Suunnittelupeli* (engl. the planning game) – Seuraava julkaisu tulee ennalta määritellä nopeasti vertailemalla liiketoiminnallisia prioriteetteja ja teknisiä arvioita, suunnitelmaa voidaan tarvittaessa myöhemmin päivittää
- *Pienet julkaisut* (engl. small releases) – Julkaisut tulee viedä yksinkertaisina nopeasti tuotantoon ja pienissä sykleissä julkaista uusia versioita
- *Järjestelmän metafora* (engl. metaphor) – Tuotettavasta järjestelmästä tulee jakaa kokonaisvaltainen, yksinkertainen tarina

- *Yksinkertainen rakenne ja suunnittelu* (engl. simple design) – Järjestelmä tulee suunnitella niin yksinkertaisena kuin mahdollista ja liika monimutkaisuus tulee karsia sitä esiintyessä
- *Testaus* (engl. testing) – Ohjelmoijat tekevät yksikkötestausta jatkuvasti ja sen tulee sujua virheettömästi kehittämistyön etenemiseksi. Asiakkaat testaavat vain valmiiksi saatettuja tuoteominaisuuksia
- *Uudelleenrakentaminen* (engl. refactoring) – Ohjelmoijat uudelleenrakentavat järjestelmää säilyttäen sen tarkoituksenmukaisuuden, poistaen kaksinkertaisuutta, edistäen kommunikaatiota ja yksinkertaisuutta sekä lisäämällä sen joustavuutta
- *Pariohjelmointi* (engl. pair programming) – Kaikki koodi tuotetaan pareittain – kaksi ohjelmoijaa yhdellä koneella
- *Koodin yhteisomistajuus* (engl. collective ownership) – Kuka tahansa voi muokata koodia missä tahansa järjestelmän osassa, missä tahansa vaiheessa
- *Jatkuva integrointi* (engl. continuous integration) – Järjestelmää työstetään monta kertaa päivässä, aina kukin vaihe valmiiksi kerrallaan
- *40-tuntinen työviikko* (engl. 40 hour week) – Työviikon pituus on rajoitettu 40:een työtuntiin
- *Paikan päällä oleva asiakas* (engl. on-site customer) – Tiimiin tulee kuulua lopullinen käyttäjä, jolta saada vastuksia milloin tahansa projektin aikana
- *Koodausstandardit* (engl. coding standards) – Ohjelmoijat koodaavat sääntöjen mukaisesti korostaen kommunikaatiota

XP:ssa yksikään edellä mainituista käytännöistä – testausta lukuun ottamatta – ei toimi yksinään. Ne vaativat ympärilleen muita käytäntöjä pysyäkseen tasapainossa. Arvojen ja käytäntöjen yhteenkuuluvuutta korostetaan tutkielman seuraavassa alaluvussa 4.1.2, joka kuvaa XP:n sääntöjä, jotka määrittelevät menetelmän työskentelytavat.

4.1.2 Säännöt

Don Wells (2013) kuvaa XP:n säännöt kaikkein yllättävimmäksi näkökulmaksi menetelmässä. Yksittäisinä monet pienet osaset eivät ole järkeenkäypiä, mutta yhdessä ne muodostavat selkeän kuvan. XP:n säännöt perustuvat edellä kuvattuihin arvoihin ja käytäntöihin. Alla on kuvattu projektin keskeisimpiä sääntöjä korostaen työskentelytapaa XP:ssa.

Suunnittelu on kaksijakoinen käytäntö XP:ssa – koko projekti alkaa suunnittelulla (engl. planning), jonka merkittävimpänä sisältönä asiakas laatii käyttäjätarinat sen perusteella, mitä järjestelmän tulee voida tarjota heille. Suunnittelun toinen ulottuvuus on järjestelmän rakenteen suunnittelu (engl. designing), jonka toteuttaa kehitystiimi. Sen merkittävimpiä sääntöjä ovat muiden muassa yksikertaisuuden korostaminen, järjestelmä metaforan valitseminen, koodin yhteisomistajuuden korostaminen käyttämällä ideointityökaluina CRC-kortteja (engl. collaboration cards), yksinkertaisten kohdesovellusten (engl. spike solutions) suosiminen riskien minimoimiseksi ja lopulta säännöt korostavat, ettei tässä (aikaisessa) vaiheessa mitään toiminnallisuutta lisätä tuotteeseen sekä pyritään uudistamaan kokonaisuutta tehokkaammaksi aina kun mahdollista.

Wells (2013) linjaa käytännön hallinnoinnin (engl. managing) säännöiksi avoimen työtilan antamisen kehitystiimille, tasaisen etenemisen korostamisen, kehitystyön aloittamisen päivittäin tapaamisella, projektin etenemisvauhdin mittaamisen sekä ihmisten tiiminsisäisen liikkumisen osaamisen varmistamiseksi. Kuten kaikissa ketterissä projekteissa, tulee myös XP:n tapauksessa kyetä projektin aikana omaksumaan käytettävää menetelmää ja soveltamaan sitä omaan ympäristöön sopivaksi tarpeellisin korjauksin.

Koodausta (engl. coding) koskevia sääntöjä ovat muiden muassa asiakkaan jatkuva läsnäolo, koodin tulee olla standardinmukaista, kaikki koodi tuotetaan pariohjelmointina, vain yksi pari integroi koodia kerrallaan ja integrointia tulee tehdä usein sekä koodi on aina yhteisomistuksessa.

Testauksen (engl. testing) säännöt ovat: kaikki koodi tulee yksikkötestata ja ennen sen julkaisemista, tulee koodin läpäistä yksikkötestit. Kun vikoja löydetään, luodaan sille tarvittavat testit. Hyväksymistestejä tehdään mahdollisimman usein ja niiden tulokset tulevat julkisiksi.

4.1.3 Roolit ja tiimit

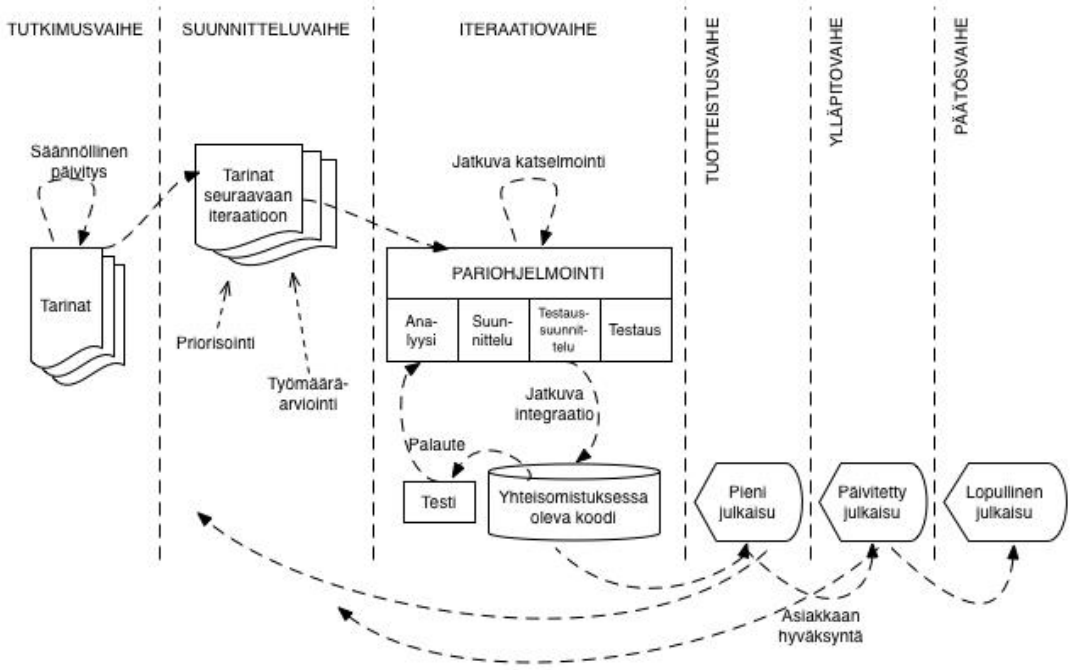
Extreme Programming toimii parhaiten pienissä kehitystiimeissä – vähintään kaksi ja enimmillään 12 on optimaalisin henkilömäärä tiimiä kohden, joskin laajimmissa projekteissa saadaan XP toimimaan tarkoituksenmukaisesti jopa 30 hengen tiimissä (Wells 2013; Schuh 2005: 22). Kuten edellä on jo kuvattu, toimii kehitystiimi asiakasta

myöden samassa tilassa ja kaikkien tahojen tulee olla aina tavoitettavissa. XP:ssa roolit ovat: ohjelmoija, asiakas, testaaja, mittaaja, valmentaja, konsultti ja johtaja. XP:ssa on ominaista, että yksi henkilö voi toimia useammassa roolissa ja kaikki ovat yhteydessä asiakkaaseen. (Beck 1999: 105-111.)

XP:ssa *ohjelmoija* (engl. programmer) arvioi ja kehittää toimintoja asiakkaan laatimista käyttäjätarinoista ja yhdessä ohjelmoijat luonnollisesti kirjoittavat koodin ja tekevät sen vaatiman yksikkötestauksen. Ohjelmoija laatii järjestelmän metaforan yhdessä asiakkaan kanssa – ohjelmoijaa pidetään XP:n sydämenä. *Asiakas* (engl. customer) on yhtä lailla merkittävässä, jopa haastavassa asemassa XP-projektissa – tämän tulee kyetä laatimaan selkeitä käyttäjätarinoita ja kyetä vaikuttamaan projektin ominaisuuksiin kuitenkaan niitä suoranaisesti hallitsematta. Tämä on usein kaikki uutta asiakkaalle projektin alkaessa ja projektin edetessä asiakkaan tulee kyetä tekemään merkittäviä päätöksiä ominaisuuksia priorisoitaessa. *Testaajan* (engl. tester) rooli on tukea asiakasta toiminnallisten testien laatimisessa, koska vetovastuu itse koodin testaamisesta on ohjelmoijilla. *Mittaaja* (engl. tracker) seuraa projektin aikatauluja ja ohjaa kehitystiimiä niiden parantamisessa, sikäli kun projekti ei pysy alkuperäisessä aikataulusuunnitelmassa. Ketterissä menetelmissä on usein henkilö, joka tuntee hyödynnettävän menetelmän – XP:ssa on *valmentaja* (engl. coach). Hän varmistaa, että projekti toteutuu XP:n arvojen, käytäntöjen ja sääntöjen mukaisesti. XP-projektiin kuuluu *konsultti* (engl. consultant), joka toimii liiketoiminnallisena tai teknisenä asiantuntijana. Projektissa *johtaja* (engl. owner, ”big boss”) vastaa kokonaisvaltaisesti projektin päätöksenteosta, laaja-alainen kommunikaation kehitystiimin kanssa kuuluu olennaisesti johtajan tehtäviin. Hän on usein, joka kertoo, miten edetään, jos jokin poikkeaa odotuksista tai suunnitelmasta. (Beck 1999: 105-111.)

4.1.4 Elinkaari

Extreme Programmingissa ohjelmistotuotteella on kuusivaiheinen elinkaari: tutkimusvaihe, suunnitteluvaihe, iteraatiovaihe, tuotteistusvaihe, ylläpitovaihe ja päätösvaihe (Abrahammson ym. 2002). Seuraavaksi esitetyt vaihekuvaukset (Kuvio 4) perustuvat Abrahamssonin ym. (2002: 19-21) ja Beckin & Fowlerin (2000) näkemyksiin.



Kuvio 5. Extreme Programming -elinkaaren vaiheet mukailtuna (Abrahamsson ym. 2002:19)

Tutkimusvaiheessa asiakas laatii ensimmäiseen julkaisuun toivomansa toiminnot tarinakortteihin – kukin tuoteominaisuus omaan korttiinsa. Tässä vaiheessa kehitystiimi omaksuu projektissa käytettävät kehitystyökalut, teknologian ja käytännöt. Ominaista on, että projektin alkuvaiheessa laaditaan järjestelmästä prototyyppi, jonka avulla voidaan testata valittu teknologia ja tutkia järjestelmään sopivia arkkitehtuurivaihtoehtoja. Vaiheen kesto riippuu siitä, kuinka hyvin kehitystiimi –pääasiassa ohjelmoijat– tuntevat käytettäväksi valitun teknologian.

Suunnitteluvaiheessa tarinat priorisoidaan ja ensimmäisen pienen julkaisun sisällöstä tehdään päätös. Ohjelmoijat tutkivat tarinoiden vaativuutta, jonka perusteella luodaan alustava aikataulu. Suunnitteluvaihe kestää noin kaksi päivää.

Ennen ensimmäistä julkaisua suoritetaan useampia iteraatioita *iteraatiovaiheessa*. Suunnitteluvaiheessa tehty alustava aikataulu on pilkottu iteraatioihin, jotka toteutetaan tässä vaiheessa ennen niiden käyttöönottoa. Ensimmäisen iteraation tuloksena syntyy kokonaisvaltainen järjestelmäarkkitehtuuri, joka saavutetaan valitsemalla järjestelmän rakentamista tukevat tarinat. Asiakas valitsee toivomansa tarinat kuhunkin iteraatioon ja jokaisen iteraation päätteeksi suoritetaan asiakkaan laatimat toiminnalliset testit. Viimeisen iteraation päätteeksi järjestelmä on valmis tuotantoon.

Ennen kuin järjestelmä julkaistaan virallisesti asiakkaalle, tehdään *tuotteistusvaiheessa* erilaisia testejä ja varmistuksia koskien järjestelmän suorituskykyä. Tässäkin vaiheessa on vielä mahdollista tehdä muutoksia ja niihin liittyvät päätökset tulee tehdä viimeistään tässä vaiheessa ennen julkaisemista. Jos muutoksia vielä vaaditaan, lyhennetään iteraation pituutta tässä vaiheessa olemaan enimmillään viikon pituinen. Mahdolliset myöhemmin toteutettavat ehdotukset tai valmiit ideat dokumentoidaan myöhempää implementointia varten.

Ensimmäisen asiakkaalle tehdyn julkaisun jälkeen, tulee projektin kyetä pitämään olemassa oleva julkaisu tuotannossa ja samanaikaisesti tuottamaan uusia iteraatioita. Tämä on *ylläpitovaihe*, jossa vaaditaan myös asiakastuen ylläpitämistä ohjelmistokehityksen ohella. Kehitystiimi voi tässä vaiheessa kokea rakenteellisia muutoksia, kun aktiivinen kehitysnopeus on hieman taantunut järjestelmän ollessa asiakkaan käytössä.

Päätösvaihe perustuu siihen, ettei asiakkaalla ole enää uusia käyttöönotettavia tarinoita. Tämä toki vaatii, että järjestelmä tyydyttää asiakkaan kaikilta osin – myös suorituskyvyn ja luotettavuuden näkökulmista. Päätösvaiheessa kaikki tarpeellinen dokumentaatio on laadittu, eikä arkkitehtuuria, järjestelmärakennetta tai koodia enää muokata. Päätösvaiheeseen on mahdollista päätyä myös, jos järjestelmä ei vastaa odotuksia tai sen jatkokehittäminen on liian kallista.

4.2 Scrum

Scrum on ketterä lähestymistapa innovatiivisten tuotteiden ja palveluiden kehittämiseen. Se on hallintalähtöinen menetelmä, joka on muihin menetelmiin verrattuna vaivaton ja yksiselitteinen ja sitä voidaan toteuttaa useissa erilaisissa projekteissa (Schuh 2005: 23). Scrum sopii monimutkaisimpiin toimintaympäristöihin ja projekteihin erityisen hyvin, koska juuri Scrum-ympäristöissä vaaditaan kykyä tutkia ja vastata havaittuihin vaatimuksiin (Rubin 2013: 8). Scrum perustuu hyvin perinteiseen ketterän kehityksen ajatukseen – kehitystyö toteutetaan lyhyissä iteraatioissa, joiden pituus vaihtelee viikosta kuukauteen. Kunkin iteraation aikana itseorganisoitunut, poikkitoimintoinen tiimi toteuttaa kaiken työn suunnittelusta testaukseen. Kunkin iteraation tuloksena tiimin tulisi kyetä tuottamaan valmiita, toimivia toimintoja. Tällä periaatteella kehitettävä järjestelmä toteutetaan iteraatioissa vaiheittain ja yhden iteraation loppuessa, alkaa seuraava jälleen nollatilanteesta suunnittelutyöllä. Scrumia käytetään eniten ohjelmistotuotteiden kehittämiseen, mutta menetelmän ydinarvoja –ja käytäntöjä voidaan käyttää ja niitä on käytettykin erilaisten tuotteiden kehittämiseen. Menetelmä sopii esimerkiksi erityyppisten töiden organisointiin – esimerkiksi laitteistojen kehittämisessä tai markkinointi –ja myyntihankkeissa. (Rubin 2013: 1-3.)

Tässä alaluvussa esitellään Scrumin taustalla oleellisesti vaikuttavaa empiiristä prosessien hallintaa, Scrumissa esiintyviä rooleja sekä kokonaisvaltaista viitekehystä tämän iteratiivisen menetelmän taustalla.

4.2.1 Empiirinen prosessienhallinta

Scrum ei ole standardoitu prosessi, jossa toteutetaan perättäisiä vaiheita, joiden taataan tuottavan toivottuja tuloksia, vaan kyse on työn organisoimisen ja hallinnan viitekehyksestä. Se perustuu joukkoon arvoja, periaatteita ja käytäntöjä joka toimii pohjana projektissa lopulta käytettäville relevanteille käytännöille ja spesifeille lähtökohdille joiden mukaisesti Scrumia tulkitaan. Scrum on toimiva menetelmä monimutkaisten eli ennustamattomasti käyttäytyvien ongelmien ratkaisemiseksi. Scrumin taustalla on empiirinen prosessien hallinta (engl. empirical process control), joka on vastakohta määritellylle prosessienhallinnalle (engl. defined process control). Määritellyssä prosessien hallinnassa on kyse prosesseista, jotka toistuvasti tuottavat laadultaan hyväksyttäviä ja toimivia tuloksia kohtuullisin kustannuksin korkeimmalla mahdollisella laa-

dulla – empiirisessä projektien hallinnassa tällainen tilanne ei ole saavutettavissa johtuen juuri aktiviteettien kompleksisuudesta. Scrumissa empiirisellä prosessien hallinnalla kuluja ja laatua voidaan hallita:

- Korostamalla asiakkaalle projektin lopputuloksen kannalta merkittävimpiä näkökulmia
- Sallimalla asiakkaan tutkia tuotetta säännöllisesti ja täten maksimoimalla vaatimustenmukaisuuden
- Jos asiakas näkee tuotteen poikkeavan tavoitteesta, sopeuttamalla tämän mahdollisimman pian kehitysprosessiin tai tuotteeseen. (Rubin 2013: 13; Schuh 2005: 23; Schwaber 2004a.)

4.2.2 Scrum-roolit

Scrumissa roolitus on hyvin selkeä – menetelmässä on vain kolme eri roolia: tuotteen omistaja (engl. product owner), scrum-mestari (engl. ScrumMaster) ja kehitystiimi (engl. development team). Kaikki projektia koskeva hallintavastuu on jaettu näille kolmelle. Tuotteen omistaja vastaa siitä, että kaikki projektissa panoksellaan mukana olevat tahot tulevat kuulluiksi. Luomalla projektin alkuperäiset yleisvaatimukset, tuottavuusobjektit (engl. return of investment (ROI) objectives) ja julkaisusuunnitelmat tuotteen omistaja vastaa projektin alulle saamisesta. Alkuperäisten yleisvaatimusten listaa kutsutaan tuotteen työlistaksi (engl. product backlog) ja tuotteen omistaja vastaa sen käytöstä varmistaen, että tärkeimmät toiminnot toteutetaan ensimmäisenä. Käytännössä tuotteen omistaja priorisoi jatkuvasti tuotteen työlistaa ja ohjaa tärkeimpien toimintojen kehittämisen aina seuraavassa iteraatiossa eli on vastuussa siitä, mitä toteutetaan ja missä järjestyksessä. Kehitystiimi on taas vastuussa toiminnallisuuden kehittämisestä. Tiimi on aina itseorganisoitunut, itsehallinnoitu ja poikkitoiminnollinen eli sen jäsenet toimivat yli tehtävärajojen. Kehitystiimin tehtävänä on päättää kuinka muuntaa tuotteen työlistassa odottavat vaatimukset osaksi projektin kohteena olevan tuotteen tai palvelun toiminnallisuutta. Tiimin jäsenet ovat kollektiivisesti vastuussa kunkin iteraation ja yhtälailla koko projektikokonaisuuden onnistumisesta. Scrum-mestarin tehtävänä on opettaa projektissa mukana oleville tahoille Scrumin perustoiminta, jotta menetelmä saadaan toimimaan organisaatiokulttuurin mukaisesti, menetelmän perustana oleva joukko arvoja, periaatteita ja käytäntöjä huomioiden. (Schwaber 2004a; Rubin 2013: 14-16.)

4.2.3 Scrum-viitekehys

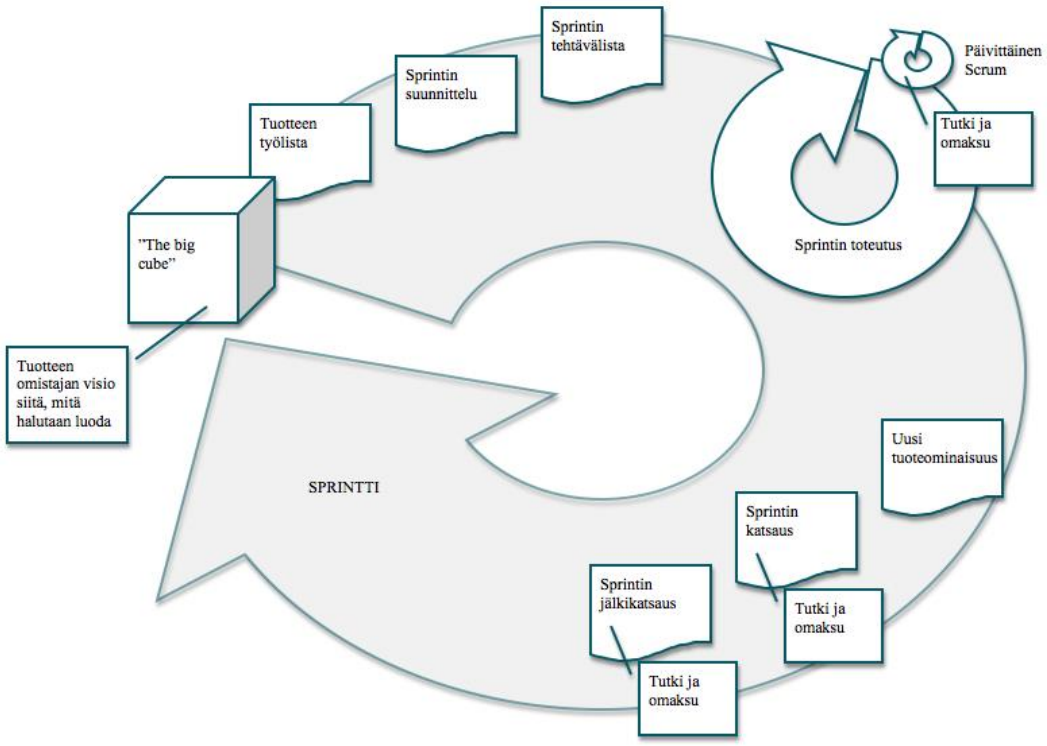
Tässä luvussa esitettävä Scrum-viitekehys perustuu Rubinin (2013) sekä koko Scrumin kehittäjien, Schwaberin ja Sutherlandin (2013) näkemyksiin. Scrum-prosessin läpivienti perustuu aiemmin kuvatusti iteraatioihin, joiden tuloksena halutaan saada aikaan *valmiita toimivia tuoteominaisuuksia* (engl. product increment tai product feature). Scrumissa iteraatiot ovat *sprinttejä* (engl. sprint). Sprintit ovat ajallisesti rajattuja – niille määritetään kesto, josta ei sen jälkeen poiketa. Yksittäinen sprintti voidaan ajatella kalenterikuukauden mittaiseksi projektiksi. Jokaisen sprintin –kuten projektinkin tavoitteena on saada aikaan jotain. Ennen sprinttiä määritellään mitä on rakenteilla ja tehdään joustava suunnitelma, joka ohjaa kuinka tämä rakennetaan. Lopulta tehdään suunnitelmanmukaisesti haluttu työ ja viedään projekti maaliin. Schwaber ja Sutherland (2013) korostavat sprinttien etuina lyhyttä kestoa, jolloin monimutkaisuus ja riskit pysyvät rajallisina. Eduksi he näkevät myös asioiden ennustettavuuden, kun kyetään läpi sprintin tutkimaan ja omaksumaan ympäristöä sekä kustannusriskien pysymisen alhaisina puhuttaessa vain kuukauden kustannusjaksoista.

Sprintti alkaa sprintin suunnittelulla, joka on koko Scrum-tiimin yhdessä toteuttama vaihe. Suunnitteluvaihe kestää enintään 8 tuntia kuukauden kestoisessa sprintissä. Suunnitteluvaiheessa halutaan saada vastaukset kysymyksiin:

1) Mitä voidaan tehdä tässä sprintissä?
2) Kuinka haluttu työ saadaan tehdyksi?

Kunkin sprintin lähtökohta perustuu tuotteen omistajan näkemykseen siitä, mitä sekä sisäisten että ulkoisten sidosryhmien odotusten perusteella halutaan luoda – Rubin kutsuu tätä näkemystä ja lähtökohtaa nimellä *the big cube*. Koska tämä näkemys on usein varsin laaja, pilkotaan se osiin – joukoksi toivottuja tuoteominaisuuksia, jotka priorisoidaan tuotteen omistajan laatimaan *tuotteen työlistaan*. Priorisointi perustuu lisäarvon tuottoon, kustannustehokkuuteen, käytössä olevaan tietämykseen sekä riskienhallintaan. Uusia ohjelmistotuotteita kehitettäessä tuotteen työlista kattaa luonnollisesti pelkästään uusia tuoteominaisuuksia. Vastaavasti, jos kyseessä on jo olemassa olevan ohjelmistotuotteen kehitysprojekti, voi työlista sisältää muiden muassa uusia tuoteominaisuuksia, muutoksia olemassa oleviin ominaisuuksiin, virheiden korjauksia tai teknisiä parannuksia. Se, mitä sprintin aikana kyetään siis toteuttamaan perustuu sii-

hen, että kehitystiimin, tuotteen omistajan ja tuotteen työlistan välillä on selkeä ymmärrys. Kun sprinttikohtainen tavoite on asetettu, ja on selkeää, mitä kyetään ja halutaan saavuttaa, määritetään, kuinka tämä kaikki tapahtuu. Kehitystiimi päättää, kuinka se rakentaa toiminnalliseen muotoonsa tuotteen työlistasta poimitun tuoteominaisuuden sprintin aikana. Tuotteen työlistasta tässä sprintissä toteutettavaksi valittu ominaisuus ja suunnitelma siitä kuinka se toimitetaan muodostavat *sprintin tehtävälistan*. Sprintin tehtävälistan avulla, koko suunnitteluvaiheen päätteeksi, kehitystiimin pitäisi kyetä selittämään tuotteen omistajalle ja Scrum-mestarille kuinka se tulee työskentelemään itseorganisoidusti saavuttaakseen sille asetetun tavoitteen ja luomaan odotetunlaisen tuoteominaisuuden. Kuviossa 6 on kuvattu Scrum viitekehys sprintin sisällön muodossa. (Rubin 2013: 20-22; Schwaber & Sutherland 2013.)



Kuvio 6. Scrum -viitekehys (Rubin 2013:17)

Suunnitteluvaiheen päätyttyä kehitystiimi siirtyy Scrum-mestarin ohjaamana sprintin toteutukseen. Kuten edellä esitettiin, sprintin käytännön toteutus tapahtuu päivittäisinä yhteisinä *Scrum-tuokioina* (engl. daily Scrum), joiden perusteella kehitystiimi kykenee edelleen toimimaan itseorganisoidusti. Päivittäinen Scrum kestää yleisesti 15 minuuttia ja siinä kehitystiimi tutkii editymistä kohti sprintin tavoitetta sekä kuinka edistymi-

nen vastaa sprintin tehtävälistaa ja omaksuu toistensa näkemykset yhtenäisiksi saavuttaen käsityksen tulevan vuorokauden työstään. Kehitystiimin kunkin jäsenen tulisi selittää seuraavat kysymykset:

1) Mitä tein tavoitteemme saavuttamiseksi eilisen päivittäisen Scrumin jälkeen?
2) Mitä teen tänään tavoitteemme saavuttamiseksi?
3) Tunnistanko jotain esteitä tavoitteen saavuttamiseksi?

Päivittäiset Scrumit edistävät kommunikointia, minimoivat muut tapaamiset, edistävät esteiden löytämistä, jolloin vaaditaan myös nopeaa päätöksentekoa sekä edistävät ennen kaikkea koko kehitystiimin tietämyksen tasoa. Tähän perustuu Scrum-kehittäjien ajatus asioiden tutkimisesta ja omaksumisesta tapaamisissa. (Rubin 2013: 23-25; Schwaber & Sutherland 2013)

Sprintin toteutusvaiheen jälkeen lähtökohta on se, että Scrum-tiimillä tulisi olla käsissään lopulliseen kehityksen kohteena olevaan ohjelmistotuotteeseen valmis tuoteominaisuus lisättäväksi. Rubin (2013: 25) kuvaa valmista tuoteominaisuutta nimellä *toimituspotentiaalinen tuoteinkrementti* (engl. potentially shippaple product increment). Tällä viitataan siihen, että sprintissä toteutettavaksi haluttu tuoteominaisuus tai tuoteominaisuuksien joukko on todellakin valmiiksi tehty ja täysin tarkoituksenmukaisella tavalla. Edellytyksinä pidetään korkealaatuisuutta ja potentiaalista toimitusvalmiutta. Sekä Rubin (2013:26) että Schwaber ja Sutherland (2013) pitävät tässä kohtaa tärkeänä, että tuote on todellakin ”tehty” (engl. ”done”) ja määrittelevät tuotteen valmiina olon tarkasti. Heidän mukaan ”tehty” -tila vaihtelee merkittävästi Scrum-tiimistä riippuen, mutta oleellista tässä on, että jokaisella projektissa mukana olevalla taholla on yhteisymmärrys siitä, mitä tehdyn työn valmiina oleminen tarkoittaa, jotta voidaan taata yhdenmukainen hyväksyttävyys. Schwaber ja Sutherland (2013) mainitsevat, että Scrum-tiimit usein kokemuksen myötä kehittyvät oman ”tehty”-tilansa arvioimisessa – tämä näkyy esimerkiksi siinä, mitä lopulta kirjataan sprintin tehtävälistaan kyeten lukemaan ja huomioimaan tiimin suorituskyky ja mitä todellisuudessa kyetään yhden sprintin aikana saattamaan ”tehty”-tilaan.

Jokaisen sprintin päätteeksi suoritetaan *sprintin katsaus* (engl. Sprint review), jossa tutkitaan aikaan saatua tuoteominaisuutta ja omaksutaan tuotteen työlistan tila – saatiinko aikaan haluttu tulos ja tuleeko työlistaan tässä kohtaa muutoksia. Sprintin katsauksessa koko Scrum-tiimi ja kaikki sidosryhmät käyvät yhdessä läpi, mitä sprintin

aikana on tehty. Katsauksessa halutaan kyetä tuottamaan monitahoista palautetta ja miettimään yhdessä mitä tehdään seuraavaksi lisäarvon optimoimiseksi. Sprintin katsaukseen osallistuvat Scrum-mestari ja tuotteen omistajan kutsumat avainasemassa olevat sidosryhmät. Katsauksessa tuotteen omistaja kuvaa, mitä kohtia tuotteen työlistasta on saatettu ”tehty”-tilaan ja mitä ei sekä kehitystiimi kertoo, mitkä asiat sprintin aikana onnistuivat, mitkä epäonnistuivat ja kuinka esiintyneet esteet ratkaistiin. Kehitystiimi myös demonstroi tehdyksi saatetun työn ja vastaa tuoteominaisuutta koskeviin kysymyksiin. Katsaukseen kuuluu myös, että tuotteen omistaja esittää sen hetkisen tuotteen työlistan ja tekee siihen mahdollisia muutoksia tarvittaessa. Koko Scrum-tiimi ja sidosryhmät käyvät läpi, mitä tehdään seuraavaksi, saadakseen aikaan arvoa tuottavan pohjustuksen seuraavan sprintin suunnittelulle. Tämän lisäksi tehdään katsaus koskien mahdollisiin muutoksiin tuotteen käyttöön tai tulevaan käyttöalueeseen liittyen sekä katsaus ajanhallintaan, budjettiin, potentiaalisiin esiin nousseisiin kykyihin ja tuleviin odotettuihin tuotejulkaisuihin. Sprintin katsauksen tuloksena on uudelleenarvioitu tuotteen työlista, josta ilmenee mahdolliset seuraavassa sprintissä toteutettavat tuoteominaisuudet. Tässä kohtaa tuotteen työlistaan on myös mahdollista tehdä muutoksia uusien mahdollisuuksien varalta.

Sprintti päättyy aina mennyttä sprinttiä koskevaan *jälkikatsaukseen* (engl. Sprint retrospective). Siinä Scrum-tiimin on mahdollista tutkia itseään ja luoda suunnitelma kehityskohteille ennen seuraavaa sprinttiä. Sprintin jälkikatsauksen tarkoituksena on tutkia ihmisten, suhteiden, etenemisen ja työkalujen kannalta, kuinka viimeisin sprintti sujui sekä tunnistaa ja laittaa järjestykseen pääkohdat onnistumisista ja kehityskohteista. Lopulta jälkikatsauksessa luodaan suunnitelma kehityskohteiden korjaamiseksi.

4.3 Kanban

Kanban -ajatus on syntynyt autovalmistaja Toyotan asiantuntijoiden toimesta ja sen juuret ovat jopa 1940-luvulla. Kanbania on alettu autoteollisuuden ohella myöhemmin soveltamaan myös muilla toimialoilla, kuten ohjelmistotuotannossa. Pyrkimällä tasapainottamaan tekeillä olevan työn määrää kehitystiimin kapasiteettiin, Kanban tarjoaa joustavammat suunnittelumahdollisuudet, nopeamman tuotannon, selkeän fokuksen sekä läpinäkyvyyden koko kehityssyklien ajan. Kanban edustaa juuri oikeaan tarpeeseen (engl. just-in-time) -ajattelua sen perimmäisessä merkityksessään eli tavoitellaan ihanteellista toimitusaikaa sekä –määrää. (Atlassian 2015.)

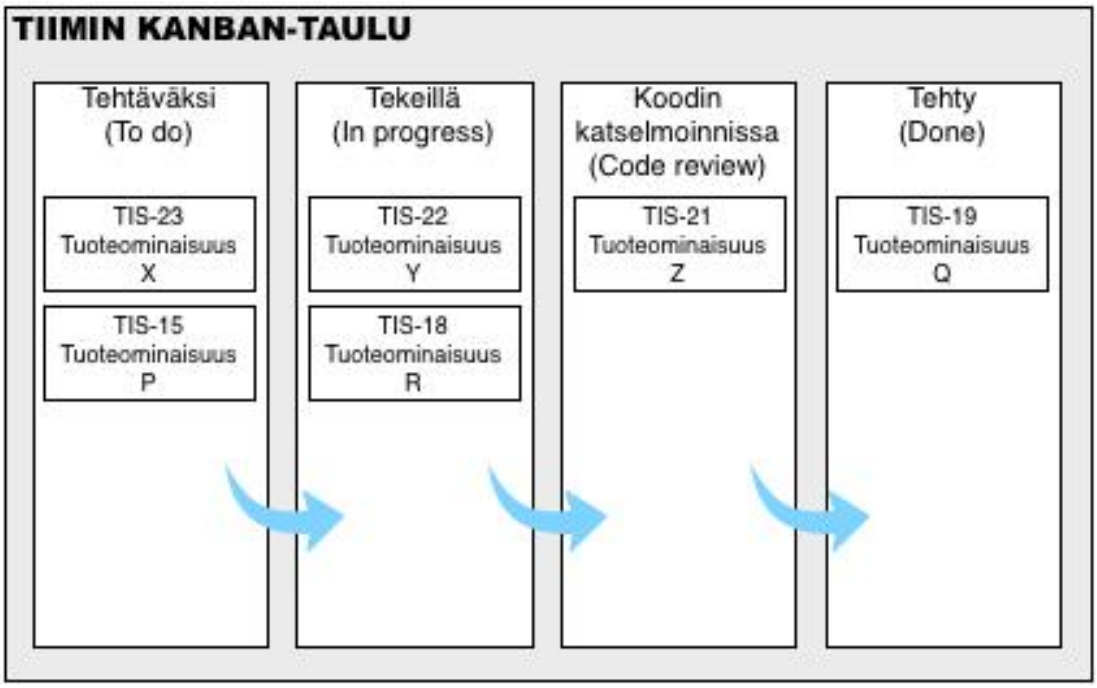
Alkuperänen Kanban on kuvattu sen kehittäjien toimesta tuotannonhallintajärjestelmäksi just-in-time tuotantoa varten. Kanbanin nähdään laskevan prosessinhallintakustannuksia, koska se ei vaadi erillisiä, reaaliaikaisia järjestelmiä kutakin erillistä prosessia ja ostajaa sekä näiden edellyttämiä tuotantoaikatauluja varten. Kanbanin nähdään tuottavan nopeaa dataa muuttuvassa tuotantokapasiteetissa, toimintavalmiudessa tai henkilöstön määrässä. Ajatus vain tarvittavan määrän tuottamisessa myös minimoi ylituotannon sekä mahdollistaa tuotannon optimoinnin kysynnän mukaiseksi. (Atlassian 2015; Sugimori, Kusunoki, Cho & Uchikawa 1977.)

4.3.1 Viitekehys

Kanban-tiimi keskittyy ainoastaan siihen työhön, joka on juuri sillä hetkellä tuotannossa. Kun työkokonaisuus saadaan valmiiksi, otetaan tuotteen työlistasta (engl. product backlog) seuraava työkokonaisuus käsittelyyn. Tuotteen omistaja (engl. the product owner), joka hallinnoi työlistaa, voi vapaasti uudelleenpriorisoida työlistan järjestystä, koska tiimi työskentelee jatkuvasti yhden työkokonaisuuden kanssa, joten mahdolliset muutokset työlistassa eivät siten vaikuta heidän sen hetkiseen työhönsä. Maksimaalisen liiketoiminnallisen arvon tuottamiseksi on merkityksellistä, että tärkeimmät tuoteominaisuudet ja työkokonaisuudet ovat työlistan kärjessä eli valmiina työstettäväksi seuraavaksi. Kanbanissa tärkein työn mittari on aikataulu ja suunnitellun iteraation keston mukaisesti toimiminen. Iteraation kesto tässä tarkoittaa ajallista pituutta siitä, kun työ otetaan kehitystiimin toimesta vastaan siihen, kun tuoteominaisuus tai työkokonaisuus toimitetaan eteenpäin. Kanban-tiimi muodostuu yksilöistä, joilla ei ole Scrumin tai eXtreme Programmingin tapaan nimettyjä tehtäviä eikä rooleja missään määrin, vaan kehitystiimin sisäisen tietämyksen jakaminen on erittäin merkittävässä asemassa ja sitä on määrä edistää läpi työskentelyn. Heterogeenisin työtehtävin voidaan myös edistää iteraation kestoa. Kanban-viitekehyksessä koko tiimi on yhteisvastuussa siitä, että työkokonaisuus etenee prosessin läpi sulavasti. Kanbanissa halutaan pyrkiä rajoittamaan ja minimoimaan käsitteillä olevan työn (engl. WIP, work in progress). Mitä enemmän keskeneräisiä työkokonaisuuksia tai mitä useammassa projektissa tiimiläiset ovat samanaikaisesti mukana, sitä vaikeampaa ja monimutkaisempaa on työkokonaisuuksien tai yksittäisten tuoteominaisuuksien valmiiksi saattaminen.

4.3.2 Kanban-taulu

Kanbanissa on usein työn etenemisen ja yleisen seurannan tehostamiseksi käytössä jonkinlainen Kanban-taulu (Kuvio 6). (Atlassian 2015; VersionOne 2015.)



Kuvio 7. Esimerkki Kanban -taulusta (VersionOne 2015).

Kanban-taululla tuotetaan selkeä kuva projektin etenemisestä, koska se näyttää kunkin kehitystiimin jäsenen tekemät tehtävät, selventää kunkin ominaisuuden priorisointia sekä sen avulla voidaan löytää projektin etenemistä hankaloittavat mahdolliset pullonkaulat. Sen avulla seurataan, mitkä tuoteominaisuudet ovat tuossa tehtäväksi, mitkä ovat parhaillaan tekeillä, mitä ominaisuuksia katselmoidaan ja mitkä ovat saatettu täysin valmiiksi ”tehty”-tilaan. Taulun avulla tuotetaan siis vain halutut ominaisuudet ja minimoidaan edellä kuvatusti tekeillä olevan työn määrä. Keskityttäessä vain muutamaan tuoteominaisuuteen kerrallaan, julkaisut asiakkaalle ovat jatkuvia. Kanbanin avulla kyetään omaksumaan projekti asiakkaan tarpeiden mukaiseksi, koska käytössä ovat lyhyet palautekierrokset. Menetelmän toiminnan avain on keskittyminen jatkuvaan etenemiseen välttämättömien iteraatioiden sijasta. (Ahmad, Markkula & Oivo 2013.)

Kanbania pidetään jokseenkin samankaltaisena menetelmänä kuin Scrum, mutta toki molemmilla on tietyt ominaispiirteensä, esimerkiksi Scrumissa tarkoin määritellyt roolit (tuotteen omistaja, Scrum-master ja kehitystiimi). Kanbanissa ei rooleja ole ennalta määritelty vaan kehitystiimi voi oman näkemyksensä ja toimivaltansa puitteissa määrittää haluamansa roolit tiimille. (Kiberg & Skarin 2010: 11.)

5 TUTKIMUSASETELMA

Tässä luvussa käsitellään tutkimuksen toteutusympäristö, esitellään kohdeorganisaatio, jossa tapaustutkimus on toteutettu sekä esitellään tutkimuksessa käytetyt aineistonkeruumenetelmät, jotka ovat kysely sekä teemahaastattelu. Luvussa käsitellään myös aineiston analysointiin käytettyä teemoittelutekniikkaa.

Tapaustutkimuksen kohdeorganisaatio oli IT -palvelualan yritys, joka toimii merkittävin markkinaosuuksin sekä Suomessa että kansainvälisesti. Yrityksessä työskentelee noin 13 000 työntekijää useissa eri maissa. Yritys palvelee asiakkaitaan lukuisilla eri toimialoilla – niin yksityisellä kuin julkisellakin sektorilla, kuten pankki -ja vakuutusalalla sekä muiden muassa metsäteollisuuden toimialalla. Yritys pyrkii IT -ratkaisuillaan- ja palveluillaan kehittämään asiakasyritystensä liiketoimintaa muuntautumiskykyisemmäksi sekä edistämään näiden kilpailuetua. Kohdeorganisaatiosta osallistui tutkimukseen kolme henkilöä, jotka toimivat projektipäällikön tehtävissä. Tutkimusaineisto muodostetaan heidän vastausten pohjalta seuraavissa alaluvuissa esiteltävään kyselyyn että teemahaastatteluun.

5.1 Aineistonkeruumenetelmät

Tämän tutkielman aineisto hankitaan kahdella menetelmällä, teemahaastattelulla sekä haastattelua tukevalla esikyselyllä. Menetelmien perusominaisuudet ovat esitelty luvussa 1.2. Seuraavissa alaluvuissa esitellään tutkimusmenetelmät yksityiskohtaisemmin ja niiden käytännön toiminta tässä tutkimuksessa.

5.1.1 Kysely

Tutkimuksen aineisto muodostetaan ensimmäisessä vaiheessa esikyselyllä. Ennen haastatteluita kutakin haastateltavaa pyydettiin antamaan näkemyksensä projektin onnistumisesta ketteryyden näkökulmasta Likert –asteikolla (asteikko 1-5) muodostettuun kyselyyn. Kysely on laadittu käyttäen pohjatietona tutkimuksen teoreettista viitekehystä, mutta yksityiskohtaisemmin kysymykset on muodostettu pohjautuen luvussa 3.4 kuvattuun Chown ja Caon (2007) kyselytutkimukseen, jossa he selvittivät kriittisimpiä menestystekijöitä ketterässä ohjelmistokehityksessä. Kyselyssä – kuten teemahaastattelussakin – kysymykset esitettiin vastaajille teemoittain. Kyselyn teemat

on johdettu Chown ja Caon (2007) näkemysten perusteella ja kuvaavat merkittävimpiä menestystekijöitä projektin ketteryyden takaamiseksi. Taulukossa 5 on esitelty kyselyn teemat.

Teema
1. Toimitusstrategia
2. Ketterät tekniikat
3. Tiimin kyvykkyys/taidot
4. Projektijohto
5. Toimintaympäristö
6. Asiakkaan läsnäolo

Taulukko 5. Esikyselyn teemat

Koko esikysely on kuvattu kysymyksineen liitteessä 1. Kysely toimii tutkimuksessa merkittävässä asemassa, sillä siinä saatuja ennakkotietoja kyettiin hyödyntämään avoimien jatkokysymysten laatimisessa itse teemahaastattelussa. Teemahaastattelu toisin sanoen täydentää kyselyä ja kyselyvastauksia yksityiskohtaisesti.

5.1.2 Teemahaastattelut

Haastattelut olivat puolistrukturoituja ja ne toteutettiin kohdeorganisaation tiloissa. Haastatteluiden teemat perustuivat ketteriin arvoihin ja alakysymykset ketteriin periaatteisiin, lisäksi ketterillä menetelmillä saavutettuja hyötyjä pyrittiin selventämään avoimin lisäkysymyksin. Haastattelukysymykset on esitetty liitteessä 2. Taulukossa 6 on esitetty teemahaastattelun teemat.

Teema
1. Yksilöt ja kanssakäyminen
2. Toimiva ohjelmisto (dokumentaation sijaan)
3. Asiakasyhteistyö
4. Muutokseen vastaaminen

Taulukko 6. Teemahaastattelun teemat

Haastateltavat olivat valittuja niin, että mukaan saatiin sekä ketteryydeltään onnistuneita että heikommin onnistuneita projekteja. Jotkin projekteista olivat siis sisällöllisten käytäntöjensä perusteella puhtaammin ketteriä, kuin toiset. Tällä halutaan tuottaa vastakkaisia tuloksia, jotta saadaan konkreettinen näkemys myös ketterän ohjelmistokehityksen kehityskohteista organisaatiossa ja voidaan mahdollisesti löytää motiiveja esiintyville epäkohdille. Tieteellisessä tutkimustyössä tavoiteltaessa tai ennustettaessa samankaltaisia tuloksia, puhutaan ns. sananmukaisesta toistosta (*literal replication*) ja haluttaessa saada aikaan vastakkaisia tuloksia, puhutaan teoreettisesta toistosta (*theoretical replication*). Useammalla tapauksella kyetään kuvaamaan teoriaa paremmin kuin esimerkiksi määritelmillä ja tapauksia vertaamalla kyetään testaamaan teoreettisia väitteitä ja kirkastamaan käsitteitä (Järvinen & Järvinen 2011: 78). Kaikki kolme (3) haastateltavaa olivat projektipäälliköitä. Kukin haastattelu oli kestoltaan noin tunnin ja jokainen haastattelu nauhoitettiin sekä kirjattiin haastattelun aikana pääkohdiltaan muistiinpanoiksi.

5.2 Tutkimusaineiston analysointi

Tutkimusaineiston analysoinnissa hyödynnetään teemoittelua. Teemoittelun perusajatuksena on hahmottaa tutkielman kannalta keskeisimpiä aihepiirejä eli teemoja. Teemoina toimivat tämän tutkielman keskeisimpänä sisältönä olevat ketterät arvot ja periaatteet, jotka on esitelty Agile Manifeston (2001) mukaisesti tämän työn luvussa 2. . Tämän tutkielman tapauksessa teemat muodostetaan ketteristä arvoista ja niitä tarkennetaan kunkin teeman alle asettuviin ketteriin periaatteisiin, joiden pohjalta muiden muassa rakennetaan teemahaastattelun kysymykset (Jyväskylän yliopiston Koppa 2015). Tutkimusaineistoa analysoitaessa esikyselyn tapauksessa vastaajilta pyydetään heidän näkemyksensä kunkin teeman alla esitettäviin kysymyksiin. Kukin teema antaa kuvan tietystä projektissa esiintyvästä käytäntökokonaisuudesta ja kunkin teeman kysymyksiin saaduista vastauksista johdetaan keskiarvo, joka antaa kuvan siitä, kuinka hyvin kyseinen teema on projektissa onnistunut. Teemahaastattelun tapauksessa analysointi tapahtuu käymällä haastattelun vastaukset läpi ja vertaamalla niitä Agile Manifeston sisältöön eli ketteriin arvoihin ja periaatteisiin. Vertaamalla saadaan selville, miten projektissa on hyödytty kunkin projektissa esiintyneen ketterän arvon ja periaatteen käyttämisestä. Esikyselyn vastausten teemoittain esitetyt keskiarvot sekä teemahaastattelun vastaukset muodostavat yhdessä aineiston, jonka pohjalta voidaan myös

havaita mahdolliset heikkoudet, joita ketterien arvojen ja periaatteiden hyödyntämisestä on kohdeorganisaatiolle seurannut.

6 TUTKIMUSAINEISTO JA -TULOKSET

Tutkielman aineisto perustuu kolmeen kohdeorganisaation toteuttamaan ohjelmistokehitysprojektiin. Ennen varsinaista aineistonkeruuta kerättiin projektien perustiedot, joiden perusteella saatiin tietoon kunkin projektin menetelmä, projektilokaatio sekä projektin kohteen tyyppi. Aineistoksi koottiin tutkittujen esikyselylomakkeen vastaukset sekä teemahaastattelukysymysten vastaukset. Tutkimusaineistossa esitellyt projektit ovat toteutettu vuosien 2014 ja 2015 aikana sekä osittain jatkuvat edelleen. Projektit on toteutettu yrityksen finanssi- ja vakuutusasiakkaita palvelevassa yksikössä, joka on Suomessa hyvin merkittävän markkinaosuuden omaava liiketoiminta-ala kohdeorganisaatiolle. Tässä luvussa esitellään tutkielman aineiston muodostava osio luvuissa 6.1 – 6.3, joissa kuvataan tutkittujen projektien perustiedot, projektipäälliköille esitetyn kyselyn vastaukset sekä teemahaastattelussa ilmenneet näkökohdat. Luvut 6.4 – 6.6 sisältävät edellä kuvatun aineiston pohjalta saavutetut tutkimustulokset.

6.1 Projektien perustiedot

Tutkimusaineiston muodostaneet projektit olivat kukin toteutettu eri ketterällä menetelmällä, niiden tuloksena syntyi erilaisia tuotteita tai palveluita sekä projektien toteutuslokaatioissa oli vaihtelevuutta. Tutkielmassa projekteihin viitataan nimillä projekti 1, projekti 2 ja projekti 3. Tässä luvussa esitellään aineiston muodostavien projektien perustiedot, jotka on esitetty taulukossa 7.

Perustiedot	**Projekti 1**	**Projekti 2**	**Projekti 3**
Käytetty menetelmä	Kohdeorganisaation oma menetelmä	Scrum	Kanban
Projektilokaatio	Globaali: Suomi, Intia	Suomi	Suomi
Projektikohde	Kehitysprojekti. Olemassa olevan verkkopalvelun uudistaminen.	Uusi ohjelmistoprojekti. Asiakasorganisaation julkinen sivusto ja sähköinen asiointipalvelu.	Versiokehitys. Perusjärjestelmien ja verkkopalveluiden versiotoimitusta.

Taulukko 7. Projektien perustiedot

Projekti 1 toteutettiin kohdeorganisaation itse kehittämällä menetelmällä. Menetelmä mukaili pitkälti Scrumin periaatteita perustuen sprintteihin ja niiden jälkitoimina toteutettuihin Demo -tilaisuuksiin, joissa käytiin sekä kohdeorganisaation että asiakasorganisaation projektipäälliköiden toimesta läpi mennyttä sprinttiä. Projekti 1 oli lokaationsa puolesta globaali – se toteutettiin osittain Suomessa, osittain Intiassa. Projektihenkilöstö koostui toimittajaorganisaation puolesta sisäisestä ohjausryhmästä, projektijohdosta, laatuvastaavista sekä projektin avainhenkilöistä. Tutkielman kohdeorganisaation eli projektin toimittajaosapuolen projektihenkilöstön 23 henkilöstä viisi toimi Intiasta käsin, muutoin projektihenkilöstö sekä asiakastaho toimivat Suomessa. Projektin tavoitteena oli tuottaa erään verkkopalvelun seuraava kehitysversio, jonka oli määrä tulla sisältämään uusia palveluita sekä aiemmin toteutettujen palveluiden korjauksia ja muutoksia. Projektin tehtävälista sisälsi toteutustöitä, vaatimusmäärittelyä ja toiminnallista määrittelyä eri sovellusalueille.

Projekti 2:ssa hyödynnetty ketterä menetelmä oli Scrum. Projektilokaatio oli selkeä ja ketterästä näkökulmasta katsottuna tarkoituksenmukainen – projekti toteutettiin yhdessä maassa ja yhdessä, koko henkilöstölle yhteisessä paikassa. Henkilöstö oli muodostettu täysin Scrum -lähtökohtien mukaisesti sisältäen kaikki kolme eri roolia: tuotteen omistajan (engl. product owner), scrum-mestarin (engl. ScrumMaster) ja kehitystiimin (engl. development team) (Rubin 2013: 14-16). Projektihenkilöstö muodostui kuudesta henkilöstä. Projektin tuloksena tuotettiin asiakasorganisaatiolle uusi julkinen sivusto sekä sähköinen asiointipalvelu.

Projekti 3:n toteutusmenetelmä oli Kanban. Projekti oli henkilöstöltään kansallinen ja toteutettiin Suomessa. Projektihenkilöstö poikkesi muista tutkimuskohteina olleista projekteista, koska se koostui kuudesta eri Kanban -tiimistä. Projektissa toteutettiin versiotoimitusta asiakasorganisaatiolle ja projekti kattoi yhteensä 20 sovelluksen versiokehityksen. Kohdesovellukset olivat asiakasorganisaation perusjärjestelmiä ja verkkopalveluita.

6.2 Esikyselyn tulokset

Esikyselyssä esitettiin kolmelle projektipäällikölle kuuden teeman alle koottuja väittämiä, joita heidän tuli arvioida asteikolla 1–5. Asteikossa merkitykset väittämän onnistumiselle projektissa olivat: erittäin huonosti (1), jokseenkin huonosti (2), en osaa sanoa (3), jokseenkin hyvin (4) ja erittäin hyvin (5). Vastaukset on esitetty taulukossa 8. keskiarvoina, jolla on pyritty luomaan kullekin teemalle arvosanan omainen tulos, joka kuvaa teeman kokonaisonnistumista projektissa. Kukin esikyselyn teemakokonaisuus koostuu useammasta kysymyksestä ja kysymysten määrä vaihteli teemoittain. Alla on kuvattu sanallisesti kunkin projektin saamat keskiarvot. Keskiarvoilla saadaan suuntaa tutkielman seuraavassa luvussa esiteltävälle haastatteluaineistolle.

Teema	**Projekti 1**	**Projekti 2**	**Projekti 3**
Toimitusstrategia	3,0	5,0	1,0
Ketterät tekniikat	2,0	4,5	4,5
Tiimin kyvykkyys/ taidot	3,4	4,6	4,0
Projektinjohto	2,4	4,0	4,2
Toimintaympäristö	1,25	4,25	3,5
Asiakkaan läsnöolo	2,66	4,33	4,66

Taulukko 8. Esikyselyn teemoittain saadut vastauskeskiarvot projektikohtaisesti

Toimitusstrategia-teeman alla projektipäälliköt ottivat kantaa ohjelmiston säännölliseen toimittamiseen asiakkaalle sekä tärkeimpien tuoteominaisuuksien ensisijaiseen toimitukseen. Projekti 1:ssa voidaan kyselyn perusteella sanoa toimitusstrategian onnistuneen jokseenkin hyvin (3,0), projekti 2:ssa erittäin hyvin (5,0) ja projekti 3:ssa erittäin huonosti (1,0). Ketterät tekniikat -teema sisälsi projektipäälliköiden näkemyksen projektin rakenteen yksinkertaisuudesta sekä sopivasta dokumentaation määrästä. Projekti 1:ssa näissä onnistuttiin jokseenkin heikosti (2,0), projekti 2:ssa (4,5) ja projekti 3:ssa (4,5) erittäin hyvin. Tiimin kyvykkyys/ taidot -teeman alla selvitettiin projektipäälliköiden näkemys tiimin jäsenten asiantuntevuudesta ja motivoituneisuudesta, tiimin johdon käsityksestä ketteryydestä ja toimintatavoista sekä tiimin saamasta ketteryyden koulutuksen tasosta.

Projekti 1:ssa tiimin kyvykkyys ja taidot olivat hyvällä, keskivertoa paremmalla tasolla (3,4), projekti 2:ssa (4,6) ja projekti 3:ssa (4,0) erittäin hyvällä tasolla. Projektijohto-teeman alla projektipäälliköt antoivat näkemyksensä vaatimusmäärittelyn ketteryydestä, projektihallinnan ketteryydestä, etenemisen seurannasta, kommunikaation tasosta sekä aikataulun selkeydestä. Projekti 1:ssa tässä onnistuttiin jokseenkin heikosti (2,4), kun taas projekti 2:ssa ja projekti 3:ssa tämän teeman nähtiin onnistuneen erittäin hyvin (4,0 ja 4,2). Toimintaympäristö -teeman alla esitetyillä kysymyksillä haluttiin näkemys projektilokaation yhtenäisyydestä, tiimin itseorganisoituneisuuden tasosta, tiimin koosta sekä tiimin yhtenäisyyden tasosta. Projekti 1:ssa tämä onnistui heikosti (1,25), projekti 2:ssa erittäin hyvin (4,25) ja projekti 3:ssa jokseenkin hyvin (3,5). Kuudes teema oli asiakkaan läsnäolo. Siinä haluttiin käsitys asiakassuhteen yleisluonteen positiivisuudesta, asiakkaan vaikutuksesta päätöksentekoon ja läsnäoloon sekä asiakkaan täydelliseen päätösvaltaan. Projekti 1:ssa asiakkaan läsnäolon taso oli jokseenkin heikko (2,66), projekti 2:ssa erittäin hyvä (4,33) ja projekti 3:ssa erittäin hyvä (4,66).

6.3 Teemahaastattelun tulokset

Tutkielmassa toteutetun teemahaastattelun teemat on tuotu esiin luvussa 5. Teemat ovat yksilöt ja kanssakäyminen, toimiva ohjelmisto, asiakasyhteistyö sekä muutokseen vastaaminen. Tässä luvussa käydään teemahaastattelun tulokset läpi teemoittain.

6.3.1 Yksilöt ja kanssakäyminen

Ensimmäinen teema – yksilöt ja kanssakäyminen on merkittävä osa-alue ketterässä ohjelmistokehityksessä. Pääajatuksena ketterältä kannalta katsoen on, että projektihenkilöstö – toimittaja- ja asiakastahot – työskentelevät mahdollisimman paljon yhteisessä lokaatiossa, jotta kanssakäyminen on tehokkaimmillaan ja molemmat organisaatiot hyötyvät tilanteesta jakamalla näkemyksiään, antamalla molemminpuolista palautetta. Taulukossa 9. on esitetty haastattelussa löytyneet, teemalla saavutetut hyödyt projekteittain.

Teema	**Projekti 1**	**Projekti 2**	**Projekti 3**
Yksilöt ja kanssakäyminen	- Viikkokokoukset - Palaute etuna - Asiakas ei aktiivinen - Globaali projekti vaikeutti - Toiminta osittain itseorganisoituvaa	- Toimittaja ja asiakas työskentelevät samassa tilassa - Kommunikaatio ja asiakkaan rooli toimivaa ja selkeää - Tiimi erittäin asiantunteva - Toiminta itseorganisoitunutta	- Tapaamiset virtuaalisia, noin kerran kuukaudessa - Asiakkaan nähtiin olleen valta-asemassa - Tiimi asiantunteva, muutokset vaikeuttavat motivoitumista

Taulukko 9. Yksilöt ja kanssakäyminen tutkimuskohteena toimineissa projekteissa

Projekti 1:ssa isoimmat hyödyt saavutettiin ketteryydelle ominaisella, iteraation jälkeen välittömästi annettavalla palautteella sekä kasvokkain toteutuneilla tapaamisilla. Projektipäällikön näkemyksen mukaan erittäin tärkeää projektin etenemisen kannalta oli välitön asiakkaalta saatu palaute, joka saatiin asiakasorganisaatiolta kasvokkain toteutetuissa viikkokokouksissa. Kasvokkain pidetyistä tapaamisista oli merkittävää hyötyä, koska tapaamisissa saatiin asiakkaalta suoraa palautetta ja nähtiin, mihin suuntaan projekti on menossa. Asiakastaho oli tyytyväinen, että sai tapaamisissa tarkan tilannekatsauksen siitä, missä vaiheessa projektin etenemisen suhteen sillä hetkellä ollaan.

Projekti 1:ssa ketteryyttä hankaloittavia tekijöitä olivat hajautettu ja globaali toimintalokaatio, asiakkaan passiivisuus sekä oman organisaation henkilöstön tietämättömyys ketterien toimintatapojen osalta. Asiakas ei toiminut läpi projektin samoissa tiloissa toimittajaorganisaation kanssa ja projektipäällikön näkemyksen mukaan asiakas ei ollut riittävän aktiivinen, jotta suurin mahdollinen hyöty yksilöistä ja kanssakäymisestä olisi saavutettu. Ketteriä arvoja vastoin projektihenkilöstö toimi globaalisti, mikä selkeästi vaikeutti projektin ketteryyttä. Projekti 1:n henkilöstön nähtiin toimineen osittain itseorganisoituneesti, mutta henkilöstön tietämys ketterien menetelmien suhteen ei ollut projektipäällikön näkemyksen mukaan riittävällä tasolla, jotta täysi itseorganisoidusti toimiminen olisi ollut mahdollista. Esikysely tuki haastattelussa tehtyjä havaintoja, sillä toimintaympäristö-teeman keskiarvo oli heikko (1,25) ja siinä teemassa projektipäällikkö arvioi juuri itseorganisoituneisuuden tasoa, tiimin yhteistä lokaatiota sekä tiimin kokoa ja eheyttä.

Projekti 2:ssa isoimmat hyödyt saavutettiin yksilöiden ja kanssakäymisen suhteen asiantuntevalla henkilöstöllä, yhteisellä toimintalokaatiolla, itseorganisoituneella tiimillä, tahojen välisellä toimivalla kommunikaatiolla sekä asiakastaholla, joka kykeni omaksumaan roolinsa osana ketterää tiimiä hyvin. Projekti 2 edusti ketterän menetelmän puolesta puhtaasti Scrumia ja oli organisoitu ketteryyden hyödyntämisen kannalta esimerkillisesti. Kehitystiimin muodostaneet yksilöt olivat hyvin asiantuntevia ja ketterien menetelmien suhteen kokeneita. Projektin toimittaja- ja asiakasorganisaatio toimivat samassa tilassa, jonka vuoksi kommunikointi asiakastahon kanssa oli esteetöntä ja aina toteutettavissa, kuten esikyselyssäkin ilmenee – toimintaympäristö-teeman keskiarvo oli erittäin hyvä (4,25). Kehitystiimi oli siis hyvin itseorganisoitunut, joka näkyi muiden muassa ns. kiertävän Scrum-mestarin roolituksessa eli kaikki tiimin jäsenet olivat vuorollaan Scrum-mestarin roolissa. Yhteisen työympäristön, henkilöstön asiantuntevuuden ja kasvokkain toteutettujen tapaamisten nähtiin selkeästi hyödyntävän projektin osapuolten välistä kommunikaatiota. Toimiva kommunikaatio mahdollisti organisaatioiden välisen palautteen sekä kehityskohteiden nopeimman mahdollisen huomioimisen, kun asiakkaalta saatiin jatkuvia linjanvetoja, kommentteja sekä priorisointia toimittajan esittämiin ratkaisuihin. Hyötynä nähtiin myös se, että asiakastaho oli selkeästi omaksunut roolinsa osana kehitystiimiä ja oli selkeästi päättävä osapuoli. Kehityskohteena nähtiin kuitenkin organisaation arkaluontoisen toimialan myötä monimutkaiset sopimusprosessit ja se, ettei asiakastahon yksittäinen edustaja voinut toimia täysin valtuuksin, vaan päätösten läpivieminen saattoi olla pidempikestoinen prosessi.

Projekti 3:ssa suurin hyöty yksilöiden ja kanssakäymisen suhteen saavutettiin oikeanlaisella kommunikaatiolla, jossa asiakkaalla oli selkeästi valta ja toiminnot perustuivat asiakkaan näkemyksiin. Kommunikaatio oli toimittajan ja asiakkaan välillä ketterästä näkökulmasta katsottuna oikeanlaista, koska asiakastahon nähtiin olevan päättävä osapuoli ja sen oli mutkatonta lopulta esittää uudet tarpeensa. Projektissa tuotettiin perusjärjestelmien ja verkkopalveluiden versiopäivityksiä, jotka toteutettiin aiemmin kuvatusti Kanban-menetelmällä.

Yksilöiden ja kanssakäymisen teemassa projekti 3:ssa suurin heikkous oli ketteryydenvastaisesti toteutetut tapaamiset. Projektin kommunikointi toimittaja- ja asiakastahon välillä toteutettiin pääasiassa virtuaalisesti. Kommunikointiin ja työkokouksiin hyödynnettiin joko verkkoviestinjärjestelmää tai projektissa hyödynnettyä projektinhallintajärjestelmää, johon asiakas antoi kommentteja toteutuneesta työstä. Projektipäällikön näkemys oli, että virtuaalisesti toteutetut tapaamiset eivät olleet ketteryyden hyödyntämisen kannalta paras järjestely. Toimivammaksi olisi nähty kasvokkain tapaamiset, jolloin myös ketteryyttä ja olisi kyetty korostamaan ja saamaan suurin mahdollinen hyöty irti. Esikysely tuki edellä kuvattua, sillä asiakkaan läsnäolo oli kyselynkin perusteella erittäin hyvällä tasolla (keskiarvo 4,66) ja projekti koettiin toimintaympäristöllisesti hyvin (keskiarvo 3,5) toteutetuksi.

6.3.2 Toimiva ohjelmisto

Perinteisen ohjelmistokehitystyön on nähty olevan raskasta ja monimutkaista dokumentaation runsauden vuoksi. Agile Manifestossa (2001) ketterän ohjelmistokehityksen toinen arvo on toimivan ohjelmiston korostaminen dokumentaation sijaan. Ketterät periaatteet pyrkivät muiden muassa korostamaan sitä, että toimivat ohjelmistoversiot tulisi toimittaa asiakkaalle enimmillään kuukauden välein, että toimivan ohjelmiston tulisi olla ensisijainen laadun mittari ja, että teknisen laadun sekä hyvän rakenteen huomioinen edesauttaa ketteryyttä. Tämän teeman alle koottuja huomioita teemahaastattelusta on esitetty tässä luvussa ja havainnot on esitetty taulukossa 10.

Teema	Projekti 1	Projekti 2	Projekti 3
Toimiva ohjelmisto (dokumentaation sijaan)	- Toimitukset asiakkaalle 3 viikon välein - Ohjelmisto nähtiin parhaana laadun mittarina - Oma projektin hallintajärjestelmä selkeytti laadun ja hyvän rakenteen tarkkailua	- Julkaisut asiakkaalle maaliskuu, toukokuu, syyskuu - Paras mittari ohjelmiston hyväksymistestaus sekä asiakkaalta saatava jatkuva palaute - Laatu ja rakenne: sonar-mittaus, koodin tekninen mittaus, testiautomaatiot ja laadun verifiointi	- Sopiva toimitussykli 2 kuukautta versiotoimituksessa - Ensisijainen mittari aikataulussa pysyminen - Teknistä laatua ja rakennetta huomioidaan katselmoiden ja vaalien hyvää ohjelmointitapaa

Taulukko 10. Toimiva ohjelmisto tutkimuskohteena toimineissa projekteissa

Projekti 1:ssa toimivan ohjelmiston teemassa suurimpana hyötynä nähtiin toimivan ohjelmiston toimittaminen sopivien iteraatiosyklien välein, jolloin työt saatiin tehtyä oikeankokoisissa osissa. Projekti 1:ssa ohjelmistosta toimitettiin versioita asiakkaalle kolmen viikon välein, jota voidaan pitää ketterien arvojen ja periaatteiden perusteella hyvänä iteraatiosyklinä. Projektissa oli käytetty kokeiluina myös kahden ja neljän viikon pituisia iteraatioita, mutta kolmen viikon kesto oli kokemuksien perusteella paras, koska toteutettavat työt olivat laajuudeltaan oikeanlaisia kolmen viikon iteraatioihin. Projektin etenemistä mitattiin toimivalla ohjelmistolla, joka oli projektipäällikön näkemyksen mukaan hyvä mittari. Etenemistä seurattiin myös aikataulun avulla – kuinka hyvin suunnitelmassa pysyttiin. Luvussa 2.2 esitetyt sopimusprosessit olivat merkittävässä asemassa. Riippuvuus kolmansista osapuolista ja heidän asettamista aikatauluista nousi kehityskohteeksi sekä hankaloitti aikataulussa pysymistä. Kolmannet osapuolet ovat ns. monitoimittajalähtöisessä ohjelmistokehityksessä muita IT –palvelualan organisaatioita, jotka tuottavat kehitettäviin ohjelmistotuotteisiin joitain muita osakokonaisuuksia. Ohjelmistorakennetta ja laatua seurattiin projektissa hyödynnetyssä projektinhallintajärjestelmässä. Laadun parhaana mittarina pidettiin testausta sekä tiettyihin ohjelmiston osa-alueisiin tehtyä koodikatselmointia. On oleellista, että iteraatiopituudet nähtiin hyötynä, sillä esikyselyssä ilmennyt hyvä arvosana ensimmäisessä teemassa – toimitusstrategia – viittasi onnistunein väliajoin toimitettuun ohjelmistoon sekä tuoteominaisuuksien oikeanlaiseen priorisointiin.

Projekti 2:ssa toimivan ohjelmiston toimittamisella suurin saavutettu hyöty oli toimitusten jälkeen saadut asiakaspalautteet. Projektissa käytettiin toteutusympäristön mukaan parhaaksi todettua toimitussykliä – ohjelmistoa julkaistiin asiakastaholle epäsäännöllisesti kahden kuukauden ja neljän kuukauden välein. Julkaisutahti perustui

kustannustehokkuuden korostamiseen ja toimialalle ominaisten lainalaisuuksien projektiin vaikuttamiseen. Lainalaisesti toimimisen edellyttämät toimet ohjasivat juuri iteraatiopituuksia sekä monia muita asioita projektin toteutukseen liittyen. Toimivaa ohjelmistoa pidettiin toimivimpana mittarina – kunkin sprintin päätyttyä asiakkaan hyväksymät tulokset mitattiin ja tiimi sai konkreettista palautetta. Mittausten ja palautteen nähtiin kannustavan parempaan suunnitteluun sekä mahdollistavan tulosten täyttymisen arvioinnin jo ennen sprintin päättymistä. Ohjelmiston teknistä laatua ja hyvää rakennetta huomioitiin tekemällä mittauksia Sonar-työkalulla, teknistä koodimittausta sekä katselmointikäytännöin, testiautomaatioin ja laatua verifioiden.

Projekti 3:ssa ketterällä tavalla toteutettu ohjelmiston toimitus johti hyötyihin enimmäkseen aikataulujen seurattavuuden kannalta. Suurimmat haitat tässä teemassa olivat useiden toimittajien myötä esiintynyt aikataulujen venyminen sekä toimitussykliin vaikuttaneet sovellusalueelle ominaiset standardit ja lait. Projektissa koettiin ketterän periaatteen mukainen, enimmillään kuukauden välein tehtävä ohjelmiston toimitus liian tiheäksi toimintasykliksi. Toimialalle on ominaista useat toimittajaosapuolet yhtä asiakasorganisaatiota kohden, jolloin niin sanotut kolmannet osapuolet saattavat hidastaa toimitusten tekemistä. Projektipäällikön näkemyksen mukaan asiakastahon ennakointi ja aiempi suunnittelu kehitysprojekteissa voisi mahdollistaa tiheämmän toimitussyklin. Tässä projektissa versiotoimitusta tehtiin kahden kuukauden välein. Projektissa onnistumisen ensisijaisena mittarina toimi aikataulu. Versiotoimitusprojekteissa ilmenee projektipäällikön mukaan hyvin vähän virheitä, jonka vuoksi aikataulussa pysyminen luo selkeimmän kuvan onnistumisesta sekä etenemisestä. Teknistä laatua ja hyvää rakennetta huomioitiin projektissa korostamalla hyvää ohjelmointitapaa projektihenkilöstöä selkeästi ohjeistaen. Lisäksi koodia katselmoitiin suurien ohjelmistomuutosten sekä täysin uusien ohjelmistojen osalta.

6.3.3 Asiakasyhteistyö

Ketterässä ohjelmistokehityksessä pyritään poikkeuksetta alleviivaamaan läheistä yhteistyötä ohjelmoijien ja liiketoiminta-asiantuntijoiden välillä, ja ajatus näiden tahojen välisestä yhteistyöstä on yksi Agile Alliancen syntymisen taustalla merkittävimmin vaikuttavimmista tekijöistä (Martin 2003: 3). Asiakasyhteistyö on kuvattu ketterissä arvoissa kolmantena ja ketterissä periaatteissa se esitetään asiakkaan tarpeiden huomioimisena ja varhaisia ohjelmistoversioiden toimituksia sekä kestävää toimintatapaa

korostaen. Alla esitellään tutkimuskohteena olleiden projektien tapauksissa esiintyneitä näkökulmia asiakasyhteistyöstä, jotka ovat tiivistetty projekteittain taulukossa 11.

Teema	**Projekti 1**	**Projekti 2**	**Projekti 3**
Asiakasyhteistyö	- Asiakas esitti selkeästi tarpeensa projektin alussa - Ensimmäinen julkaisu asiakkaalle 2 viikon kuluessa - Asiakkaalta suullinen palaute - Lyhyt iteraatio etuna - Sopimusprosessit eivät tue ketteryyttä	- Asiakkaan esittämät mallikonseptit lähtökohtina tuotteelle - Asiakas antoi palautetta läpi sprinttien ja kommentoi demo-tilaisuuksissa aikaan saatua - Lyhyt iteraatio etuna - Toimintatapa hyvin kestävä, haasteet johtuvat hallinnollisista tekijöistä	- Asiakas esittää esiintyviä ongelmakohtia, jotka arvioidaan ja sikäli kun toteutuskelpoisia, tehdään määrittely - Ensimmäinen julkaisu asiakkaalle noin 4 viikon kuluessa - Versiot arvioidaan kyselyin - Lyhyt iteraatio ainoa tapa toimia - Toimintatapa ei toistaiseksi tunnu luonnolliselta

Taulukko 11. Asiakasyhteistyö tutkimuskohteena toimineissa projekteissa

Projekti 1:ssa asiakasyhteistyön kannalta suurimmat hyötytekijät olivat ohjelmiston varhainen toimitus asiakkaalle sekä lyhyet iteraatiot. Projektin lähtökohtana oli asiakkaan tekemät tilaukset. Asiakkaan kanssa oli tehty versiotoimitusten muodossa yhteistyötä aiemmin ja myöhemmin tarpeiden vaatiessa ryhdyttiin kehittämään kokonaan uutta tuotetta. Tämä uusi kehitysprojekti käynnistettiin priorisoimalla esitettyjä kehitysideoita. Nämä toimet kuvastavat selkeästi asiakkaan tarpeiden huomioimista ja niiden pohjalta aloitettua kehitystyötä. Projektin kohteena olevasta ohjelmistosta ja sen käyttöliittymästä tehtiin asiakkaalle ensimmäinen julkaisu noin viiden viikon kuluttua projektin alusta. Tähän jaksoon mahtui kaksi sprinttiä, joista ensimmäinen oli kestoltaan kaksi viikkoa ja sisälsi määrittelyiden tarkennuksia, teknistä suunnittelua sekä pohjatyötä. Toisessa kolmen viikon kestoisessa sprintissä tehtiin jo toteutusta, jonka jälkeen käyttöliittymää oli mahdollista ensi kertaa esittää asiakkaalle, jota voidaan pitää varhaisena toimituksena. Läpi projektin toimitukset tehtiin asiakkaalle kolmen viikon iteraatioissa ja asiakkaalta kerättiin palautetta suullisesti sekä projektin puolivälissä ohjelmistoa testattiin käytännössä. Projektipäällikön näkemyksen mukaan varhaisesta toimituksesta ja lyhyistä iteraatioista oli projektille merkittävää hyötyä – iteraatioiden loppuihin asetetut välitavoitteet selkeyttivät projektin etenemistä ja pienemmissä osissa toteutettava ohjelmisto edesauttaa asiakaspalautteen nopeaa saamista sekä helpottaa toiminnan suunnittelua toteutettavan kokonaisuuden ollessa suppeampi.

Projekti 1:ssa asiakasyhteistyö toimi hyvin, mutta projektipäällikkö löysi paljon kehitettävää teeman alla. Hän ei nähnyt tässä projektissa toteutunutta toimintatapaa kestävänä. Suurimpina kehityskohteina tämä näki sopimusten ketteryydenvastaisuuden, toimittajaorganisaation henkilöstön töiden päällekkäisyyden, asiakkaan passiivisuuden sekä kehityspäätöksissä esille nousseen päätöksentekovaltuuden puuttumisen. Globaali toimintaympäristö hankaloitti ketteryyttä kokonaisuutena sekä juuri työtahdin tasapainoisuutta ja toimintatapojen kestävyyden edistämistä.

Projekti 2:ssa asiakasyhteistyön ketteryyden suurimmat hyödyt olivat sujuva, yhteisen toimintalokaation mahdollistama työskentely sekä lyhyiden iteraatioiden mahdollistama palautteen saaminen. Projekti 2:ssa asiakkaan tarpeet olivat esitetty ennen projekti alkua selkeästi ja tämä oli antanut tiedoksi tavoitellun kokonaisuuden. Asiakastaho toimitti valmiit konseptit, jotka toimivat kehitystyön pohjana. Asiakkaan tarpeiden selvittyä, ensimmäinen julkaisu kohdetuotteesta julkaistiin asiakkaalle kolmen kuukauden kuluttua projektin alkamisesta. Jatkossa iteraatiopituudet eivät olleet säännöllisiä, mutta noin kolmen kuukauden pituisia. Asiakasyhteistyö oli projekti 2:ssa projektipäällikön näkemyksen mukaan erittäin sujuvaa, koska asiakas työskenteli läpi projektin samassa toimitilassa kehitystiimin kanssa. Tämä mahdollisti palautteen saamisen jo sprinttien aikana ja iteraation päätyttyä käytiin tuotteen demoversio läpi, jonka pohjalta asiakas antoi selkeän näkemyksensä sen hetkisestä tilanteesta ja tarpeiden täyttymisen tasosta. Lyhyistä iteraatiopituuksista nähtiin olevan suurta hyötyä projektin läpinäkyvyyden kannalta – koko kehitystiimillä, asiakas mukaan lukien oli selkeä käsitys siitä mitä ollaan tällä hetkellä tekemässä ja miten tulevaisuudessa tullaan etenemään. Projektin seurattavuus parantui myös merkittävästi iteraatioiden ollessa eheitä, lyhyempiä osakokokonaisuuksia. Projektipäälliköllä oli hyvin selkeä ja yksiselitteinen näkemys siitä, että projektissa toteutettu toimintatapa on kestävä ja ainoa oikea tapa toimittaa ohjelmistoa.

Projekti 2:ssa esiintyi myös asiakasyhteistyöhön perustuva heikkous. Haasteeksi muodostui asiakastahon – Scrumissa tuotteen omistajan – toimintavaltuuden rajallisuus sekä projektin riippuvuus muista samanaikaisista hankkeista. Toisena haasteena nähtiin hallinnollisten päätösten joustamattomuus perustuen muiden muassa lakeihin ja toimialan arkaluontoisuuteen.

Projekti 3:ssa suurin hyöty ketterällä asiakasyhteistyöllä saavutettiin lyhyiden iteraatioiden mahdollistamalla välittömällä palautteella sekä toimittajan ja asiakkaan välisellä luottamuksella. Kuten jo aiemmin esitettyjen teemojen pohjalta on ilmennyt, erosi projekti 3 muista kahdesta sekä käytetyn menetelmän, että projektin luonteen osalta. Asiakasyhteistyön osalta lähtökohta oli myös muihin tutkimuskohteina olleisiin projekteihin nähden erilainen. Versiotoimitusprojektit käynnistyvät tyypillisesti asiakkaan esittäminä ongelmina (*engl. issue*). Tässäkin projektissa asiakas esitti tarpeensa projektinhallintajärjestelmän kautta toimittajaorganisaatiolle ongelmina, jotka arvioitiin ensin karkealla tasolla. Sikäli, kun asiakas hyväksyi toteuttamiskelpoisiksi nähdyt ongelmat, tehtiin niiden pohjalta vaatimusmäärittelyt ja lopulta työsuunnitelmat. Tämän prosessin valmistuttua ja asiakkaan tarpeiden tällä keinoin selkeästi ilmettyä, otettiin kaikki valmiit ongelmat kerralla tuotantoon. Projekti 3:ssa ensimmäinen julkaisu tehtiin edellä kuvattujen vaiheiden jälkeen kolmen kuukauden kuluttua projektin alkamisesta. Palaute saatiin asiakkaalta luovutuskokeilla ja asiakkaan tekemien hyväksymistestien perusteella sekä kunkin toimitetun version päätteeksi asiakastaho vastasi kyselyyn, jolla saatiin kokonaisvaltainen kuva sisällön toimivuudesta sekä käytetyistä toimintatavoista. Projektipäällikkö piti asiakassuhdetta ainutlaatuisena ja poikkeuksellisena pitkän toimintahistorian myötä saavutetun luottamuksen ja läheisyyden vuoksi – palaute tai mahdolliset kehityskohteet saadaan selkeästi aina esille. Lyhyttä iteraatiopituutta projektipäällikkö piti parhaana tapana toimia, koska projektin seurattavuus on hyvin vaivatonta ja selkeää.

Projekti 3:ssa asiakasyhteistyötä hankaloitti henkilöstön töiden päällekkäisyys sekä ketterän toimintatavan uusi ja tuntematon luonne. Toimittajaorganisaatiossa on tyypillistä, että sama henkilöstö on samanaikaisesti mukana useissa projekteissa, jolloin pienemmissä kokonaisuuksissa tehty työ on paras vaihtoehto toimia. Tämä projektien päällekkäisyys korostaa projektijohdon ja kehitystiimin välistä suhdetta – johdon tulee olla tietoinen kehitystiiminsä päällekkäisistä töistä individuaalitasolla. Tämänhetkisten toimintatapojen kestävyyden – Kanban-menetelmän ja lyhyiden toimitussyklien – projektipäällikkö näki positiivisena ja koki, että toimintatapa voi vakiintua. Ketterän toimintatavan omaksuminen ei kuitenkaan ole ollut täysin mutkatonta ja uusi menetelmä on aiheuttanut hämmennystä, koska perinteisten toimintatapojen on nähty toimivan aiemmissa versiotoimitusprojekteissa hyvin. Ketterien toimintatapojen myötä projektipäällikkö kokee toistaiseksi projektien monimutkaistuneen yksinkertaistumisen sijaan ja ketteryys on tuonut projekteihin ylimääräistä byrokratiaa.

6.3.4 Muutokseen vastaaminen

Yksi neljästä ketterästä arvosta on pyrkimys muutokseen vastaamiseen ennemmin kuin suunnitelmissa pitäytymiseen. Tällä viitataan ketterissä periaatteissa suhtautumiseen myönteisesti myöhäisiä vaatimuksia kohtaan, joita väistämättä jokaisessa ohjelmistokehitysprojektissa esiintyy sekä asiakkaan kilpailukyvyn maksimointiin uusia teknologioita hyödyntäen tämän liiketoiminnassa ja yksinkertaisuuden korostamiseen projektitoiminnassa. Yksinkertaisuuden korostamisella viitataan pyrkimykseen maksimoida tekemättä jätettyä työtä. Tutkimuksessa ilmenneet pääkohdat muutokseen vastaamisen teemassa ovat esitetyt projekteittain taulukossa 12.

Teema	**Projekti 1**	**Projekti 2**	**Projekti 3**
Muutokseen vastaaminen	- Vaatimusmuutokset selkeitä, esitettiin demo-tilaisuuksissa, asiakkaan toiveisiin reagoitiin - Arkkitehtuuria vaikea muokata tämän kaltaisessa projektissa	- Vaatimusmuutokset luonnollisia ja jatkuvia, suhtautuminen uusiin vaatimuksiin positiivista - Ohjelmistoon hyödynnetty MCP-platformia uutena teknologiana - Yksinkertaisuutta korostettiin kaikin mahdollisin keinoin	- Vaatimusmuutoksiin suhtauduttiin positiivisesti - Kolmannet osapuolet vaikeuttavat myöhäisiä muutoksia - Ylimääräinen dokumentaatio yksinkertaistamisen tiellä

Taulukko 12. Muutokseen vastaaminen tutkimuskohteena toimineissa projekteissa

Projekti 1:ssa muutokseen vastaamisessa nähtiin saavutettavan hyötyä asiakasvaatimusten mukaisen tuotteen rakentamisessa. Vaatimusmuutoksiin suhtauduttiin positiivisesti ja niiden koettiin olevan osa nykyaikaista projektityötä. Muutoshallinta toimi projektipäällikön mukaan perinteiseen tyyliin – ilmenneet muutostarpeet muotoiltiin virallisiksi muutoksiksi, joista tehtiin työarviot ja annettiin asiakkaan päätettäviksi. Tässä projektissa mahdolliset muutostarpeet ilmenivät demovaiheessa, kun toteutusta esiteltiin asiakkaalle sprinttien jälkeen. Toimittajaorganisaatio esitti myös asiakkaalle muutosehdotuksia tuotteen yksinkertaistamiseksi tai suoraviivaistamiseksi. Mahdolliset määrittelyissä ilmenneet ristiriidat pyrittiin myös korjaamaan. Projektin aikana ilmeni joitakin arkkitehtuuritason muutosehdotuksia, joihin ei voitu vastata enää projektin myöhemmässä vaiheessa johtuen aiemmin tehdyistä määrittelyistä ja projektin toimintaympäristön arkaluontoisuudesta. Uutta teknologiaa – kuten big dataa, sosiaalista mediaa tai teollista internetiä – ei tässä projektissa kyetty hyödyntämään johtuen projektin luonteesta, toimialan vaatimasta autentikoinnista sekä kohderyhmästä.

Aiempien teemojen tapaan, myös muutokseen vastaamisen tapauksessa, oli projekti 3:ssa globaalista toimintalokaatiosta haittaa ketteryyden toteutumiselle. Yksinkertaisuuden korostaminen oli projektille vaativaa. Ennalta laaditut tiukat määrittelyt sotivat projektipäällikön näkemyksen mukaan yksinkertaisuuden korostamista vastaan. Toimivampi ratkaisu olisi ollut tämän mukaan ennalta määritellyt vaatimukset, mutta tarkemman tason määrittelyiden laatiminen projektin edetessä, joka olisi mahdollistanut niin sanotun hyödyttömän työn pienemmän määrän. Myös globaalin kehitystiimin nähtiin tuottaneen ylimääräistä työtä, mikä hankaloitti yksinkertaisuuden korostamista. Toimittajaorganisaation sisäiset tekniset prosessit erityisesti dokumentaation kannalta hankaloittivat yksinkertaisuuden korostamista ja aiheuttivat projektipäällikön näkemyksen mukaan ylimääräistä työtä. Projektin edetessä koettiin, että toimintatavat kehittyivät ja ylimääräistä työtä kyettiin karsimaan, mutta ajatus toiminnan yksinkertaisuudesta koettiin etenkin projektin alussa monimutkaiseksi.

Projekti 2:ssa muutokseen vastaamisen korostamisen suurimmat saavutetut hyödyt olivat sen mahdollistama uuden teknologian hyödyntäminen sekä yksinkertaisuuden korostamisen tulokset. Projekti 2:ssa vaatimusmäärittelyt oli laadittu asiakkaan kanssa projektinhallintajärjestelmään ja vaatimusmuutokset projektin aikana olivat varsin luonnollisia – vaatimusten tiedettiin ja oletettiin elävän projektin edetessä. Vaatimusmuutoksiin törmättiin projektipäällikön mukaan jatkuvasti ja ne olivat usein laajuudeltaan varsin suuria, koska asiakasorganisaatiolla ei ollut lähtökohtaisesti ollut resursseja katselmoida määrittelyjä riittävällä tasolla, joten muutostarpeet todettiin vasta projektin aikana. Muutoksiin suhtauduttiin kuitenkin positiivisesti. Uutta teknologiaa hyödynnettiin projektissa toimittajaorganisaation kehittämällä ohjelmistoalustalla sekä applikaatiokokonaisuudella, joiden projektipäällikkö näki edustavan juuri uusia ja asiakkaan kilpailukykyä edistävinä teknologioina. Yksinkertaisuus oli projektissa hyvin merkittävässä asemassa ja projektipäällikkö näki suurimpana, yksittäisenä yksinkertaisuutta korostavana esimerkkinä projektissa aikaansaadun kehitystyön sekä ylläpitotyön yhtenäistämisen yhdeksi kokonaisuudeksi.

Muutokseen vastaamista hankaloittava tekijä projekti 2:ssa oli ketterää kehitystä sekä muutosta tukematon toimintaympäristö. Projektissa pyrittiin erityisesti kehitystiimin toiminnan tasolla korostamaan yksinkertaisuutta kaikin mahdollisin keinoin, mutta toimintaympäristö koettiin yksinkertaisuutta hankaloittavaksi. Yksittäisten työkoko-

naisuuksien hyväksyttämisprosessit asiakasorganisaatiossa ovat monimutkaiset ja projektipäällikkö näki niiden aiheuttavan kohtuuttomia työmääriä. Päätösten kierrättäminen asiakasorganisaation ohjausryhmässä saattoi kestää useita päiviä ja siten hankaloittaa itse kehitystiimin työtä merkittävästi. Toimintaa merkittävästi kehittävänä toimena projektipäällikkö pitäisi sopimusprosessien viemistä ketteryyttä tukevaan suuntaan, joka mahdollistaisi yksinkertaisuuden korostamisen.

Projekti 3:ssa muutokseen vastaamisella saavutettu suurin hyöty oli uuteen teknologiaan perustunut tilaustenhallintajärjestelmä, jota projektissa käytettiin. Projektissa suhtauduttiin vaatimusmuutoksiin ja myöhäisiin vaatimuksiin edelleen positiivisesti, tosin Kanban-menetelmässä muutokset voitiin huomioida vain sopivina hetkinä – ennen iteraatiota tai sen jälkeen – toimitettuina. Tämä edellytti asiakkaalta riittävän ajoissa liikkeellä olemista muutosten suhteen ja ajoittain projektin aikana kohdattiin tilanteita, joissa asiakastaho ei ollut riittävän ajoissa liikkeellä vaatimusmuutosten suhteen. Projektipäällikkö näki, että suhtautumista myöhäisiin vaatimuksiin voisi kehittää aikaistamalla asiakastahon muutosehdotusaikatauluja sekä parantamalla kommunikaatiota selkein palauttein iteraatioiden jälkeen. Tällöin myös asiakas olisi selkeästi tietoinen, miksi johonkin tiettyyn myöhäiseen muutosehdotukseen ei enää kyetty reagoimaan. Versiotoimitusprojektit ovat usein riippuvaisia jo aiemmin esiin nousseista kolmansista osapuolista eli asiakkaan kanssa yhteistyötä tekevistä muista toimittajaorganisaatioista. Tämä aiheuttaa vaatimusmuutoksia usein ja niihin suhtaudutaan tutkimuskohteena olleessa toimittajaorganisaatiossa rajallisesti sekä niitä pyritään minimoimaan, koska nämä muutokset tehdään niin sanotusti projektiversion ulkopuolisena työnä. Ulkopuolisella viitataan työhön, jota ei ole huomioitu projektia suunniteltaessa. Tämä kaikki edellä kuvattu korostaa projektipäällikön mukaan asiakkaan oikeanlaisen aikataulusuunnittelun tärkeää merkitystä. Merkittävin uusi käytössä ollut teknologia on tässä projektissa ollut projektipäällikön näkemyksen mukaan tilaustenhallintajärjestelmä, joka on edistänyt kommunikaatiota toimittajan ja asiakkaan välillä henkilökohtaisten sähköpostiviestien kaltaisten viestintekniikoiden siirryttyä yhteen paikkaan, jonne sekä asiakkaalla että toimittajalla on suojattu pääsy. Yksinkertaisuuden korostuminen näkyy projekti 3:ssa projektipäällikön mukaan siinä, että ketterään toimintatapaan siirtymisen myötä yhdellä projektipäälliköllä on vastuullaan vain yksi versioprojekti kerrallaan, johon on pakattu aiemmin erillisinä projekteina toteutetun sovellukset. Tämä parantaa aikataulujen pitävyyttä. Ongelmaksi on koettu ketteryyden

myötä lisääntynyt dokumentaatio, joka ei vastaa Agile Manifestossa esitettyjä periaatteita. Lisääntynyt dokumentaatio on syntynyt projektipäällikön mukaan iteraatioiden jälkeen pidetyistä jälkikatsauksista sekä erilaisista palautetilaisuuksista.

6.4 Yhteenveto tutkimustuloksista

Tässä luvussa esitetään yhteenveto tämän tutkimuksen tuloksista perustuen tutkimuksen tavoitteisiin ja esitettyihin tutkimuskysymyksiin. Tutkimuksen tavoitteena oli saada selville, kuinka IT -ratkaisuja –ja palveluita tarjoava organisaatio hyödyntää ketterän ohjelmistokehityksen arvoja ja periaatteita ohjelmistoprojekteissa käytännössä. Tutkielman päätavoitteena oli selvittää mitä hyötyjä tutkimuksen kohteena oleva organisaatio voi saavuttaa ketterillä toimintatavoilla ja toisaalta, voiko ketteryys muodostua haitaksi tai uhaksi organisaatiolle.

Tutkimuksessa selvitettiin, mitä ketteriä menetelmiä kohdeorganisaatiossa käytetään ja miten se hyödyntää niitä erilaisissa projekteissa. Selvitettiin myös mahdolliset käytössä olevat omat menetelmät ja niiden suhde yleisiin käytettyihin menetelmiin. Edellä esitetyn tutkimusaineiston mukaisesti kohdeorganisaatiossa käytetään ketteristä menetelmistä Scrumia ja Kanbania. Kohdeorganisaation voidaan selkeästi nähdä hyödyntävän eri menetelmiä erilaisissa projekteissa. Aineiston muodostaneissa projekteissa oli käytössä Kanban ja Scrum, joiden lisäksi kohdeorganisaatiossa on käytössä myös oma menetelmä. Tämän oman menetelmän voidaan nähdä aineiston perusteella onnistuneen heikoiten sovellusalueensa puolesta johtuen sen globaalista toimintaympäristöstä, joka vaikeutti ketterästi toimista. Globaali toimintaympäristö oli esteenä täysin ketterästi toimimiselle, mutta ketterä toimintatapa koettiin potentiaaliseksi ja voisi oikeassa ympäristössä toimia paremmin. Tämä hyödynnetty oma menetelmä muistutti hyvin paljon Scrumia perustuen muiden muassa sprinttilähtöisyyteen, mutta esimerkiksi suuri projektihenkilöstön määrä teki siitä omanlaisensa.

Ketterien menetelmien ajatellaan syntyneen vastavetona niin sanotuille perinteisille menetelmille ja niillä pyritään eheyttämään ohjelmistokehitysorganisaation kykyä havainnoida ja vastata nopeasti teknisiin muutoksiin ja uusiin liiketoiminnallisiin mahdollisuuksiin pyrkien parantamaan asiakkaan kilpailukykyä. Ketterien arvojen ja periaatteiden hyödyntäminen ohjelmistokehitysprojekteissa voi olla vaiheittaista ja lähteä liikkeelle jonkin tietyn yksittäisen ketterän käytännön omaksumisesta. Kun ketteriä

käytäntöjä alkaa ilmetä ajan myötä projekteissa enemmän, nähdään niiden ennemmin tai myöhemmin edistävän projektin tarkoituksen mukaisuutta sekä asiakkaan odotusten täyttämistä. Tämän tutkielman aineiston muodostaneissa ohjelmistokehitysprojekteissa oli saavutettu ketterillä arvoilla ja periaatteilla konkreettisia hyötyjä, mutta myös tuloksissa ilmeni vastarintaa ketteriin toimintatapoihin siirtymistä kohtaan sekä ketteryyttä jopa kyseenalaistettiin joihinkin projekteihin sopivana toimintamallina. Yhteenvetoa tuloksista kuvataan seuraavissa kappaleissa kunkin ketterään arvoon perustuvan tutkimusteeman mukaan.

Yksilöiden ja kanssakäymisen kannalta jokaisessa projektissa saavutettiin hyötyä kommunikaation korostamisesta sekä asiakkaalta suoraan saadusta palautteesta, joka edisti projektin etenemistä ja asiakkaan näkemysten ilmenemistä. Myös tiimien itseorganisoituneisuus nähtiin merkittävänä hyötynä, mikä perustui pitkälti kehitystiimien ja niihin kuuluneiden yksilöiden ammattitaitoon sekä tietämykseen. Projekteissa 2 ja 3 oli asiakas selkeästi omaksunut asemansa osana kehitystiimiä ja projektipäälliköt näkivät, että asiakkaat olivat selkeästi valta-asemassa sekä kaikki päätökset perustuivat heidän tarpeisiinsa ja päätöksiinsä. Tämä hyödyttää asiakkaan tarpeisiin vastaamista ja tarpeiden mukaisen tuotteen rakentamista. Projekti 1:ssa projektipäällikön näkemyksen mukaan asiakkaan aktiivisuus ei ollut riittävällä tasolla, joka hankaloitti ketterästi toimimista sekä erityisesti asiakkaan tarpeiden ilmi saamista.

Ketterän toimintatavan merkittävimpiin tekijöihin kuuluu toimivan ohjelmiston toimittaminen asiakkaalle säännöllisin väliajoin ja tämän tutkielman aineiston toisen teeman muodosti juuri toimiva ohjelmisto. Tutkimusaineiston projekteissa suurimmat hyödyt saavutettiin ohjelmiston toimittamisessa lyhyissä iteraatioissa. Hyödyt perustuivat jatkuvasti saatavaan, iteraatioden jälkeiseen palautteeseen. Iteraatiosykli vaihteli projektikohtaisesti, mutta perustui kussakin projektissa projektin luonteeseen huomioiden asiakkaan tarpeet sekä toimialalla vallitsevat lainalaisuudet. Toinen merkittävä hyöty saavutettiin sillä, että toimiva ohjelmisto toimitettiin osissa ja säännöllisesti asiakkaalle. Tämä edistää asiakkaan tarpeiden mukaisen tuotteen rakentamista, koska säännöllistä palautetta sekä kehitysehdotuksia saadaan läpi projektin. Tämä toteutui kaikissa kolmessa projektissa ja jatkuva palaute nähtiin jopa merkittävimpänä hyötynä kaikissa kolmessa projektissa. Ohjelmiston säännöllinen toimittaminen mahdollistaa myös laadun ja rakenteen huomioimisen läpi projektin ja jokaisessa projektissa tätä ketterää arvoa hyödynnettiin erilaisin testaus– ja katselmointimenetelmin.

Asiakasyhteistyö ketterissä menetelmissä perustuu alkujaan siihen, että nimenomaan asiakkaan tarpeet toimivat kaiken lähtökohtana ja koko projekti rakennetaan niihin perustuen. Tutkielman aineistona toimineet projektit olivat kaikki juuri edellä kuvatusti rakennettu – perustuen asiakkaan esittämiin tarpeisiin. Kaikissa projekteissa nähtiin, että lyhyissä iteraatioissa tehty ohjelmiston toimitus edisti asiakasyhteistyötä johtuen asiakkaalta usein saadusta palautteesta. Palautteen saaminen olikin lyhyiden iteraatioiden myötä saavutettu merkittävin hyöty. Asiakasyhteistyö-teeman alla selvitettiin myös projektipäällikön näkemys käytetyn menetelmän ja ketterän toimintatavan kestävyydestä. Projekti 2 oli ainoa, jossa ketterä toimintatapa nähtiin ainoana oikeana tapana tehdä ohjelmistokehitystä, kun taas projekteissa 1 ja 3 ilmeni kyseenalaiseksi jääneitä näkemyksiä siitä, miksi toimitaan ketterästi. Projekti 1:ssa ongelmaksi muodostui eritoten koko kohdeorganisaation projektihenkilöstön kokemattomuus ketteristä menetelmistä sekä globaali toimintaympäristö. Lisäksi kyseessä ollut projekti oli rakennettu ketteryyttä tukemattomalla sopimuksella. Projekti 3:ssa taas Kanban-menetelmä toimi kohtalaisesti, mutta projektipäällikkö ei nähnyt syytä, miksi aiemmasta, perinteisestä toimintatavasta haluttiin siirtyä ketterään, koska sen nähtiin toimineen hyvin ja nyt koettiin, että projektin toimittaminen vain mutkistui.

Muutokseen vastaaminen oli tutkielmassa toteutetun teemahaastattelun neljäs teema ja on edelleen hyvin olennainen osa ketterää ajattelua. Teeman alla merkittävimpänä nähdään projektin aikana muuttuvat vaatimukset sekä ympärillä vallitsevan muutoksen hyödyntäminen asiakkaan kilpailuedun kasvattamiseksi. Kaikissa tutkimusaineiston muodostaneissa projekteissa vaatimusmuutokset projektin aikana koettiin positiivisina ja niiden nähtiin hyödyttävän toimittajaorganisaation kykyä vastata asiakkaan tarpeisiin. Merkittävässä asemassa muutokseen vastaamisessa ketterässä ympäristössä on muuttuvan ja jatkuvasti uudistuvan teknologia hyödyntäminen ja sen muuttaminen edelleen asiakkaan kilpailuedun kasvattamiseksi. Tutkituissa projekteissa ei kyetty hyödyntämään sosiaalisen median, big datan tai teollisen internetin kaltaisia teknologioita johtuen projektien luonteesta ja kehitetyistä ohjelmistotuotteista. Kehityksen kohteena olivat arkaluontoista dataa sisältävät järjestelmät, jolloin voitiin pitää odotettuna, ettei muiden muassa sosiaalisen median hyödyntäminen ole mahdollista. Kaikkien projektien hallinnassa hyödynnettiin uutta hallintajärjestelmää, joka oli uusi jokaiselle projektipäälliköille ja sen koettiin hyödyttäneen juuri projektien hallinnan yksinkertaisuutta.

Tutkimusaineiston muodostaneet projektit oli ennalta pyritty valitsemaan perustuen ketterästä näkökulmasta paremmin ja heikommin onnistuneisiin projekteihin. Tutkielmassa aiemmin esitetty esikysely antoi selkeän kuvan siitä, kuinka ketteriä projektit todellisuudessa olivat. Kyselyn pohjalta oli relevanttia tuottaa teemahaastattelu, jossa ilmeni huomattavasti yksityiskohtaisempia näkemyksiä projektien onnistuneisuudesta. Projekti 1:ssa ilmeni selkeästi pienin määrä Agile Manifestossa esitettyjä ketteriä arvoja ja periaatteita, jota tuki myös esikyselyn teemojen keskiarvot. Merkittävimpinä ongelmina projektin ketteryyden kannalta olivat juuri hajautunut, globaali toimintaympäristö, ketteryyttä tukematon sopimus sekä henkilöstön ketterä kokemattomuus ja tietämyksen puute. Projekti 2 oli taas selkeästi onnistunein, josta antoi evidenssiä esikyselyn tulokset sekä jokaisen teemahaastattelun teeman alla tehdyt havainnot ketterien arvojen ja periaatteiden voimakkaasta ilmentymisestä. Projekti 3 oli onnistunut, mutta tutkimuksen perusteella jäi kyseenalaiseksi, oliko ketterä toimintatapa juuri tätä ympäristöä tukeva ja oliko projektihenkilöstö valmis muutokseen perinteisestä ketterään.

Kaikkien projektipäälliköiden näkemyksiin perustuen on merkittävää tuoda esiin, että toimittaessa toimialalla, jossa asiakasorganisaatioiden hallussa on arkaluontoista, laein ja standardein suojattua dataa, ei ketterään toimintatapaan siirtyminen ole nopea tai yksinkertainen prosessi. Kehitettävät ohjelmistotuotteet ovat poikkeuksetta riippuvaisia ns. kolmansista osapuolista eli muista ohjelmistoa asiakasorganisaatioille tuottavista toimittajaorganisaatioista. Tämänkaltaiset riippuvuudet muihin projekteihin eivät tue ketteryyttä. Toisekseen ongelmana ilmeni kohdeorganisaation henkilöstön kuormitus – useat asiantuntijat ovat samanaikaisesti mukana useammissa projekteissa, jolloin täydellinen panostaminen yhteen projektiin on mahdotonta. Havaittiin myös, että näkemys ketteryydestä tulisi olla yhdenmukainen sekä asiakasorganisaatiolla että toimittajana toimivalla osapuolella ja molempien organisaatioiden tulisi olla henkilöstöstä johtoon halukkaita siirtymään ketterään toimintatapaan. Mahdollinen tietämyksen puute aiheuttaa juuri tämänkaltaisia ristiriitoja. Kolmanneksi projektin lähtökohtana toimivat sopimusprosessien tuloksena syntyvät projektikohtaiset sopimukset tulisi rakentaa ketteryyttä tukeviksi. Ongelmia sopimusten suhteen on tutkituissa projekteissa ilmennyt muiden muassa asiakasorganisaatiota edustavan tuotteen omistajan vajaavaltaisuudessa – toiminta– ja päätösvalta eivät olleet täydellisiä – sekä sopimus tulisi rakentaa sellaiseksi, että sitä on ketteryyden edellyttämällä tavalla mahdollista muokata projektin edetessä.

Ketteryys on tutkimuksen perusteella hyödyttänyt ohjelmistokehitystoimintaa tutkielman kohdeorganisaatiossa. Merkittävimpinä hyötyinä koettiin ketteryyden mahdollistama osa-kokonaisuuksien suunnittelu, kun tuotetaan projekteja iteraatioissa, organisaation kehittyminen ja projektihenkilöstön – asiakkaan sekä toimittajan – oppiminen sekä kyky ammentaa tietämystä projektien aikana. Lisäksi merkittäväksi hyödyksi muodostui asiakkaalta jatkuvasti saatavan palautteen mahdollistamat edut sekä toimittajaorganisaation tiimien monipuolisuus niiden ollessa ketteriä – tiimit olivat kokonaisvaltaisia ja asiantuntevia useista näkökulmista.

7 JOHTOPÄÄTÖKSET

Tutkimuksen taustalla oli tarkoitus selvittää, kuinka IT -ratkaisuja –ja palveluita tarjoava organisaatio hyödyntää ketterän ohjelmistokehityksen arvoja ja periaatteita ohjelmistoprojekteissa käytännössä. Selvitettiin, millaisia menetelmiä kohdeorganisaatio käytti kolmessa tarkastelun kohteena olleessa ohjelmistokehitysprojektissa. Tutkielman päätavoitteena oli selvittää mitä hyötyjä tutkimuksen kohteena oleva organisaatio voi saavuttaa ketterillä toimintatavoilla ja toisaalta, voiko ketteryys muodostua haitaksi tai uhaksi organisaatiolle. Tutkielman tutkimuskohteena toimi kolme erilaista, ketterästi toteutettua ohjelmistokehitysprojektia. Tutkielman aineisto hankittiin kolmelle projektipäällikölle esitetyistä esikyselystä sekä teemahaastattelusta.

Tutkimuksessa esitettiin teemoitteluun perustuen, mitä hyötyjä ketteristä toimintatavoista ja menetelmistä tällä hetkellä kohdeorganisaatiolle on. Lisäksi tuotiin esiin merkittävimmät kehityskohteet tämänhetkisissä toimintatavoissa. Merkittävimmin kohdeorganisaatio hyötyy ketteryydestä kyetessään tuottamaan asiakkaan vaatimusten ja tarpeiden mukaisia ohjelmistotuotteita sekä kyetessään hallinnoimaan projekteja selkeinä osakokonaisuuksina iteraatiolähtöisessä ohjelmistotuotannossa. Iteraatiot mahdollistavat työn suunnittelun osissa ja asiakkaalta jatkuvasti saatavan palautteen. Kohdeorganisaatio toimii toimialalla, jossa ketteryyteen siirtyminen ja ketterien menetelmien soveltaminen ei ole yksiselitteistä tai nopeaa, mutta siirtyminen ketteryyteen on organisaatiolle kuin organisaatiolle vaiheittain tapahtuva prosessi.

Tutkimustuloksia voidaan pitää odotettuina, mutta tuloksiin tulee suhtautua varauksella. Tutkielman aineisto muodostui vain kolmesta projektista, joten otosta voidaan pitää kohtalaisen pienenä. Toisaalta projektit olivat tarkoin valitut, täysin erilaiset ja niiden pohjalta kyettiin esittämään, että ketterä toimintatapa on sovellettavissa erilaisissa ympäristöissä, erilaisin menetelmin. Jatkotutkimuksena tulisi tutkia, kuinka organisaatio kykenee tulevaisuudessa viemään projektinhallintaansa ketterämpään suuntaan sekä kykeneekö se tulevaisuudessa soveltamaan ketteriä arvoja ja periaatteita enemmän Agile Manifeston mukaisesti projekteissaan. Lisäksi jatkotutkimuksena voisi selvittää kuinka ketteryyden mahdollistamista voisi edistää toimialalla, jolla sen hyödyntäminen lähtökohtaisesti on vaativaa.

LÄHDELUETTELO

Abrahamsson, P., O. Salo, J. Ronkainen & J. Warsta, (2002). *Agile software development: Review and analysis*. VTT Publications 478, Technical Research Centre of Finland, Espoo, Finland.

Agile Alliance (2001). *Agile Manifesto* [online]. [Lainattu 17.3.2015] Saatavilla: http://agilemanifesto.org/

Ahmad, M. O., Markkula, J. & Oivo, M. (2013). Kanban in software development: A systematic literature review. In: *Software Engineering and Advanced Applications (SEAA)*, 2013 39th EUROMICRO Conference on (pp.9-16). IEEE.

Aitken, A. & Ilango, V. (2013). A Comparative Analysis of Traditional Software Engineering and Agile Software Development [online]. Teoksessa: *2013 46th Hawaii International Conference on System Sciences*, pp. 4751–4760. IEEE Computer Society.

Atlassian (2015). A brief introduction to kanban [online]. Saatavilla: https://www.atlassian.com/agile/kanban

Awad, M. A. (2005). *A comparison between Agile and Traditional Software Development Methodologies* [online]. The University of Western Australia. Saatavilla: https://xa.yimg.com/kq/groups/71601056/324265895/name/A_comparision_between_Agile_and_Traditional_SW_development_methodologies.pdf

Basili, V. R. & Larman, C. (2003). *Iterative and Incremental Development: A Brief History* [online]. IEEE Computer Society. Saatavilla: http://www.craiglarman.com/wiki/downloads/misc/history-of-iterative-larman-and-basili-ieee-computer.pdf

Beck, K. & Fowler, M. (2000). *Planning Extreme Programming*. Addison Wesley. ISBN 0-201-71091-9. 160 s.

Beck, K. (1999). *Extreme Programming Explained: Embrace change*. First Edition, September 29, 1999. ISBN 0201616416. 224 s.

Chow, T. & Cao, D. (2007). A survey study of critical success factors in agile software projects. In: *The Journal of Systems and Software 81 (2008).* PP 961-971.

Cockburn, A. (2000). *Agile Software Development*. Draft version: 3b. [Lainattu 17.3.2015]

Conboy, K., (2002). Agility from first principles: reconstructing the concept of agility in information systems development. *Information Systems Research*, 20 (3), 329-354. Informs.

Haikala, I. & Mikkonen, T. (2011). *Ohjelmistotuotannon käytännöt*. 12. uudistettu painos. Hämeenlinna: Talentum Media Oy. 242 s. ISBN 978–952–14–1754–2.

Highsmith, James, A. (2002). *Agile Software Development Ecosystems*. Pearson Education Inc. ISBN 0-201-76043-6.

IEEE Std 15288™, (2008). *Systems and software engineering – System life cycle processes*. The Institute of Electrical and Electronics Engineers, Inc.

ISO/IEC 15288:2002, (2005). *Systems Engineering-System Life Cycle Processes*. The Institute on Electrical and Electronics Engineers, Inc.

Jyväskylän yliopiston Koppa (2015). *Teemoittelu* [online]. Saatavilla: https://koppa.jyu.fi/avoimet/hum/menetelmapolkuja/menetelmapolku/aineiston-analyysimenetelmat/teemoittelu

Järvinen, P. & Järvinen, A., (2011). *Tutkimustyön metodeista*. Tampere: Opinpajan kirja. ISBN 978-952-99233-4-2.

Kniberg H. & Skarin, M. (2010). *Kanban and Scrum – Making the Most of Both*. C4 Media Inc. ISBN 978-0-557-13832-6.

Lyytinen, K. & Rose, G. (2006). Information systems development agility as organizational learning. *European Journal of Information Systems* 15(2), 183-199.

Martin, Robert C., (2003). *Agile development: principles, patterns and practices*. Upper Saddle River: Pearson Education, Inc. ISBN 0-13-597444-5.

Oxford Dictionaries (2015). *Definition of agile in English* [online]. [Lainattu: 17.3.2015] Saatavilla: http://www.oxforddictionaries.com/definition/english/agile

Royce, W. (1970). Managing the development of large software systems: Concepts and techniques. In: *ICSE '87 Proceedings of the 9th international conference on Software Engineering*. PP 328-338.

Rubin, K., (2013). *Essential Scrum: a practical guide to the most popular agile process*. Pearson Education, Inc. ISBN-13: 978-0-13-704329-3.

Schuh, Peter., (2005). *Integrating agile development in the real world*. Charles River Media. ISBN 1-58450-364-5.

Schwaber, K. & Sutherland, J. (2013). *The Scrum Guide™ – The Definitive Guide to Scrum: The Rules of the Game* [online]. Scrum.org. Saatavilla: http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf#zoom=100

Schwaber, K. (2004). *Agile Project Management with Scrum* [online]. Microsoft Press. ISBN 0-7356-1993-X. Saatavilla: https://spemarti.google-code.com/files/Schwaber2004.pdf

Sugimori, Y., Kusunoki, K., Cho, F., Uchikawa, S., (1977). *Toyota production system and Kanban system Materialization of just-in-time and respect-for-human system* [online]. International Journal of Production Research,

15, 553 – 564. Saatavilla: http://www.tandfonline.com/doi/pdf/10.1080/00207547708943149

VersionOne (2015). *9th Annual State of Agile Development Survey* [online]. Saatavilla: http://info.versionone.com/state-of-agile-development-survey-ninth.html f

VersionOne (2015). *Kanban Board Software* [online]. Saatavilla: http://www.versionone.com/product/kanban-board/

Wells, D. (2013). *Extreme programming: a gentle introduction* [online]. [Lainattu: 6.5.2015]. Saatavilla: extremeprogramming.org

Wells H., Dalcher, D. & Smyth, H. (2015). *The adoption of agile management practices in a traditional project: An IT/IS Case Study* [online]. In: 2015 48th Hawaii International Conference on System Sciences, pp. 4446–4453. IEEE Computer Society. Available: http://conferences.computer.org/hicss/2015/papers/7367e446.pdf

LIITTEET

LIITE 1. Esikysely

Henri Nyholm
Pro gradu -tutkielma

TAPAUSTUTKIMUS: MITEN IT-ORGANISAATIO HYÖDYNTÄÄ KETTERÄN KEHITYKSEN ARVOJA JA PERIAATTEITA OHJELMISTOPROJEKTEISSA?

TEEMAKYSELY – PROJEKTIN ONNISTUMINEN

Seuraavat 6 teemaa kysymyksineen ovat ketterän ohjelmistoprojektin onnistumisen kannalta keskeisimmät. Valitse vaihtoehdoista 1-5, kuinka ne toteutuivat omassa projektissanne?
(1 erittäin huonosti, 2 jokseenkin huonosti, 3 en osaa sanoa, 4 jokseenkin hyvin, 5 erittäin hyvin)

TEEMA 1: TOIMITUSSTRATEGIA

	1	2	3	4	5
Projektin kohde toimitettiin asiakkaalle vaiheissa, säännöllisin väliajoin					
Tärkeimmät tuoteominaisuudet toimitettiin ensin					

TEEMA 2: KETTERÄT TEKNIIKAT

	1	2	3	4	5
Projektin rakenne oli yksinkertainen					
Dokumentaatiota tuotettiin sopiva määrä					

TEEMA 3: TIIMIN KYVYKKYYS/TAIDOT (CAPABILITY)

	1	2	3	4	5
Tiimin jäsenet olivat asiantuntevia					
Tiimin jäsenet olivat motivoituneita					
Tiimin johdolla oli selkeä käsitys ketteryydestä ennestään					
Tiimin johdolla oli sopeutuvaiset toimintatavat					
Tiimi koulutettiin projektin alussa tarkoituksenmukaisesti					

TEEMA 4: PROJEKTIJOHTO

	1	2	3	4	5
Vaatimusmäärittely oli ketterää (vaatimukset elivät projektin aikana)					
Projektihallinta oli ketterää					
Projektin etenemistä seurattiin selkein ja toimivin mekanismein					
Kommunikointi oli voimakasta (esim. tiimin päivittäiset tapaamiset)					
Projektin aikataulu oli selkeä (tasainen, esteetön eteneminen)					

TEEMA 5: TOIMINTAYMPÄRISTÖ

	1	2	3	4	5
Koko tiimi toimi samassa lokaatiossa					
Tiimin työskentely oli itseorganisoituvaa					
Tiimi oli kooltaan pieni (5=tiimi oli pieni, 1=tiimi oli suuri)					
Tiimi oli yksittäinen, eheä kokonaisuus					

TEEMA 6: ASIAKKAAN LÄSNÄOLO

	1	2	3	4	5
Asiakassuhde oli yleisluonteeltaan positiivinen					
Asiakas oli voimakkaasti läsnä ja vaikuttamassa päätöksentekoon					
Asiakkaalla oli täysi päätösvalta					

LIITE 2. Teemahaastattelukysymykset

TEEMA 1: YKSILÖT JA KANSSAKÄYMINEN

Liiketoiminnan edustajat ja kehittäjät yhteistyössä päivittäin koko projektin ajan (Periaate 4), toimivin keino kommunikoida on kasvokkain (Periaate 6)

Miten yhteistyö asiakastahon ja organisaationne edustajiston välillä toteutettiin?
Kuinka usein asiakasta tavattiin?
Mitä etuja mielestänne kasvokkain tapaamisista saatiin?
Kokiko asiakas olleensa ns. vallan kahvassa projektissa? Tai miten te näitte asian?

Projektit motivoituneiden ihmisten ympärille: heille tarvitut puitteet ja tuki ja luotto heihin (Periaate 5)

Kuinka koette projektihenkilöstönne motivaation tason projektin alussa? Piditkö henkilöstöä motivoituneena?
Millä keinoin takaat, että projekti henkilöstösi on korkeasti motivoitunutta voidaksesi täysin luottaa heihin?

Itseorganisoituvat tiimit (Periaate 11)

Miten tiimin työskentely on organisoitu projektin aikana? Kuka johtaa? Kuka välittää tiedon hierarkiassa ylöspäin? Kuka tapaa asiakasta?
Miten näet tilanteen, jossa tiimi toimisi täysin itseorganisoidusti? Sinun ja tiimin välillä toimisi esimerkiksi jonkinlainen coach, joka vastaisi siitä, että kehitysmenetelmä toimii ja, että sinä tiedät missä tilassa projekti on?

Tiimi tarkastelee ja tutkii, kuinka parantaa tehokkuuttaan ja mukauttaa toimintansa (Periaate 12)

Miten projektin aikana toimintatapoja pyrittiin kehittämään? Kenen aloitteesta parannustoimiin ryhdyttiin?

TEEMA 2: TOIMIVA OHJELMISTO (DOKUMENTAATION SIJAAN)

Toimivat ohjelmistoversiot enimmillään kuukauden välein (Periaate 3)

Kuinka usein ohjelmistosta toimitettiin versioita asiakkaalle? Miten sen olisi voinut tehdä paremmin ja miksi tai nähtiinkö toteutettu vaihtoehto parhaana mahdollisena?

Toimiva ohjelmisto on ensisijainen mittari (Periaate 7)

Mikä oli teille selkein projektin etenemisen mittari? Mitä tietoa se tarjosi siitä, että asiakas on tyytyväinen ja vaatimukset täyttyy?

Teknisen laadun ja hyvän rakenteen huomiointi edesauttaa (Periaate 9)

Miten laatua tai hyvää ohjelmistorakennetta huomioidaan? Jokin malli? Miten toimii?

TEEMA 3: ASIAKASYHTEISTYÖ

Asiakkaan tarpeiden huomioinen ja varhainen toimitus (Periaate 1)

Miten projekti alkoi? Miten lähditte selvittämään vaatimuksia? Kuinka pian projektin alkamisesta ensimmäinen release?
Kuinka usein toimitukset toteutettiin jatkossa? Miten palaute kerättiin?
Miksi mielestäsi lyhyt iteraatio on parempi tapa toimittaa tuotetta ja miten koet sen hyödyttävän projektin etenemistä?

Kestävä toimintatapa - Työtahti ja tavat samoina pitkään (Periaate 8)

Millä tavalla näette nykyisten toimintatapojen kestävyyden ja onko tämä mielestänne juuri oikea tapa toimia? Miksi / miksi ei?

TEEMA 4: MUUTOKSEEN VASTAAMINEN

Myöhäiset vaatimukset ja asiakkaan kilpailukyvyn edistäminen muutosta hyödyntäen (Periaate 2)

Miten vaatimusmuutoksia hallinnoitiin? Jos asiakas ilmoitti muuttuneista olosuhteista tai tilanteesta liiketoiminnassaan, joka johtaa vaatimusten elämiseen, miten siihen reagoitiin teillä? Kuinka useasti muuttuviin vaatimuksiin törmättiin? Näetkö tässä jotain keinoa kehittää?
Miten projektissa hyödynnettiin muutosta? Uutta teknologiaa etc? Big Data, pilvi, SoMe jne?
Näkikö asiakas saavuttaneensa kilpailullista hyötyä projektin päätteeksi? Millaista?

Yksinkertaisuus: tekemättä jätetyn työn maksimointi (Periaate 10)

Millä tavoin yksinkertaisuutta pyrittiin korostamaan ja tehtävää työtä minimoimaan?