



Jussi Niinistö

**KETTERÄN KEHITYKSEN JA  
KÄYTTÄJÄKESKEISEN SUUNNITTELUN  
INTEGRAATIO  
OHJELMISTOTUOTANNOSSA**  
Systemaattinen kirjallisuuskatsaus

Informaatioteknologian ja viestinnän tiedekunta  
Pro gradu -tutkielma  
Huhtikuu 2019

# TIIVISTELMÄ

Jussi Niinistö: Ketterän kehityksen ja käyttäjäkeskeisen suunnittelun integraatio  
ohjelmistotuotannossa  
Pro gradu -tutkielma  
Tampereen yliopisto  
Tietojenkäsittelytieteiden tutkinto-ohjelma  
Huhtikuu 2019

---

Ketterä kehitys on ohjelmistotuotannon lähestymistapa, jossa ohjelmiston vaatimuksia ja implementaatioita kehitetään tiiviillä yhteistyöllä projektitiimin, asiakkaan sekä loppukäyttäjien kanssa. Sen vahvuuksina nähdään muun muassa nopeat ja säännölliset toimitusajat sekä kyky reagoida yllättäviin muutoksiin ilman, että niillä olisi projektin kulkuun merkittäviä negatiivisia vaikutuksia. Tämä varmistetaan esimerkiksi sillä, että projektien vaatima etukäteissuunnittelu ja dokumentaatio pidetään mahdollisimman pienenä.

Koska ketterä kehitys on hyvin ohjelmistokeskeistä ja sen periaatteena on toimivien iteraatioiden jatkuva toimittaminen, saattaa suunnittelutyölle ja käyttökokemukselle varattu aika jäädä tällaisissa projekteissa liian pienelle huomiolle. Perusteellisen käyttäjäkeskeisen suunnittelun puuttuessa kasvaa myös riski siitä, ettei kehitetty ohjelmisto vastaakaan sen loppukäyttäjien tarpeita.

Tässä tutkielmassa tarkastellaan systemaattisen kirjallisuuskatsauksen keinoin sitä, miten ketterä kehitys ja käyttäjäkeskeinen suunnittelu saadaan toimimaan yhteistyössä ilman, että niiden periaatteille oleelliset prosessit ovat ristiriidassa keskenään. Tutkielmassa käydään läpi tällaisen integraation erilaisia haasteita ja niihin suunnattuja kehitysmahdollisuuksia. Tutkielman tuloksena saatiin tietoa siitä, millaiset haasteet ovat integraatiossa yleisimpiä ja millaiset ratkaisut nähdään parhaimpina kyseisten haasteiden minimoimiselle.

Avainsanat: ketterä ohjelmistokehitys, Agile, käyttäjäkeskeinen suunnittelu, ohjelmistotuotanto, Scrum, XP, Kanban, UX.

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck –ohjelmalla.

## Sisällys

<b>1. Johdanto.....</b>	<b>1</b>
<b>2. Tutkimusmenetelmät .....</b>	<b>3</b>
2.1 Tavoitteet ja tutkimuskysymykset.....	3
2.2 Systemaattinen kirjallisuuskatsaus .....	4
2.3 Tiedonhaku ja aineiston rajausta .....	5
2.4 Analysoitava aineisto .....	10
<b>3. Ketterät menetelmät .....</b>	<b>12</b>
3.1 Ketterän kehityksen periaatteet .....	12
3.2 Scrum .....	16
3.3 Extreme Programming (XP).....	18
3.4 Kanban .....	20
<b>4. Käyttäjäkeskeinen suunnittelu ohjelmistotuotannossa .....</b>	<b>22</b>
4.1 Käyttäjäkokeemus (UX) .....	22
4.2 Käytettävyys.....	25
<b>5. Ketterä käyttäjäkeskeinen suunnittelu .....</b>	<b>28</b>
5.1 Haasteet .....	29
5.1.1 Ajankäytön haasteet .....	30
5.1.2 Roolien ja tehtävien epäselvyydet.....	31
5.1.3 Kommunikointien haasteet .....	33
5.2 Hyödyt ja kehitysmahdollisuudet.....	35
5.2.1 Etukäteissuunnittelu .....	36
5.2.2 Viestinnän kehittäminen.....	39
<b>6. Keskustelu .....</b>	<b>42</b>
<b>7. Yhteenveto .....</b>	<b>44</b>
<b>Lähteet.....</b>	<b>45</b>

## 1. Johdanto

Ketterä kehitys on nykyisen ohjelmistotuotannon suurimpia trendejä ja se on haastanut suosiollaan perinteisen vesiputousmallin. Sitä voidaankin pitää jo eräänlaisena ohjelmistokehityksen standardina. [Jeremiah, 2017].

Parhaimmillaan ketterä kehitys johtaa laadukkaisiin tuotteisiin, jotka vastaavat asiakkaiden ja käyttäjien tarpeisiin, ovat kustannustehokkaita sekä valmistuvat nopeasti. Ketterä kehitys ei pidä kattavaa dokumentaatiota tai alkupään vaatimusmäärittelyä tärkeinä, vaan pyrkii keräämään kaiken oleellisen tiedon projektin edetessä. Nopeaan ja iteratiiviseen ominaisuuksien toimittamiseen liittyy kuitenkin vaara siitä, että ohjelmiston suunnittelu, käytettävyys ja käyttökokemus asetetaan pienemmälle prioriteetille kehitystyössä. Mikäli kehitystyön fokus on liian tiukasti aikatauluissa ja teknisessä toteutuksessa, saattaa lopputuloksena syntyä ohjelmisto, joka ei vastaa asiakkaan tarpeita käyttäjäkeskeisestä näkökulmasta. [Brhel et al., 2015]

Käyttäjakeskeisen suunnittelun näkökulmasta onkin erittäin oleellista, että hyvän käytettävyyden ja käyttökokemuksen saavuttamiseksi on suoritettava muun muassa käyttäjien etukäteistutkimusta, vaatimusmäärittelyä sekä käytettävyydetutkimuksia. Tämän avulla pystytään varmistumaan siitä, että kehitettävä ohjelmisto vastaa asiakkaiden ja käyttäjien tarpeisiin ominaisuuksiltaan, käytettävyydeltään sekä käyttökokemukseltaan. [Brehl et al., 2015]

Koska nopea toimitus, tekninen laadukkuus, ketterä muutoksiin vastaaminen sekä käyttäjakeskeisyys ovat kaikki tärkeitä arvoja ohjelmistokehityksessä, on loogista tavoitella lopputulosta, jossa näiden kahden työskentelymetodin parhaat toimintatavat yhdistyvät [Brehl et al., 2015]. Vaikka molemmissa metodeissa yhdistyvät ainakin jossain määrin iteratiivisuus, käyttäjakeskeisyys ja säännölliset kommunikaatiot, on niiden sulavassa ja tehokkaassa yhdistämisessä paljon haasteita [Nadikattu, 2016].

Tässä tutkielmassa pyrittiin löytämään vastauksia siihen, millä tavoin käyttäjakeskeinen suunnittelu saadaan integroitua osaksi ketterää ohjelmistotuotantoa ilman, että näille työskentelytavoille oleelliset prosessit ovat konfliktissa keskenään. Tutkielmassa tehdyt havainnot perustuvat olemassa oleviin tutkimuksiin, joita analysoitiin Kitchenhamin [2004] systemaattisen kirjallisuuskatsauksen keinoin.

Aluksi käydään läpi tutkimuksessa käytetyt tutkimusmenetelmät luvussa 2. Siinä kuvataan tutkimuksen tavoitteet ja tutkimuskysymykset, systemaattisen kirjallisuuskatsauksen menetelmät sekä lopuksi esitellään tiedonhaun prosessit, aineiston rajaukseen vaikuttavat kriteerit sekä tutkimukseen valittu lopullinen tieteellinen aineisto.

Luvuissa 3 ja 4 käydään läpi tutkimukselle oleellista teoretietoa ketteristä menetelmistä ja käyttäjäkeskeisestä suunnittelusta. Näiden aiheiden ymmärtämistä pidetään oleellisena lähtökohtana varsinaisen integraation ja siihen liittyvien hyötyjen ja haasteiden ymmärtämisen kannalta.

Luvussa 5 tarkastellaan hyväksytyn tutkimusaineiston pohjalta ketterien menetelmien ja käyttäjäkeskeisen suunnittelun integraatiota valittujen tutkimusten tarjoamista eri näkökulmista. Luvussa esitellään muun muassa toistuvia haasteita ja parhaita menetelmiä toimivan integraation saavuttamiseksi.

Tämän jälkeen luvussa 6 käydään keskustelua tutkimustuloksiin pohjautuvista integraation menetelmistä sekä niiden vahvuuksista ja heikkouksista. Luvussa esitetään tämän lisäksi myös ehdotuksia mahdollisille jatkotutkimuksille. Viimeisessä luvussa esitetään tutkielman tuloksiin pohjautuva yhteenveto.

## 2. Tutkimusmenetelmät

### 2.1 Tavoitteet ja tutkimuskysymykset

Tässä tutkielmassa pyrittiin löytämään vastaus siihen, mitkä ovat parhaat menetelmät yhdistää ketterä ohjelmistotuotanto käyttäjäkeskeisen suunnittelun kanssa. Vastauksen avulla haluttiin saada kokonaisvaltainen käsitys siitä, millaiset työskentelytavat tukevat ja puolestaan haittaavat ketterän kehityksen ja käyttäjäkeskeisen suunnittelun yhdistämistä ohjelmistoprojekteissa. Tutkielman oleellisena osana olikin näin ollen myös selvittää, millaisista integraation keinoista eri tutkijat ovat samaa mieltä. Tämän lisäksi vastauksia haettiin seuraaviin lisäkysymyksiin, jotka olivat pääkysymykseen vastaamisen kannalta oleellisia:

1. Mitä on ketterä ohjelmistotuotanto?
2. Mitä on käyttäjäkeskeinen suunnittelu?
3. Mitä näiden yhdistämisestä tiedetään nykyisten tutkimusten perusteella?
4. Millaista lisätutkimusta aiheesta tulisi vielä tehdä?

Vastaus pääkysymykseen pyrittiin löytämään systemaattisen kirjallisuuskatsauksen avulla, missä tarkoituksena oli analysoida aiheita käsitteleviä tieteellisiä tutkimuksia. Aiheesta löytyi tutkielman ajankohtana jo useampia tutkimuksia, joten tutkimuskysymystä varten pyrittiin kokoamaan näiden tutkimusten tuloksista koostuva synteesi. Synteesin kokoamisessa haluttiin kiinnittää erityisesti huomiota tutkimustuloksissa toistuviin havaintoihin, kuten onnistumisiin ja epäonnistumisiin johtaviin tekijöihin ja työskentelymenetelmiin.

## 2.2 Systemaattinen kirjallisuuskatsaus

Tutkimus päädyttiin toteuttamaan Kitchenhamin [2004] esittämän systemaattisen kirjallisuuskatsauksen avulla. Menetelmä valittiin, koska tutkielman aiheesta oli tiettävästi tehty jo aikaisempaa tutkimusta eri vuosilta usean eri tutkijan toimesta. Systemaattisen kirjallisuuskatsauksen avulla pystytään luomaan saatavilla olevista tutkimuksista ja niiden tuloksista luotettava synteesi, jonka avulla voidaan esittää aiheeseen liittyviä parhaita menetelmiä, havaita ongelmakohtia, tunnistaa olemassa olevan tutkimusaineiston puutteita sekä löytää uusia potentiaalisia tutkimuskohteita [Kitchenham, 2004].

Kitchenham [2004] listaa seuraavat aktiviteetit osaksi systemaattista kirjallisuuskatsausta:

1. Kirjallisuuskatsauksen tarpeellisuuden arviointi
2. Tutkimusprotokollan kehittäminen
3. Tutkimusten tunnistaminen
4. Ensisijaisten tutkimusten valitseminen
5. Tutkimusten laadun evaluointi
6. Datat keräys ja monitorointi
7. Synteesin tekeminen

Kirjallisuuskatsauksen tarpeellisuuden arviointi oli prosessi, jota työstiin tutkielman aihetta valitessa sekä tutkimustietoa etsittäessä ja analysoitaessa. Kirjallisuuskatsausta lähdettiin tekemään sillä oletuksella, että kyseessä on ajankohtainen ja trendikäs aihe, joka tarvitsee edelleen lisätutkimusta. Ajatusta olemassa olevan tutkimustiedon yhteen kokoamisesta pidettiin myös hyödyllisenä, koska se voisi tuoda aiheeseen uusia näkökulmia ja paljastaa yleisiä ongelmia, joihin voitaisiin puolestaan kiinnittää enemmän huomiota tulevissa tutkimuksissa.

Kirjallisuuskatsauksen luotettavuuden kannalta on tärkeää, että sitä varten luodaan etukäteen tutkimusprotokolla, ja että löydettyä aineistoa kohdellaan objektiivisesti ja reilusti. Tämä tarkoittaa erityisesti sitä, että tutkijan omien hypoteesien ei tule vaikuttaa löytyneiden aineistojen valitsemiseen tai hylkäämiseen. Valinta- ja hylkäyskriteerit saavat perustua ainoastaan siihen, ovatko aineistot luotettavia ja ovatko ne relevantteja tutkimukselle. [Kitchenham, 2004]

Tarpeeksi laajan systemaattisen kirjallisuuskatsauksen hyötynä on, että sen avulla pystytään löytämään todisteita eri tilanteissa havaittujen toistuvien ilmiöiden luotettavuudesta. Mikäli löydökset olisivat epäjohdonmukaisia, voidaan tällaiseen tilanteeseen johtaneita syitä puolestaan lähteä analysoimaan tarkemmin.

### 2.3 Tiedonhaku ja aineiston rajaaminen

Kuten Kitchenhamin [2004] listaamista aktiviteeteista voidaan todeta, vaatii luotettavasti toteutettu systemaattinen kirjallisuuskatsaus tutkimusprotokollan kehittämisen [Kitchenham, 2004]. Protokollan ensisijaiseksi strategiaksi valittiin avainsanoihin pohjautuva haku muutamasta saatavilla olevasta tietokannasta ja hakukonepalvelusta. Näihin sisältyivät ACM Digital Library, ScienceDirect, Springer Link sekä Google ja Google Scholar. Avainsanoilla suoritettu hakutulosten suodattaminen oli osa Kitchenhamin [2004] mainitsemaa tutkimusten tunnistamista.

Koska tutkielman aihe koskee ketteriä menetelmiä ja käyttäjäkeskeistä suunnittelua, valittiin avainsanoiksi aluksi termien englanninkieliset vastineet ”agile” sekä ”user-centered design”. Tämän lisäksi tuloksissa huomioitiin ainoastaan suomen ja englannin kieliset lehtiartikkelit sekä konferenssipaperit.

Ensimmäisenä hakuprioriteettina avainsanat yhdistettiin operaattorin AND kanssa, jotta tuloksista saatiin karsittua pois sellaisia epärelevanttejä tutkimuksia, jotka sisälsivät vain toisen avainsanan. Hakusana ”agile” pidettiin jokaisessa haussa mukana, mutta ”user-centered design” -avainsanan tilalla käytettiin myös variaatioita ”user-centered” sekä ”user-centred”. Tämän lisäksi kyseisen hakusanan tilalla käytettiin termejä ”user experience” sekä ”design thinking”. Design Thinking valittiin, koska kyseisellä metodologialla todettiin olevan paljon yhtäläisyyksiä käyttäjäkeskeisen suunnittelun kanssa. Kyseisellä hakusanalla ei kuitenkaan löydetty hyödyllisiä uniikkeja tuloksia.

Koska hakutuloksia löytyisi oletettavasti runsaasti, oli tärkeää määrittää myös valintakriteerit, joiden avulla valittavien tutkimusten määrää pystyttäisiin rajaamaan ja näin ollen tunnistamaan tutkielmalle oleelliset tutkimukset. Kitchenham [2004] listaakin seuraavaksi aktiviteetiksi ensisijaisten tutkimusten valitsemisen, joka pitää sisällään myös valintakriteerien määrittelemisen, minkä avulla



pystytään arvioimaan löydettyjen tutkimusten laatua. Päädyttiin luomaan seuraavanlaiset esikarsintakriteerit näiden tutkimusten valitsemiselle:

**Kriteeri #1:** Vaikuttaako tutkimuksen otsikko oleelliselta tutkielman kannalta?

**Kriteeri #2:** Vaikuttaako tutkimuksen abstrakti oleelliselta tutkielman kannalta?

Taulukossa 1 ollaan eritelty tietokantakohtaisesti sellaiset haut, joiden avulla löydettiin esikarsintakriteerit täyttäviä uniikkeja tutkimuksia.

Lyh.	Tietokanta	Haku	Hakutuloksia yhteensä	Esikarsinnan läpäisseet hakutulokset
A1	ACM Digital Library	agile AND user-centered design	661	1
A2	ACM Digital Library	agile AND user experience	456	5
SD1	ScienceDirect	agile AND user-centered design	6370	5
SL1	Springer Link	agile AND user-centered	979	2
GS1	Google Scholar	agile AND user experience	158 000	1

Taulukko 1. Tietokannoissa suoritettut haut, jotka tuottivat esikarsintakriteerit täyttäviä hakutuloksia.

Osaan löytyneistä tutkimuksista ei saatu suoraa pääsyä tietokannan kautta, joten niihin etsittiin suora linkki Googlen avulla. Tällaisten tutkimusten lähteeksi lasketaan tästä huolimatta tietokanta, jonka kautta ne alun perin löydettiin. Löydettyistä hakutuloksista käytiin läpi 40-50 ensimmäistä tulosta, minkä jälkeen haku lopetettiin.

#	Haku	Tekijä, vuosi	Tutkimuksen nimi
1	A1	Sy, 2007	Adapting Usability Investigations for Agile User-centered Design
2	A2	Nedeltcheva and Shoikova, 2017	Coupling Design Thinking, User Experience Design and Agile: Towards Cooperation Framework
3	A2	Bruun, Larusdottir, Nielsen, Nielsen and Persson, 2018	The Role of UX Professionals in Agile Development: A Case Study from Industry
4	A2	Budwig, Jeong and Kelkar, 2009	When user experience met agile: a case study
5	A2	Salah, Paige and Cairns, 2015	Patterns for Integrating Agile Development Processes and User Centred Design
6	A2	Kuusinen and Väänänen-Vainio-Mattila, 2012	How to make agile UX work more efficient: management and sales perspective
7	SD1	Brhel, Meth, Maedche and Werder 2015	Exploring principles of user-centered agile software development: A literature review
8	SD1	Gulliksen, Larusdottir and Cajander, 2016	A license to kill – Improving UCSD in Agile development
9	SD1	Da Silva, Silveira, Maurer and Silveira, 2018	The Evolution of agile UXD
10	SD1	Losada, Urretavizcaya and Fernández-Castro, 2012	A guide to agile development of interactive software with a “User Objectives” -driven methodology
11	SD1	Pereira and Russo, 2018	Design Thinking Integrated in Agile Software Development: A systematic Literature Review
12	SL1	Düchting, Zimmermann and Nebe, 2007	Incorporating User Centered Requirement Engineering into Agile Software Development
13	SL1	Chamberlain, Sharp and Maiden, 2006	Towards a Framework for Integrating Agile Development and User-Centered Design
14	GS1	Da Silva, Silveira, Maurer and Hellmann, 2012	User experience design and agile development: From theory to practice

Taulukko 2. Valitut tutkimukset hakukohtaisesti.

Esikarsinnasta valikoitui lopulta taulukossa 2 luetellut 14 tutkimusta. Koska kyseisistä tutkimuksista tiedettiin esikarsinnan jälkeen vain otsikon ja abstraktin sisältö, tuli tutkimuksia arvioida vielä niiden muun sisällön perusteella. Kitchenham [2004] puhuu tästä aktiviteetista tutkimusten laadun evaluointia. Käytännössä tämä tarkoitti jokaisen

tutkimuksen lukemista alusta loppuun, minkä jälkeen niiden oleellisuutta tutkimukselle arvioitiin uudelleen jälkikarsinnan avulla. Jälkikarsintakriteerit olivat seuraavat:

**Kriteeri #1:** Sisältääkö tutkimus tarpeeksi oleellista tietoa tutkimuskysymysten näkökulmasta?

**Kriteeri #2:** Onko tutkimuksen tuloksista hyötyä tutkimuskysymysten näkökulmasta?

Kirjallisuuskatsaukseen valittavan tutkimuksen haluttiin käsittelevän aihetta sellaisesta näkökulmasta, jossa yhdistyvät vahvasti sekä ketterä kehitys että käyttäjäkeskeinen suunnittelu. Ensimmäisen kriteerin avulla pyrittiin karsimaan pois sellaiset tutkimukset, joiden keskeinen fokus on liian kaukana tästä näkökulmasta, tai joiden sisältö koettiin liian suppeaksi.

Toisen kriteerin kohdalla huomiota kiinnitettiin tutkimuksissa esitettyihin tuloksiin, päätelmiin sekä yhteenvetoon. Tutkimuksista pyrittiin näin ollen karsimaan pois sellaiset tutkimukset, joiden tulokset eivät tarjonneet oleellista tietoa synteetin kokoamisen kannalta. Esimerkiksi tutkimukset 2 [Nedeltcheva ja Shoikova, 2017] ja 11 [Pereira ja Russo, 2018] keskittyivät liikaa Design Thinkingin teoriaan, eivätkä ne näin ollen tarjonneet tutkimuskysymyksille oleellista tietoa. Tutkimus 10 [Losada, Urretavizcaya ja Fernández-Castro, 2012] puolestaan oli sisällöltään liian tekninen ja teoreettinen, minkä vuoksi siitä puuttui käyttäjäkeskeiselle suunnittelulle oleellinen ihmislähtöinen näkökulma kokonaan. Jälkikarsinnan läpäisi lopulta 8 tutkimusta, jotka on listattu taulukossa 3.

#	Haku	Tekijä, vuosi	Tutkimuksen nimi
1	A1	Sy, 2007	Adapting Usability Investigations for Agile User-centered Design
4	A2	Budwig, Jeong and Kelkar, 2009	When user experience met agile: a case study
6	A2	Kuusinen and Väänänen-Vainio-Mattila, 2012	How to make agile UX work more efficient: management and sales perspective
7	SD1	Brhel, Meth, Maedche and Werder 2015	Exploring principles of user-centered agile software development: A literature review
8	SD1	Gulliksen, Larusdottir and Cajander, 2016	A license to kill – Improving UCSD in Agile development
9	SD1	Da Silva, Silveira, Maurer and Silveira, 2018	The Evolution of agile UXD
13	SL1	Chamberlain, Sharp and Maiden, 2006	Towards a Framework for Integrating Agile Development and User-Centered Design
14	GS1	Da Silva, Silveira, Maurer and Hellmann, 2012	User experience design and agile development: From theory to practice

Taulukko 3. Jälkikarsinnan läpäisseet tutkimukset.

Toissijaiseksi hakustrategiaksi valittiin ensisijaisella haulla valikoituneiden tutkimusten lähdeaineiston tarkastelu. Tämä hakustrategia noudatti samoja kriteerejä kuin ensisijainen haku, tosin tutkimusten valinnassa kiinnitettiin huomiota myös kontekstiin, jossa niihin oltiin viitattu. Toissijaisella haulla löydettiin taulukossa 4 kuvatut kolme tutkimusta, jotka läpäisivät esi- ja jälkikarsinnan hyväksytysti.

#	Tekijä, vuosi	Tutkimuksen nimi	Lähde
1	Kuusinen, Mikkonen and Pakarinen, 2012	Agile User Experience Development in a Large Software Organization: Good Expertise but Limited Impact	[Brhel et al., 2015]
2	McInerney and Maurer, 2005	UCD in Agile Projects: Dream Team or Odd Couple?	[Da Silva et al., 2012]
3	Adikari, Campbell and McDonald, 2009	Little Design Up-Front: A Design Science Approach to Integrating Usability into Agile Requirements Engineering	[Da Silva et al., 2012]

Taulukko 4. Toissijaisella hakustrategialla valikoituneet aineistot.

## 2.4 Analysoitava aineisto

Kaiken kaikkiaan 17 eri tutkimuksesta valikoitui lopulta hyväksytyin kriteerein taulukossa 5 kuvatut 11 tutkimusta. Näihin tutkimuksiin sisältyi 5 konferenssipaperia ja 6 lehtiartikkelia, joista 1 [Brhel et al., 2015] oli tyypiltään kirjallisuuskatsaus. Tutkimukset on järjestetty taulukossa niiden julkaisuvuoden mukaan.

Kitchenham [2004] listaa toiseksi viimeiseksi kirjallisuuskatsauksen aktiviteetiksi datan keräämisen ja monitoroinnin. Tällä tarkoitetaan nimensä mukaisesti informaation keräämistä valituista tutkimuksista. Tätä tietoa puolestaan hyödynnetään myöhemmin kirjallisuuskatsauksen synteesin tekemisessä. Tässä tutkielmassa informaatiota kerättiin talteen erilaisilla taulukoiduilla muistiinpanoilla, jotta tutkimuksista saatu tieto oli helpommin referoitavissa synteesiä tehdessä. Tutkimuksia ja niistä kerättyä tietoa analysoidaan tutkimuskysymysten kannalta tarkemmin luvussa 5.

Lyh.	Tekijä, vuosi	Tutkimuksen nimi	Julkaisija	Tyyppi
T1	McInerney and Maurer, 2005	UCD in Agile Projects: Dream Team or Odd Couple?	Interactions	Lehtiartikkeli
T2	Chamberlain, Sharp and Maiden, 2006	Towards a framework for integrating agile development and user-centered design	Extreme Programming and Agile Processes in Software Engineering	Konferenssi-paperi
T3	Sy, 2007	Adapting Usability Investigations for Agile User-centered Design	Journal of Usability Studies	Lehtiartikkeli
T4	Adikari, Campbell and McDonald, 2009	Little Design Up-Front: A Design Science Approach to Integrating Usability into Agile Requirements Engineering	Human-Computer Interaction – HCII 2009	Konferenssi-paperi
T5	Budwig, Jeong and Kelkar, 2009	When User Experience Met Agile: A Case Study	CHI EA 2009	Konferenssi-paperi
T6	Da Silva, Silveira, Maurer and Hellmann, 2012	User Experience Design and Agile Development: From Theory to Practice	Journal of Software Engineering and Applications	Lehtiartikkeli
T7	Kuusinen, Mikkonen and Pakarinen, 2012	Agile User Experience Development in a Large Software Organization: Good Expertise but Limited Impact	International Conference on Human-Centered Software Engineering	Konferenssi-paperi
T8	Kuusinen and Väänänen-Vainio-Mattila, 2012	How to make agile UX work more efficient: management and sales perspective	NordiCHI '12 Proceedings of the 7 <sup>th</sup> Nordic Conference on Human-Computer Interaction	Konferenssi-paperi
T9	Brhel, Meth, Maedche and Werder 2015	Exploring principles of user-centered agile software development: A literature review	Information and Software Technology	Lehtiartikkeli
T10	Gulliksen, Larusdottir, Cajander et al., 2016	A license to kill – improving UCSD in Agile Development	Journal of Systems and Software	Lehtiartikkeli
T11	Da Silva, Silveira, Maurer and Silveira, 2018	The evolution of agile UXD	Information and Software Technology	Lehtiartikkeli

Taulukko 5. Kirjallisuuskatsaukseen valittu lopullinen aineisto.

### 3. Ketterät menetelmät

Tässä luvussa esitellään ketterän ohjelmistokehityksen periaatteita sekä kolme erilaista näihin periaatteisiin pohjautuvaa menetelmää. Nämä menetelmät ovat Scrum, Extreme Programming sekä Kanban. Kyseiset esiteltävät menetelmät on valittu, koska niihin viitataan useasti kirjallisuuskatsaukseen valituissa tutkimuksissa ja ne ovat myös yleisimpien tunnettujen ketterien menetelmien joukossa. Ketterän kehityksen periaatteiden ymmärtäminen on tutkielman kannalta oleellista, jotta voidaan paremmin ymmärtää ketterien menetelmien ja käyttäjäkeskeisen suunnittelun integraatioon vaikuttavia tekijöitä eri näkökulmista.

#### 3.1 Ketterän kehityksen periaatteet

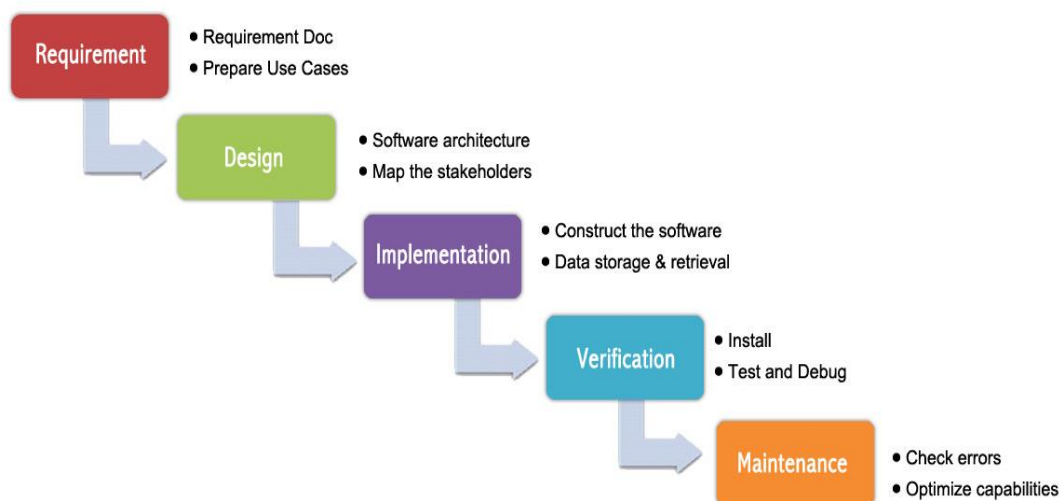
Ketterästä kehityksestä (engl. *Agile development*) löytyy kirjallisuutta jo 1980-luvulta [Takeuchi ja Nonaka, 1986], mutta sen varsinaisen syntymäajankohdan voidaan ajatella olevan vuosi 2001. Silloin 17 kyseisen kehitysajatuksen puolestapuhujaa kokoontuivat yhteen Yhdysvalloissa ja loivat niin sanotun ketterän manifestin (engl. *Agile Manifesto*), joka pohjautuu ketterän kehityksen 12 peruseriaatteeseen. Nämä periaatteet ovat seuraavat [Beck et al., 2001]:

1. Asiakkaiden tyytyväisyyden takaus ohjelmistoversioiden lyhyellä ja säännöllisellä toimitusajalla
2. Muuttuvien vaatimusten vastaanottaminen kehityksen eri vaiheissa
3. Toimivan ohjelmistoversion toimittaminen säännöllisesti lyhyellä aikavälillä
4. Liiketoiminnan edustajien ja ohjelmistokehittäjien päivittäinen yhteistyö
5. Yksilöiden motivoiminen
6. Kasvotusten käytävän kommunikaation mahdollistaminen
7. Toimiva ohjelmisto edistymisen ensisijaisena mittarina
8. Ketterät menetelmät kannustavat kestävään kehitykseen
9. Ohjelmiston laadun ja rakenteen jatkuva huomiointi
10. Yksinkertaisuus
11. Itseorganisoituvat tiimit
12. Säännöllinen kehityskeskustelu

Näiden periaatteiden lisäksi manifestissa luetellaan neljä arvoa, joita ketterässä kehityksessä arvostetaan erityisesti [Beck et al. 2001]:

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

Ketterä kehitysmalli sai alkunsa ohjelmistokehittäjien tarpeesta luoda vaihtoehtoinen työskentelytapa perinteisemmälle vesiputousmallille, jossa tuotanto nojaa vahvasti projektin alkuvaiheessa luotuihin vaatimusmäärittäisiin ja dokumentaatioon [Beck et al. 2001]. Vesiputousmallia on havainnollistettu kuvassa 1.



Kuva 1. Vesiputousmallissa pyritään siihen, että vaatimusmäärittely, suunnittelu ja toteutus tehdään alusta asti oikein. [Subbaiah, 2016]

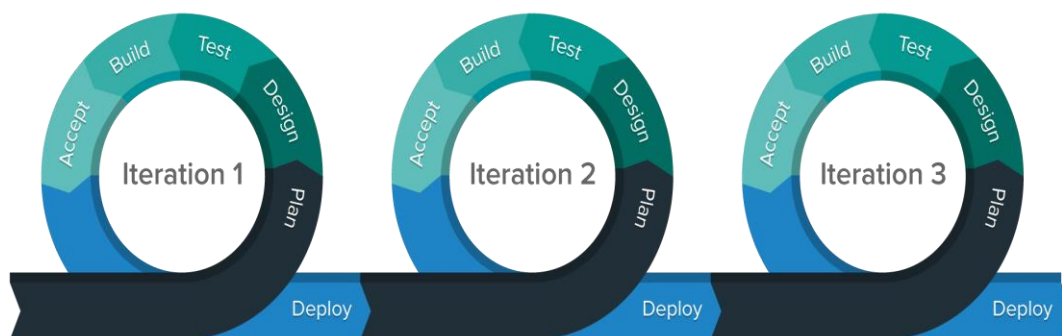
Vesiputousmallilla johdettavissa projekteissa on tyypillistä, että projekti seuraa alussa tehtyjä päätöksiä, vaatimusmäärittelyä sekä dokumentaatiota mahdollisimman tarkasti projektin loppuun asti [Lotz, 2018]. Hyvänä puolena tässä voidaan nähdä esimerkiksi se, että projektitiimillä on usein selkeä kokonaiskuva järjestelmästä jo projektin varhaisessa vaiheessa [Lotz, 2018]. Varhaiseen suunnitteluun nojaava toimintatapa saattaa aiheuttaa



ongelmia projektille sellaisissa tilanteissa, joissa vaatimusmäärittelyä joudutaan muuttamaan kesken projektin [Balaji ja Murugaiyan, 2012]. Mikäli projekti on edennyt esimerkiksi ohjelmiston testausvaiheeseen asti, tulee testaus pysäyttää siksi aikaa, että uudet vaatimukset saadaan määriteltyä, sovitettua arkkitehtuuriin sekä implementoitua. Tällainen toimintamalli voi tulla yritykselle kalliiksi ja sen lisäksi hidastaa projektin valmistumista [Balaji ja Murugaiyan, 2012].

Ketterä kehitys pyrkiikin löytämään ratkaisun vesiputousmallissa ilmeneviin ongelmiin. Sen keskiössä on ajatus siitä, että muutoksia pitää pystyä ennakoimaan ja niihin pitää pystyä reagoimaan mahdollisimman vaivattomasti ja kustannustehokkaasti. Tämä saavutetaan muun muassa iteratiivisella ja inkrementaalaisella toimintamallilla, jossa ohjelmistoon lisätään vähitellen uusia ominaisuuksia prioriteettien ja sidosryhmien toiveiden sekä tarpeiden mukaan. Osien toimitus tapahtuu yleensä aikataulutetuissa sykleissä, joiden lopussa ohjelmistosta tulee olla toimitettuna uusi toimiva versio. [Agile, 2015]

Ketterä kehitys keskittyy näin ollen vesiputousmallia pienempiin kokonaisuuksiin kerrallaan. Kuten kuvista 1 ja 2 voidaan kuitenkin havaita, sisältyy ketterän kehityksen iteraatioihin myös samoja osia kuin vesiputousmallissa.



Kuva 2. Ketterän kehityksen iteratiivinen malli. [Agile Project Management, 2019]

Asiakasperspektiivistä ketterän kehityksen vahvuuksina pidetään muun muassa sitä, että asiakkaat eivät monesti itsekään tiedosta vielä projektin alkuvaiheessa kaikkia vaatimuksia ja ominaisuuksia, jotka ohjelmiston tulee täyttää. Ketterä kehitys tarjoaa

asiakkaille mahdollisuuden seurata projektin etenemistä vesiputousmallia tiiviimmin ja tehdä yhteistyötä projektitiimin kanssa koko kehitysprosessin ajan. [Lotz, 2018]

Vesiputousmalliin verrattuna ketterä kehitys pystyy vastaamaan helpommin uusiin sekä muuttuviin vaatimuksiin, koska projektia pystytään hallitsemaan pienempinä kokonaisuuksina. Tästä syystä myös reagointi kehityksessä ilmeneviin ongelmatilanteisiin on ketterässä kehityksessä usein vaivattomampaa. [Agile, 2015]

Cockburnin ja Highsmithin [2001] mukaan kehitystiimin tehokkuus kasvaa, kun sen jäsenet työskentelevät lähellä toisiaan ja byrokraattinen dokumentointi korvataan kasvotusten käydylä kommunikaatiolla. Ongelmia saattaakin esiintyä, mikäli yhteinen fyysinen läsnäolo ei ole projektitiimin jäsenille mahdollista [Lotz, 2018]. Tällaista tilannetta voidaan helpottaa esimerkiksi videopuheluiden avulla [Lotz, 2018].

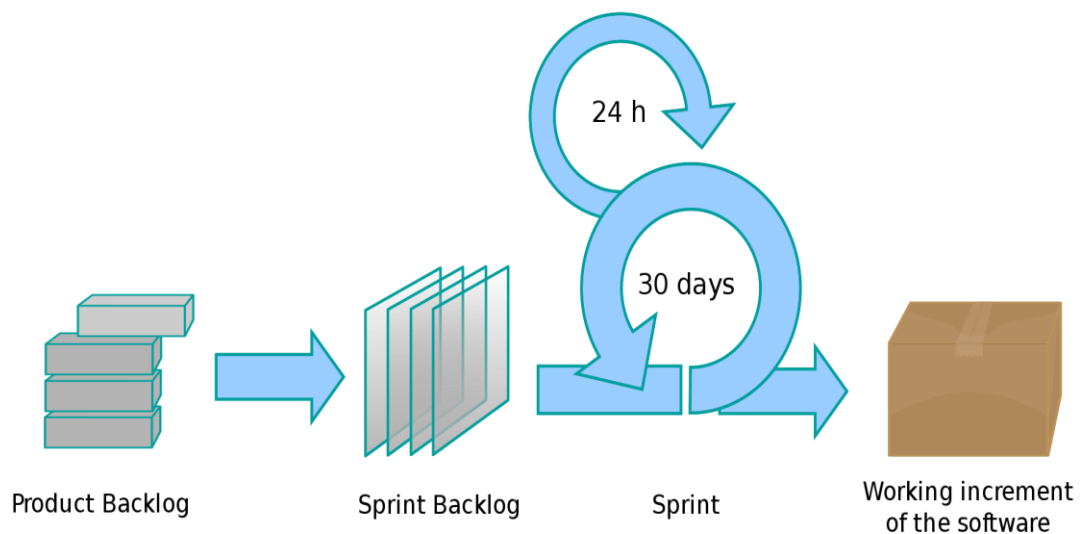
Ketterä kehitys pyrkii tuomaan esille kehitystiimin jäsenten yksilöllisiä vahvuuksia ja osaamisalueita, vaikka kyseessä onkin vahvasti tiimityöskentelyyn perustuva työskentelytapa. Jotta kehitystiimin ja sen yksilöiden osaamisesta saadaan suurin mahdollinen hyöty irti, on myös johtotason ymmärrettävä ketterän kehityksen merkitys, toimintatavat ja johtaa projektia sen mukaisesti. [Cockburn ja Highsmith, 2001]

Ketterän ohjelmistotuotannon integroiminen yritykseen ei ole yksinkertainen tai ongelmaton prosessi. Se vaatii ymmärrystä ja sitoutumista koko organisaation tasolla ja sen prosessit täytyy juurruttaa osaksi koko työyhteisön kulttuuria. Tästä syystä ketteryyden istuttaminen yritykseen voi olla raskasta ja haastavaa, eikä sitä ole välttämättä järkevää käyttää pienissä ja selkeissä projekteissa.

Seuraavaksi esitellään muutama yleisimmistä ketterään kehitykseen perustuvista viitekehyksistä, joiden avulla pyritään antamaan konkreettisempi käsitys siitä, millä eri tavoin ketterää ajatusmallia voidaan hyödyntää ohjelmistokehityksessä.

### 3.2 Scrum

Scrum on varsinkin ohjelmistotuotannossa käytetty viitekehys, joka pyrkii helpottamaan riskien ja muutosten hallintaa monimutkaisissa projekteissa [Schwaber ja Sutherland, 2016]. Takeuchi ja Nonaka [1986] esittelivät Scrumin terminä jo vuonna 1986, mutta viitekehyksen varsinaisina kehittäjinä pidetään Schwaberia ja Sutherlandia [2016], jotka olivat myös mukana kirjoittamassa ketterää manifestia.



Kuva 3. Scrum-mallin mukainen työnkulku. [Lakeworks, 2009]

Scrum-työskentely on luonteeltaan iteratiivista ja inkrementaalista, jossa projektia edistetään lyhyissä jaksoissa, eli *sprinteissä*. Kuvassa 3 ollaan havainnollistettu Scrumin mukaista prosessimallia. Yksi sprintti kestää tavallisesti maksimissaan yhden kuukauden, minkä aikana tuotteesta on määrä saada valmiiksi julkaisukelpoinen versio. Näin ollen tarkoituksena on lisätä tuotteeseen toimivia ominaisuuksia jaksoittain, kunnes tuotteen katsotaan olevan valmis. Jokaiselle sprintille tulee sen suunnitteluvaiheessa määrittää selkeä toteutettavissa oleva tavoite, johon on päästävä viimeistään sprintin loppuun mennessä. Tavoitteita ei olisi tulisi muuttaa tai lisätä sen jälkeen, kun niistä on päätetty sprintin alussa. [Schwaber ja Sutherland, 2016]

Sprintin tavoitteet määräytyvät perinteisesti tuotteen omistajan määrittämän tehtävälistan (engl. *product backlog*) prioriteettien mukaan. Tehtävälistan sisältö perustuu puolestaan käyttäjätarinoihin (engl. *user story*), jotka kuvastavat asiakkaan kuvaamia ohjelmiston vaatimuksia, kuten ominaisuuksia. Tarinat voidaan kirjoittaa esimerkiksi omille tarinakorteilleen, minkä avulla niitä voidaan helpommin pitää yleisesti näkyvillä, ja niihin voidaan lisätä havainnollistavia muistiinpanoja esimerkiksi piirustusten muodossa. Käyttäjätarinoita voi verrata vesiputousmallin alussa laadittavaan vaatimusmäärittelyyn. [McInerney ja Maurer, 2005]

Scrumtiimi koostuu perinteisesti tuotteen omistajasta (engl. *product owner*), kehitystiimistä (engl. *development team*) sekä Scrum Masterista. Tuotteen omistaja edustaa sidosryhmiä ja tällä on päävastuu tuotteesta ja sen kehityksen edistymisestä. Tämän tulee pitää huolta muun muassa siitä, että kehitystiimi on ajan tasalla työtehtävistään ja että annetut tehtävät ovat selkeitä ja loogisesti priorisoituja [Schwaber ja Sutherland, 2016]

Kehitystiimi puolestaan on nimensä mukaisesti vastuussa tuotteen kehitystyöstä. Tiimin työskentely on itseohjautuvaa, eikä sitä tulisi johtaa kehitystiimin ulkopuolelta. Tiimin tulisi olla tarpeeksi pieni pysyäkseen ketteränä, mutta tarpeeksi suuri, jotta tavoitteisiin pääseminen pysyy realistisena ilman, että työkuorma kasvaa liian suureksi. Schwaberin ja Sutherlandin [2016] mukaan tiimissä tulisikin olla maksimissaan yhdeksän jäsentä, jotta sitä pystyy koordinoimaan järkevästi. Kehitystiimin vastuulla on itse kehitystyön lisäksi myös päivittäisten scrumien (engl. *Daily Scrum*) järjestäminen sprintin aikana, joissa käydään tiimin sisäisesti läpi seuraavan 24 tunnin suunnitelma muun muassa siitä, mitä ollaan tekemässä ja mitä tullaan vielä tekemään. Tämä vaihe on myös kuvattuna kuvassa 3. [Schwaber ja Sutherland, 2016].

Scrum Masterin rooli on pitää huolta siitä, että Scrumin periaatteet on ymmärretty ja että työskentely myös sujuu niiden mukaisesti. Hän muun muassa varmistaa, että Scrumiin liittyvät säännölliset palaverit pidetään sovitusti ja että niiden sisältö ja tarkoitus ymmärretään koko tiimin laajuisesti. [Schwaber ja Sutherland, 2016]

Scrumin hyötyinä voidaan nähdä muun muassa sen yksinkertaisuus konseptitasolla, mikä myös helpottaa sen oppimista. Toinen mainitsemisen arvoinen hyöty on sprintteihin kohdistuva retrospektiivisuus, eli onnistumisten ja epäonnistumisten arviointi

sprinttipalavereissa. Tämän avulla projektitiimeillä on jatkuvasti mahdollisuus parantaa omaa työskentelyään ja kiinnittää huomiota oikeisiin asioihin. [Sergeev, 2017]

Scrumin negatiivisempänä puolena voidaan nähdä se, että metodologian asettamia sääntöjä tulisi seurata erittäin tarkasti, esimerkiksi palaverikäytäntöihin, sprinttien pituuteen sekä tiimien kokoihin liittyen. Tämä voi aiheuttaa ongelmia sellaisten yllättävien vastoinkäymisten kohdalla, joihin Scrumin asettamat toimintatavat eivät välttämättä pysty suoraan taipumaan. [Sergeev, 2017]

### 3.3 Extreme Programming (XP)

Extreme Programming (lyh. *XP*) pyrkii Scrumin tapaan madaltamaan muutoksiin vastaamisen kynnystä suosimalla nopeaa ja iteratiivista toimitusta. Siinä missä Scrumin keskiössä voidaan nähdä projektijohtaminen ja sen menetelmät, keskittyy XP huomattavasti tarkemmin itse ohjelmiston kehitystyöhön ja siinä käytettäviin työskentelymetodeihin. [Beck ja Andres, 2004]

Wells [2009] listaa viisi arvoa, joihin XP ja sen säännöt perustuvat. Nämä arvot ovat *yksinkertaisuus*, *kommunikaatio*, *palaute*, *kunnioitus* sekä *rohkeus*. *Yksinkertaisuudella* tarkoitetaan sitä, että kehitystyössä tehdään vain tarpeelliset ja pyydyt asiat, eikä sen enempää. Näin saadaan minimoitua kuluja sekä hukkaa.

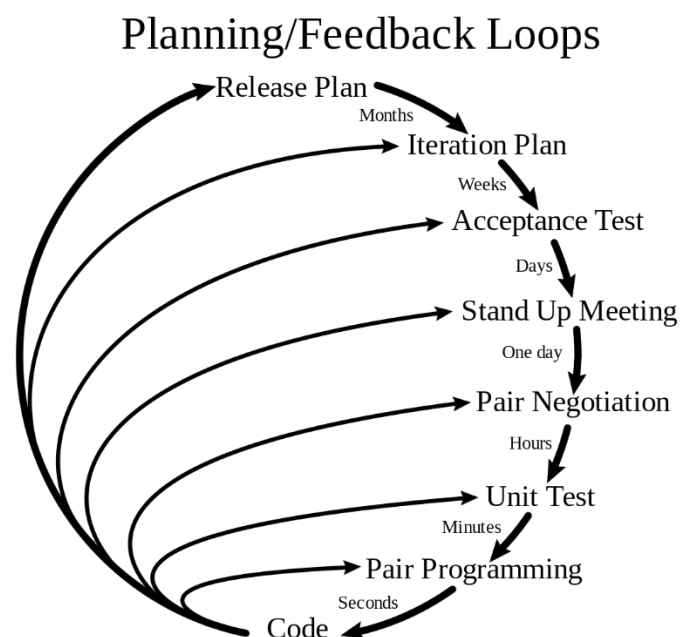
*Kommunikaatio* tarkoittaa, että kehitystiimin tulee kommunikoida niin vaatimusmäärittelystä kuin koodauksestaikin säännöllisesti kasvotusten. Työskentelytapana XP hyödyntääkin pariohjelmointia (engl. *Pair Programming*), jossa kaksi ihmistä työskentelevät aina yhdessä saman tietokoneen ääressä. Metodin tarkoituksena on saada jatkuvaa suullista palautetta työstettävästä koodista sekä helpottaa ongelmanratkaisua tilanteissa, joissa yksinäinen ohjelmoija saattaisi jäädä jumiin. Kyseisen työskentelytavan on myös havaittu vaikuttavan tarvitun koodin määrään ja laatuun positiivisesti. [Extreme Programming, 2019]

Pariohjelmoinnin lisäksi eräänä XP:n kulmakivenä voidaan nähdä yksikkötestaaminen (engl. *unit testing*), jossa hyödynnetään kehittäjien ylläpitämiä automatisoituja testejä. Tällaisen testauksen avulla voidaan tehokkaasti varmistua uusien implementaatioiden

toimivuudesta, mikä vähentää myöhempää tarvetta ohjelmointivirheiden korjaukselle. [Beck ja Andres, 2004]

*Palautteella* tarkoitetaan sitä, että ohjelmistosta pyritään toimittamaan jatkuvasti toimivia versioita, joita pystytään demonstroimaan niin sisäisesti kuin myös asiakkaalle [Wells, 2009].

Scrumin tapaisten kuukausittaisten sprinttien sijaan XP:n iteraatiot tehdään viikoittaisissa sykleissä. Syklin alussa asiakkaalle raportoidaan projektin sen hetkinen tilanne, minkä jälkeen tämän kanssa käydään keskustelua siitä, millaisia ominaisuuksia tuotteeseen halutaan alkavan viikon kuluessa implementoida. Näin työn kulusta saadaan jatkuvasti palautetta ja projektin osia pystytään hallinnoimaan ja priorisoimaan mahdollisimman ajankohtaisella tiedolla. Scrumista poiketen XP asennoituu vapaammin prioriteettien muuttamiseen kesken syklin. XP:n iteraatiota ollaan kuvattu kuvassa 4. [Extreme Programming, 2019]



Kuva 4. Kaavio XP:n prosesseista. [Wells, 2009]

*Kunnioitus* tarkoittaa yksinkertaisesti sitä, että tiimin jäsenet osaavat antaa arvoa kollegoilleen ja asiakkailleen, ja että molemmat osapuolet osaavat kunnioittaa toistensa osaamista. *Rohkeudella* taas tarkoitetaan rohkeutta kertoa totuus projektin edistymisestä ja tilannearvioista peittelemättä totuutta. [Wells, 2009]

Nopeista iteraatioista johtuen asiakkaan rooli onkin XP:ssä huomattavasti suurempi kuin esimerkiksi Scrumissa. Asiakas on XP:ssä hyvin samankaltaisessa roolissa kuin tuotteen omistaja Scrumissa, sillä tämä vastaa muun muassa käyttäjätarinoista ja ominaisuuksien priorisoinnista, eli näin ollen myös projektin tehtävälistasta. Tästä syystä asiakas osallistuu myös aktiivisesti palavereihin ja tämä voidaanakin mieltää osaksi projektitiimiä. [Kukhnavets, 2018]

Kuten voidaan todeta, XP:n hyviä puolia ovat esimerkiksi pariohjelmoinnin tuloksena syntyvä yksinkertainen ja säännöllisesti testattu koodi sekä jatkuva palautteen saaminen työstä. Liiallinen koodikeskeisyys voidaan nähdä XP:ssä myös riskitekijänä, mikäli se johtaa tuotedesignin heikompaan laatuun. [Kukhnavets, 2018]

### **3.4 Kanban**

Kanban on lean-ajatteluun perustuva metodologia, jonka peruseriaatteena on kehittää työnkulkua ja tehdä siitä helpommin hallittavaa. Mallin esitteli alun perin Toyotalle työskennellyt Taiichi Ohno 1940-luvulla, mutta IT-maailmassa konseptia hyödynsi ensimmäisten joukossa David J. Anderson vuonna 2004 [Kanban, 2018]. Kanban ei ole Scrumin ja XP:n tapainen kokonaisvaltainen toimintamalli, eikä sen ole täten määrä korvata jo käytössä olevia yrityksen toimintatapoja. Se on ennemminkin työkalu, jolla voidaan kehittää jo valmiiksi omaksuttuja työskentelymetodeja.



Kuva 5. Esimerkki klassisesta kanban-taulusta. [Kanban, 2018]

Kanbanin keskiössä on työnkulun visualisointi, joka toteutetaan yleisimmin kuvassa 5 havainnollistetun kanban-taulun sekä siihen liitettävien muistilappujen tai korttien avulla. Kanbanin hyötyinä nähdään esimerkiksi ylimääräisen ja tarpeettoman työn karsiutuminen, mikä auttaa projektitiimiä keskittymään vain oleellisiin tehtäviin ja näin ollen priorisoimaan työnkulkua tehokkaammin. Sen rakenne on visuaalisen luonteensa ansiosta helposti ymmärrettävä ja se voidaan myös tarvittaessa implementoida toisten metodologioiden päälle. Taulun avulla projektitiimit voivat suunnitella ja seurata kunkin prosessin etenemistä ja näin ollen pitää huolta järjestelmän osien säännöllisistä toimitusajoista asiakkaalle. [Kanban, 2016]

Kanbanin tehokkuus on yksinkertaisesta rakenteestaan huolimatta täysin riippuvainen siitä, että projektitiimi sitoutuu pitämään taulun jatkuvasti ajan tasalla. On myöskin projektitiimistä itsestään kiinni, että lapuilla havainnollistetut työt etenevät seuraavaan vaiheeseen oikeassa aikataulussa, eivätkä näin ollen pääse kasaantumaan. [Kanban, 2016]



## 4. Käyttäjäkeskeinen suunnittelu ohjelmistotuotannossa

Käyttäjäkeskeinen suunnittelu (engl. *User-Centered Design, UCD*) on aiheena erittäin laaja ja se pitää sisällään valtavan määrän erilaisia teorioita, ohjeistuksia sekä käytäntöjä.

Gulliksen et al. [2016] mukaan käyttäjäkeskeinen ohjelmistosuunnittelu (engl. *User-Centered Software Design, UCSD*) on käyttäjien ja käytön kontekstin ymmärtämistä sekä tähän tietoon perustuvan vaatimusmäärittelyn ja siitä syntyvien designien arviointia. Keskeisenä tavoitteena on myös, että käyttäjä pidetään mahdollisimman tiiviisti mukana projektissa koko sen elinkaaren ajan ja että tämä osallistuu aktiivisesti kehitykseen. Käyttäjäkeskeisen suunnittelun ammattilaiset vastaavat esimerkiksi käyttökokemuksen, käyttöliittymien sekä käytettävyyden suunnittelusta. Nadikattu [2016] määrittelee UCD:n periaatteet seuraavasti:

- Asiakkaiden jatkuva osallistuminen
- Käyttäjän odotusten ja tarpeiden ymmärtäminen
- Käyttäjien ja teknologian suhteen ymmärtäminen
- Designien iteroiminen
- Monitieteelliset designit

Tässä luvussa esitellään korkealla tasolla muutamia keskeisiä teemoja, jotka nousevat esille puhuttaessa nimenomaan ohjelmistotuotannon käyttäjäkeskeisestä suunnittelusta ja sen yleisistä käytännöistä. Näiden teemojen esitleminen on tutkimuksen kannalta oleellista, jotta voidaan ymmärtää paremmin niiden integroimisen haasteita ja mahdollisuuksia ketterässä ohjelmistotuotannossa.

### 4.1 Käyttäjäkokeemus (UX)

Käyttäjäkokeemus (engl. *User Experience, UX*) on käyttäjäkeskeisen suunnittelun osa, joka keskittyy nimensä mukaisesti siihen, kuinka käyttäjät kokevat järjestelmän. ISO 9241-210 -standardin [2009] määritelmän mukaan käyttäjäkokeemus koostuu käyttäjän tunteista, uskomuksista, mieltymyksistä, fyysisistä ja psyykkisistä reaktioista sekä käytöksestä ja saavutuksista, jotka tapahtuvat ennen järjestelmän käyttöä sen aikana sekä

sen jälkeen. Määritelmän mukaan tähän vaikuttavat kolme päätekijää, jotka ovat järjestelmä, käyttäjä sekä käyttöympäristö [ISO 9241-210, 2009].

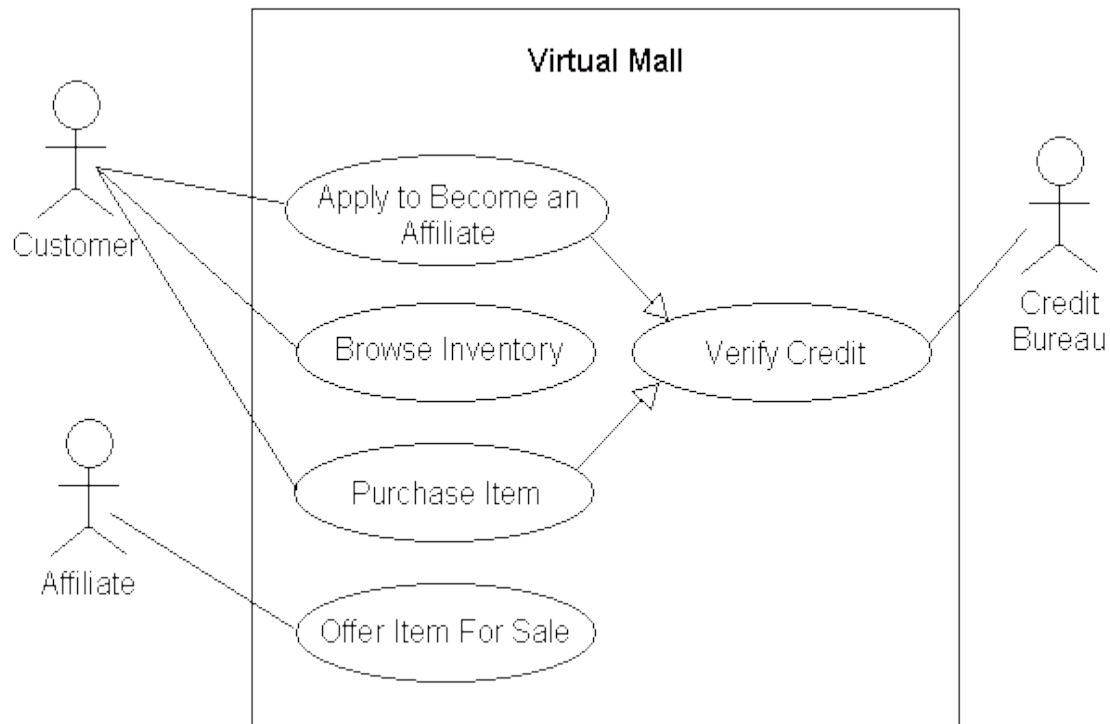
Jotta käyttäjäkokemuksesta saadaan mahdollisimman positiivinen, on kaikki kolme tekijää otettava huomioon suunnittelutyön jokaisessa vaiheessa. Erityisen tärkeänä vaiheena pidetään jo ennen kehitystä alkavaa esitutkimustyötä, minkä avulla pystytään ymmärtämään käyttäjiä ja näiden tarpeita jo projektin alusta lähtien [Adikari et al., 2009; Brhel et al., 2015; Da Silva et al., 2012; McInerney ja Maurer, 2005]. Nadikattu [2016] nimeää oleellisimmista tutkimus- ja suunnittelutyön vaiheista seuraavat:

- Käyttöympäristön ymmärtäminen
- Käyttäjän ja organisaation vaatimusten määrittely
- Vaatimusten analysointi suhteessa rajoituksiin
- Designien rakentaminen
- Designien arviointi suhteessa vaatimuksiin

Käyttöympäristön (engl. *context of use*) ymmärtämiseen sisältyy käyttäjien tunnistaminen ja ymmärtäminen sekä se, mihin järjestelmää tullaan käyttämään ja millaisissa olosuhteissa. Käyttöympäristöön kuuluvat fyysisen ympäristön, kuten työtilan, lisäksi myös tekninen ympäristö (laitteet) sekä sosiaalinen ympäristö (työntekijät ja asenteet). Käyttöympäristön analysoimisen yhteydessä eräs oleellinen työvaihe on myös eri käyttäjäryhmien määrittely. Näitä ryhmiä ovat ensisijaiset käyttäjät, joihin kuuluvat todelliset loppukäyttäjät, eli eivät välttämättä järjestelmän ostajat. Toissijaisiksi käyttäjiksi lasketaan käyttäjät, jotka eivät ole suoraan tekemisissä järjestelmän kanssa, mutta joihin järjestelmän käyttö vaikuttaa. Näihin kuuluvat esimerkiksi markkinointi sekä asiakastuki. Jokaisen eri käyttäjäryhmän kohdalla on erityisen tärkeää määrittää käyttäjän päärooli ja tavoitteet, jotta järjestelmän hyödyistä voidaan varmistua eri käyttäjien kohdalla. Näiden kaikkien tekijöiden määrittäminen ja ymmärtäminen auttaa suunnittelijoita havaitsemaan paremmin käyttäjien tarpeita sekä käytettävyyteen liittyviä ongelmia. [Maguire, 2001]

Vaatimusten määrittelyn tavoitteena on varmistaa, että järjestelmä vastaa käyttäjän ja yrityksen tavoitteita. Käyttäjäkeskeisen suunnittelun näkökulmasta tämä tarkoittaa usein käyttötapausten (engl. *use case*) ja niihin liittyvien kaavioiden suunnittelemista. Käyttötapaus on kuvaus järjestelmään liittyvästä toiminnasta, kuten sisäänkirjautumisesta, joiden avulla pystytään paremmin hahmottamaan, millaisia vaatimuksia ja toimintoja

järjestelmän tulee toteuttaa. Käyttötapaukselle kuvataan yleensä nimi, kuvaus, toimijat sekä ennakkoehdot, mutta tähän ei ole olemassa tarkkoja sääntöjä. Kuvassa 6 on kuvattu yhdenlainen esimerkki käyttötapauskaaviosta [Brandenburg, 2018]



Kuva 6. Esimerkki käyttötapauskaaviosta, joka kuvaa verkkokaupassa asiointia.

[Use cases, 2019]

Oikeiden käyttäjien haastattelun lisäksi käyttötapausten suunnittelussa käytetään joskus myös hyväksi *persoonia*, eli keksittyjä käyttäjiä. Persoonien tarkoituksena on kuvata järjestelmän todellisia käyttäjiä ja näiden tavoitteita fiktiivisten hahmoprofiilien kautta. Vaikka persoonista puhutaan paljon käyttäjäkokemuksen yhteydessä, on niiden todellisesta hyödystä eriäviä mielipiteitä, koska ne eivät välttämättä kuvaa tarpeeksi luotettavasti oikeita käyttäjiä ja näiden tarpeita [Matthews et al., 2012]. Blomquist ja Arvolan [2002] mukaan persoonia voidaan hyödyntää suuntaa-antavina malleina esimerkiksi käyttäjätestien suunnittelussa.

Käyttötapausten ja vaatimusmäärittelyn pohjalta luotavien designien rakentaminen voidaan suorittaa osissa ja monilla eri tavoilla. Alussa rakennetut prototyypit voivat olla esimerkiksi vain piirustuksia paperilla, kunnes niitä hiotaan vähitellen mock-upeiksi ja

lopulta valmiiksi designeiksi [Cao, 2016]. Oikeanlaisten prototyyppien rakentaminen on kiinni projektin luonteesta, ja jokaisella erilaisella versiolla on omat vahvuutensa ja heikkoutensa [Cao, 2016]. Esimerkiksi matalan tarkkuuden (engl. *low fidelity*) paperiprototyyppien rakentaminen on nopeaa ja halpaa, mutta ne ovat samalla epärealistisia verrattuna digitaaliseen vastineeseen [Cao, 2016]. Matalan tarkkuuden prototyyppien etuna voidaan nähdä se, että ne helpottavat demonstraatiotilanteissa kokeellisten ja nopeiden muutosten tekemisen esimerkiksi piirtämällä [Pernice, 2016]. Niiden avulla asiakkaalle on myös helpompi tehdä selväksi, että kyseessä on keskeneräinen prototyyppi, eikä valmis versio [Pernice, 2016].

Korkean tarkkuuden (engl. *high fidelity*) prototyypit, eli lopullista tuotetta paremmin vastaavat digitaaliset versiot, ovat nimensä mukaisesti tarkempia demonstraation välineitä, mutta niiden luominen ja muokkaaminen on paperiprototyyppijä työläämpi prosessi. Jatkuvasta iteroinnista johtuen projektitiimin onkin tehokasta hyödyntää tilanteesta riippuen sekä matalan että korkean tarkkuuden prototyyppijä. [Cao, 2016]

Luotuja designeja pitää pystyä arvioimaan jatkuvasti, jotta voidaan varmistua siitä, että ne vastaavat määriteltyjä vaatimuksia. Designien arviointia voidaan pitää yhtä tärkeänä prosessina kuin laadunvarmistusta ohjelmistokehityksessä. [Garcia et al., 2017]

## 4.2 Käytettävyys

Käytettävyyttä voidaan pitää yhtenä osana käyttäjäkokemusta. Huono käytettävyys johtaa usein huonoon käyttökokemukseen, kun taas hyvä käyttökokemus viestii usein siitä, että myös ohjelmiston käytettävyys on hyvällä mallilla. Siinä missä käyttäjäkokemus voidaan mieltää käyttäjän subjektiiviseksi tunteeksi, liittyy käytettävyys ISO 9241-11 -standardin [1998] mukaan ennemminkin tavoitteen saavuttamisen tehokkuuteen ja tästä koettuun tyytyväisyyteen.

Käytettävyys pitää sisällään myös järjestelmän käytön oppimisen, sen ymmärtämisen, saavutettavuuden, muistettavuuden, luettavuuden ja helppokäyttöisyyden. Käytettävyys mielletään helposti vain järjestelmän käyttöliittymää koskevaksi osa-alueeksi, mutta se liittyy järjestelmässä kaikkeen, mikä tukee käyttäjää tavoitteiden täyttämisessä. [Nadikattu, 2016]

Järjestelmän käytettävyyttä voidaan arvioida monella eri menetelmällä ja monesta eri näkökulmasta. Eräs klassisimmista käytettävyydsarvioinnin esimerkeistä on Nielsenin [1994] heuristinen analyysi, jossa tarkastellaan 10 käyttöliittymäsuunnittelun yleisperiaatetta. Nämä periaatteet ja niiden määritelmät on esitelty taulukossa 6.

<b>Heuristiikka</b>	<b>Määritelmä</b>
Näkyvyys	Järjestelmän tulee kertoa käyttäjälle, mitä tapahtuu.
Yhteensopivuus oikean maailman kanssa	Järjestelmän tulee puhua kieltä, jota käyttäjä ymmärtää.
Käyttäjän kontrolli ja vapaus	Käyttäjälle tulee tarjota mahdollisuus palata edelliselle sivulle tai vaiheeseen.
Yhdenmukaisuus ja standardit	Yhdenmukaisuus kielessä, tilanteissa ja toiminnoissa.
Virheiden ehkäisy	Ehkäistään tilanteita, jotka johtavat virheilmoituksiin.
Tunnistettavuus	Järjestelmän toimintojen tulisi olla helposti nähtävillä, jotta käyttäjän ei tarvitse muistaa kaikkea ulkoa.
Joustavuus ja tehokkuus	Järjestelmän tulisi toimia tehokkaasti niin kokeneille kuin kokemattomillekin käyttäjille.
Esteettisyys	Järjestelmän ei tulisi näyttää tarpeetonta informaatiota käyttäjille.
Virheistä toipuminen	Virheilmoitusten tulee kertoa selvällä kielellä ongelmasta ja miten se voidaan korjata.
Ohjeet	Järjestelmän tulisi tarjota dokumentaatiota sen käyttämisestä.

Taulukko 6. Nielsenin 10 käytettävyyden heuristiikkaa ja niiden selitykset.

[Nielsen, 1994]

Käytettävät menetelmät ja tutkittavat attribootit on syytä valita aina tapauskohtaisesti. Gulliksen et al. [2014] teettämän kyselytutkimuksen mukaan parhaat käytettävyystudkimuksen menetelmät ovat sellaisia, jotka on toteutettu oikean käyttäjän kanssa. Saman tutkimuksen mukaan huonoimmat menetelmät olivat sellaisia, joissa oikea käyttäjä on korvattu ammattilaisella, suorituskäytöstä tai muiden koehenkilöiden täyttämällä kyselylomakkeilla [Gulliksen et al., 2014].

Käytettävyystudkimuksia voidaan suorittaa tilanteesta riippuen niin järjestelmän todellisessa käyttöympäristössä kuin myös vaihtoehtoisesti tutkimukseen soveltuvissa laboratorio-olosuhteissa. Molempien tutkimustapojen yhdistäminen on suositeltavaa, koska niissä voidaan arvioida käytettävyyttä eri näkökulmista. Laboratorio-olosuhteissa voidaan tutkia helpommin ennalta määritettyjen käytettävyyssuhteiden toteutumista, kun taas kenttäkokeella pystytään löytämään todennäköisemmin uusia käytettävyyssuhteita, jotka ilmenevät mahdollisesti vain oikeassa käyttöympäristössä. [Maguire, 2001]

## 5. Ketterä käyttäjäkeskeinen suunnittelu

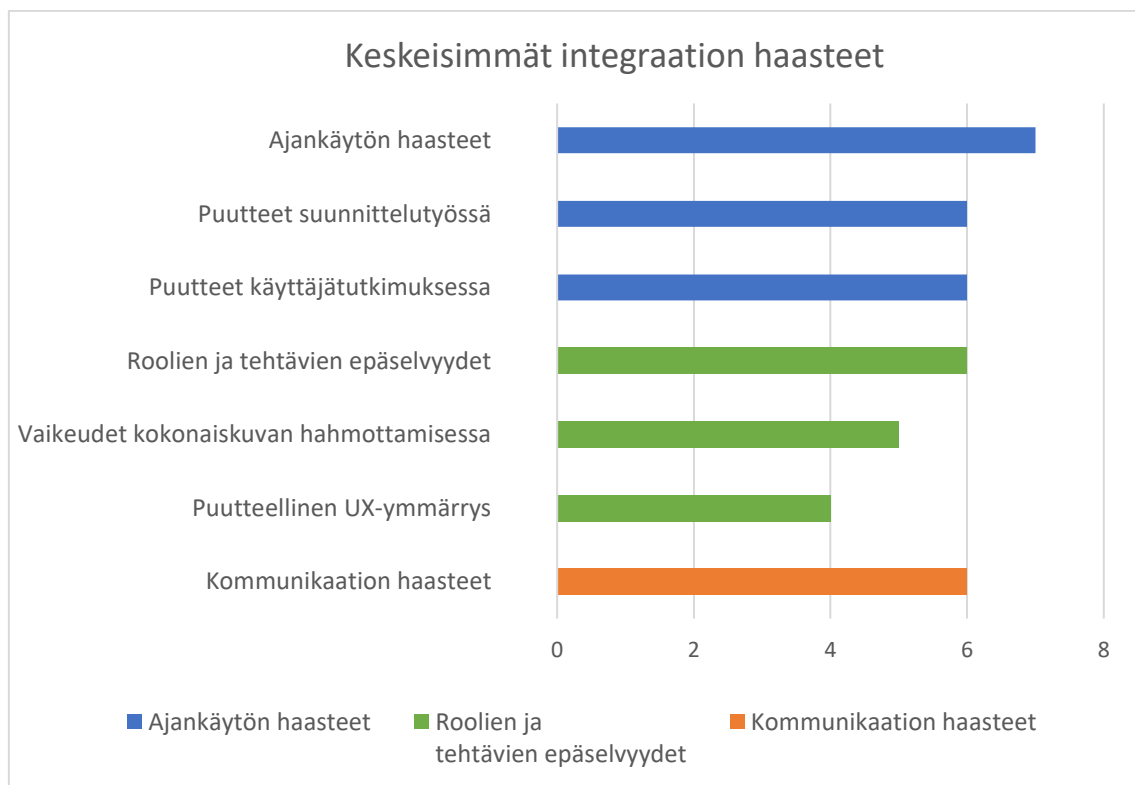
Tässä luvussa käydään läpi systemaattiseen kirjallisuuskatsaukseen valikoituja tutkimuksia, niissä esiteltyjä menetelmiä, tuloksia sekä pohdintoja. Tarkoituksena on saada kokonaisvaltainen kuva siitä, millaisia hyötyjä ja ongelmia ketterien menetelmien ja käyttäjäkeskeisen suunnittelun integraatiossa voi ilmetä sekä millaisten menetelmien ja näkökulmien avulla aihetta ollaan lähestytty erilaisissa tapauksissa. Luvussa esitellään aluksi kirjallisuuskatsauksen tutkimuksissa ilmenneitä haasteita, jotka esitellään kolmessa eri kategoriassa. Tämän jälkeen esitellään tutkimuksessa todettuja integraation hyötyjä sekä haasteisiin vastaavia kehitysmahdollisuuksia. Nämä esitellään haasteiden tapaan kolmessa eri kategoriassa.

Ketterän kehityksen ja käyttäjäkeskeisen suunnittelun yhdistäviin työskentelymetodeihin on viitattu kirjallisuuskatsauksen julkaisuissa muutamalla eri termillä. Näitä ovat esimerkiksi ”Agile UXD” [Da Silva et al., 2018] sekä ”Agile User-Centered Design” [Sy, 2007]. Kaikkien kirjallisuuskatsauksen tutkimusten näkökulmasta näillä termeillä tarkoitetaan pohjimmiltaan samaa asiaa, eli käyttäjäkeskeisen suunnittelun prosessien sulauttamista osaksi ketterää ohjelmistokehitystä. Tässä tutkielmassa päädyttiin käyttämään selkeyden ja johdonmukaisuuden vuoksi termiä Agile UXD.

Näiden kahden menetelmän prosessien sulauttamisen perimmäisenä tarkoituksena on luoda yhteinen toimintamalli, jolla saadaan hyödynnettyä molempien menetelmien parhaita ominaisuuksia. Ketterän kehityksen näkökulmasta tämän voidaan katsoa tarkoittavan iteratiivista ja inkrementaalista kehitystyötä, jonka keskiössä on joustava muutokseen vastaaminen, turhan työn kitkeminen ja sidosryhmien aktiivinen osallistuminen. Käyttäjäkeskeisen suunnittelun näkökulmasta taas tärkeimpänä arvona voidaan nähdä käytettävyyteen ja käyttökokemukseen painottuva suunnitelmallisuus, jossa käyttäjän tarpeet ovat etusijalla. [Brhel et al., 2015]

## 5.1 Haasteet

Tässä kohdassa perehdytään erilaisiin haasteisiin, joita Agile UXD:ssa on havaittu esiintyvän tutkimusten pohjalta. Kohdassa 5.2 puolestaan käydään tarkemmin läpi kehitysehdotuksia, joiden avulla kyseisiä haasteita pystyttäisiin tutkimustiedon mukaan minimoimaan.



Kuva 7. Keskeisimpien integraatiohaasteiden esiintyminen valituissa tutkimuksissa.

Kuvassa 7 ollaan kuvattu tutkimuksissa esiintyneet keskeisimmät integraatiohaasteet ja niiden esiintymisten määrät kirjallisuuskatsaukseen valituissa tutkimuksissa. Haasteet ollaan jaettu luonteensa perusteella kolmeen kategoriaan, jotka ovat ajankäytölliset haasteet, roolien ja tehtävien epäselvyys sekä kommunikaation haasteet. Haasteita käsitellään seuraavaksi näiden kategorioiden mukaisesti omissa alakohdissaan.



### 5.1.1 Ajankäytön haasteet

Tässä alakohdassa käydään läpi tutkijoiden havaintoja ajankäyttöön liittyvistä haasteista. Kuten kuvasta 7 voidaan havaita, ajankäytölliset ongelmat ovat projektitiimien suurimpia haasteita ketterän kehityksen ja käyttäjäkeskeisen suunnittelun integroimisessa. Tällaisilla ongelmilla tarkoitetaan tässä yhteydessä pääasiallisesti sitä, ettei suunnittelutiimi ole ehtinyt keskittymään tarpeeksi käyttäjäkeskeisen suunnittelun oleellisiin prosesseihin, mikä on puolestaan johtanut haasteisiin projektityöskentelyssä.

Gulliksen et al. [2016] ja Sy [2007] mukaan Scrum-kehityksen lyhyet 2-4 viikon sprintit rajoittavat oleellisesti UX-asiantuntijoiden työtä esimerkiksi käytettävyystudkimusten kannalta. Tämä saattaa johtaa muun muassa siihen, että käytettävyystudkimuksia ei ehditä tekemään perusteellisesti tai pahimmassa tapauksessa ei lainkaan, mikä puolestaan hankaloittaa kokonaisvaltaisen vision ylläpitämistä käyttökokemuksesta [Gulliksen et al., 2016; McInerney ja Maurer, 2005]. Chamberlain, Sharp ja Maiden [2006] raportoivat samanlaisia huomioita Extreme Programmingia hyödyntäneestä projektista ja toteavat, että tiukat aikataulut hankaloittavat käyttäjien ottamista huomioon suunnittelutyössä. He toteavat myös, että aikaresurssien eroavaisuudet kehitystiimin ja suunnittelutiimin välillä voivat aiheuttaa keskinäisiä ristiriitoja, sillä kehitystyö vie tavanomaisesti enemmän aikaa kuin suunnittelutyö [Chamberlain, Sharp ja Maiden 2006].

Kuusinen ja Väänänen-Vainio-Mattila [2012] puolestaan mainitsevat ajankäytön ongelmiin liittyen myös budjetointiongelmat, joiden takia UX-työlle ei oltu tutkimuksen case-esimerkissä allokoitu tarpeeksi ajallisia ja rahallisia resursseja. Tämän todettiin johtuvan siitä, ettei UX-suunnittelua ja sen merkitystä projekteille oltu ymmärretty johtaja asiakastasolla tarpeeksi hyvin. UX-suunnittelua kohdeltiinakin kyseisessä tapauksessa eräänlaisena lisäpalveluna, jonka asiakas pystyi halutessaan tilaamaan yritykseltä. Koska suunnittelua ei kohdeltu luonnollisena osana kehitystyötä, joutuivat asiakkaat joskus maksamaan jälkikäteen siitä, että huonolaatuista ohjelmistoa lähdettiin parantamaan. [Kuusinen ja Väänänen-Vainio-Mattila, 2012]

Da Silva et al. [2012] kuvaamassa case-tutkimuksessa UX-tiimin ajankäytölliset ongelmat johtuivat siitä, että tiimi oli mukana liian monessa projektissa samanaikaisesti. Tämä johti muun muassa siihen, ettei tiimillä ollut aikaa suorittaa minkäänlaista etukäteissuunnittelua sprinttejä varten. Samanaikainen työskentely monessa eri

projektissa aiheutti myös sen, etteivät UX-tiimiläiset kokeneet toimivansa projektien kehitystyön varsinaisena jäsenenä, vaan pikemminkin erillisenä, ulkoistettuna työvoimana. Tämä vaikutti myös negatiivisesti UX-tiimin sitoutumiseen ja keskittymiseen projekteissa. [Da Silva et al., 2012]

Budwig et al. [2009] tekemässä tutkimuksessa tarkasteltiin kahta erilaista työskentelytapaa Agile UXD -ympäristössä. Ensimmäisessä tavassa UX-tiimi aloitti työskentelyn yhdessä kehitystiimin kanssa, kun taas toisessa vaiheessa UX-tiimi aloitti etukäteissuunnittelun jo ennen varsinaisen sprintin alkamista. Ensimmäisessä tavassa UX-tiimi joutui taistelemaan jatkuvasti aikaa vastaan, koska suunnittelutyötä oli tehtävä samaan tahtiin kehitystyön kanssa, jolloin muutoksia tuli jatkuvasti, eikä vaatimusmäärittelyä dokumentoitu tarpeeksi selkeästi. Toisessa tavassa haasteena oli palautteen saamisen hankaluus kehitystiimiltä, koska UX-tiimi teki työtä kehitystiimin edellä. Tämän lisäksi bugien ja uusien vaatimusten syntyessä UX-tiimillä oli aikataulullisia hankaluuksia ottaa huomioon muuttuneet vaatimukset ja sen lisäksi työstää seuraavan sprintin suunnitelmaa. Tilanne synnytti tiimille stressaavat työolosuhteet. [Budwig et al., 2009]

Kuten voidaan todeta, riittämätön ajankäyttö liittyy suoraan esimerkiksi puutteisiin suunnittelutyössä ja käyttäjätutkimuksessa. Ajankäytön haasteet eivät kuitenkaan ole kokonaisuudesta erillinen ongelma vaan niiden pohjimmainen syy saattaa liittyä esimerkiksi johtotason puutteelliseen ymmärrykseen UX-suunnittelun merkityksestä [Kuusinen ja Väänänen-Vainio-Mattila, 2012].

### **5.1.2 Roolien ja tehtävien epäselvyydet**

Kuten kuvasta 7 ja aiemmassa alakohdassa mainituista esimerkeistäkin voidaan todeta, saattavat ymmärrys käyttäjäkeskeisen suunnittelun merkityksestä sekä UX-suunnittelijoiden roolista olla joissain tilanteissa epäselvä niin johdolle, asiakkaille kuin myös kehitystiimille. Roolien ja tehtävien epäselvyydet voivat puolestaan vaikuttaa negatiivisesti myös projektitiimille allokoitujen ajallisten resurssien käyttöön [Kuusinen ja Väänänen-Vainio-Mattila, 2012].

Gulliksen et al. [2016] suorittaman tutkimuksen mukaan mielipide UX-asiantuntijoista oli, että nämä sopivat harvoin mukaan ketterään työskentelykulttuuriin. Tämä voidaan havaita muun muassa arvojen ja asenteiden välisinä eroavaisuuksina suunnittelu- ja kehitystiimin välillä; siinä missä kehitystiimi arvostaa nopeutta ja tiiminsisäistä kommunikaatiota, arvostavat UX-asiantuntijat enemmän järjestelmän käytettävyyttä ja kommunikaatiota käyttäjien kanssa. Tämä näkyy myös siinä, että kehitystiimi näkee usein valmiin sovelluksen teknisestä näkökulmasta, kun taas suunnittelutiimi pikemminkin käytettävyyden ja käyttökokemuksen näkökulmasta. Tällaisen arvojen välisen kuilun vuoksi myös roolien ja vastuunjaon välillä esiintyy epäselvyyksiä. Gulliksen et al. [2016] mainitsevat tästä esimerkkinä tiimien välisen epäselvyyden siitä, kenen vastuulla järjestelmän käytettävyys on loppujen lopuksi. [Gulliksen et al., 2016]

Chamberlain, Sharp ja Maiden [2006] toteavat case-esimerkistään, että suunnittelu- ja kehitystiimin välisten vastuu- ja roolijakojen epäselvyys kumpusi pohjimmiltaan johtotasolla tehdyistä sekavista päätöksistä, joiden koettiin vaikuttavan haitallisesti kunkin tiimin omiin valtuuksiin.

Kuten Kuusinen ja Väänänen-Vainio-Mattila [2012] totesivat, ymmärtämättömyys UX-työn merkityksestä johti laadullisiin ja ajankäytöllisiin ongelmiin. Tässä tapauksessa ongelmat johtuivat Chamberlain, Sharpin ja Maidenin [2006] tapaan pohjimmiltaan johtotason ymmärryksen puutteesta, mikä puolestaan johti siihen, ettei myyntitiimi osannut välittää UX-suunnittelun merkitystä asiakkaille tarvittavalla tavalla. Yleinen käsitys kyseisessä case-yrityksessä olikin, että tehokkaan UX-suunnittelun toteuttaminen kasvattaa projektin hintaa. Tästä huolimatta suoritettujen kyselytutkimusten ja haastattelujen tuloksista kävi ilmi, että UX-suunnittelua pidettiin tärkeänä osana projektia ja että huonosta käyttökokemussuunnittelusta seuraa huonoja tuloksia. [Kuusinen ja Väänänen-Vainio-Mattila, 2012]

Kuusinen et al. [2012] suorittamassa toisessa case-tutkimuksessa johtotasolta kumpuavat vastuunjaon ongelmat, aikatauluongelmat sekä huono kommunikaatio aiheuttivat sen, että UX-tiimin roolia ei ymmärretty tai sitä väheksyttiin, jolloin sille annettiin myös kehitystiimiä vähemmän päätäntävaltaa projektissa. Tämä johti muun muassa siihen, että havaitut käytettävyysongelmat poistettiin herkemmin tehtävälialta kehitystiimin, tuotteenomistajan, tai muun tahon toimesta. Tutkimuksesta kävi myös ilmi, ettei UX:n

merkityksestä tai siitä vastaavan tiimin toimenkuvasta ollut yksimielistä näkemystä UX-tiimin ulkopuolella. [Kuusinen et al., 2012]

Kuten tutkimuksissa todetaan, voivat roolien ja tehtävien epäselvyydet aiheuttaa tiimien välisiä näkemyseroja sekä vastuunjaon ongelmia. Tämän seurauksena esimerkiksi käytettävyysongelmien löytämiseen ja korjaamiseen kohdistettu huomio voi kärsiä, eikä kaikkia ongelmia pystytä välttämättä ratkaisemaan [Kuusinen et al., 2012]. Lisäksi tiimien väliset näkemyserot voivat synnyttää asenneongelmia ja arvojen välisiä ristiriitoja [Gulliksen et al., 2016].

### 5.1.3 Kommunikaation haasteet

Haasteet kommunikaatiossa voivat esiintyä Agile UXD:ssa niin tiimien välisessä kanssakäymisessä kuin myös asiakas- ja käyttäjäkommunikaatiossa. Kuten edellisissä alakohdissa kerrotaan, johtaa puutteellinen ymmärrys projektitiimin jäsenten rooleista ja tehtävistä helposti myös uusiin ongelmiin esimerkiksi suunnittelutyön laadussa. Puutteellisen ymmärryksen voidaan katsoa puolestaan johtuvan siitä, että projektitiimillä on ongelmia eri sidosryhmien välisessä kommunikaatiossa.

Gulliksen et al. [2016] toteavat tutkimuksessaan, että palaute käyttökokemuksesta haetaan useasti käyttäjien edustajilta ja välikäsiltä sen sijaan, että sitä haettaisiin oikeilta loppukäyttäjiltä. Scrumin dokumentointivastaisesta luonteesta johtuen järjestelmän käytettävyyksvaatimuksia ei myöskään välttämättä evaluoida suoraan oikeiden käyttäjien kanssa. Palaute vaatimusten täsmällisyydestä ja niiden täyttymisestä saadaankin monesti vasta sen jälkeen, kun järjestelmä on jo toimitettu käyttäjille. Gulliksen et al. [2016] raportoivat myös, että tutkimuksessa haastateltujen kehittäjien mukaan päivittäisen kommunikaation ylläpitäminen asiakkaan ja tuotteenomistajan kanssa on usein haasteellista kiireiden vuoksi. [Gulliksen et al., 2016]

Kuusinen et al. [2012] tutkivat UX-ammattilaisten roolia suuressa ohjelmistoyrityksessä. Järjestetyssä kyselytutkimuksessa pyrittiin selvittämään, mitkä syyt vaikuttavat kehitystiimin epäonnistuneeseen yhteistyöhön UX-tiimin kanssa. Merkittävimmiä syiksi mainittiin liian myöhäinen yhteistyö (26.7%), riittämätön kommunikaatio (18.3%) sekä erimielisyydet päätöksenteossa (13.3%). Liian vähäisen kommunikaation todettiin

johtavan epäkettertiin käytäntöihin, kuten työvaiheiden huonoon tiedottamiseen ja tiimistä eristäytyneeseen työskentelyyn. UX-tiimiltä toivottiin myös kykyä iteratiivisempaan työskentelyyn. [Kuusinen et al., 2012]

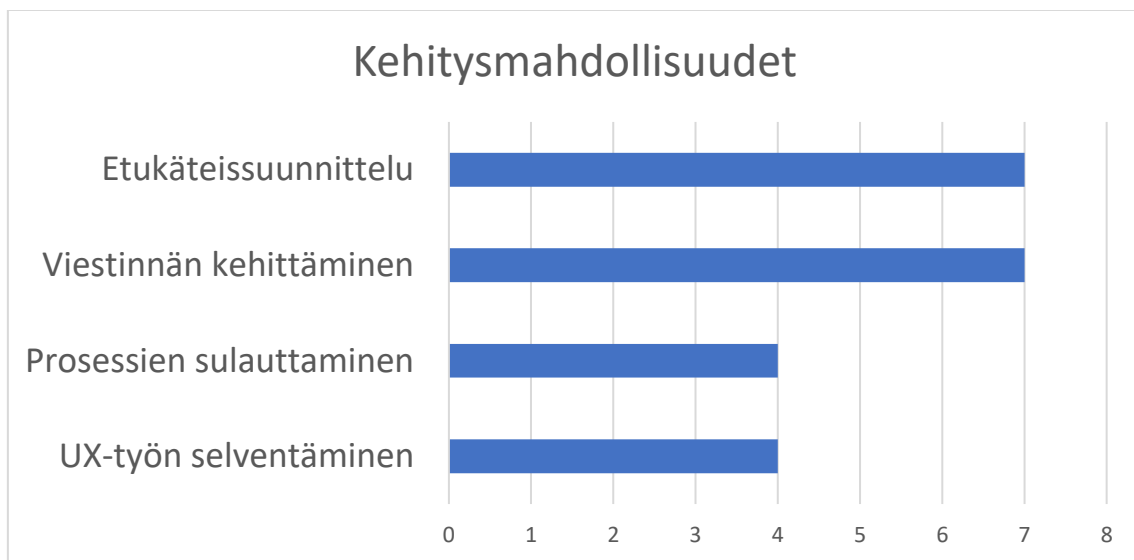
Kuusinen ja Väänänen-Vainio-Mattila [2012] suorittamassa kyselytutkimuksessa puolestaan 19.5% vastanneista mainitsivat sidosryhmien huonon kommunikaation Agile UXD:n keskeiseksi ongelmaksi ja 22% puolestaan loppukäyttäjien puutteellisen ymmärtämisen. Brhel et al. [2015], Chamberlain, Sharp ja Maiden [2006] sekä Da Silva et al. [2018] toteavat, että erilaisten artefaktien, kuten prototyyppien, mock-upien ja personien jakaminen tiimien ja asiakkaiden välillä on tärkeä kommunikaation väline Agile UXD:ssa ja se helpottaa muun muassa erilaisten ideoiden viestimistä puolin ja toisin.

Koska artefaktien tapaiset suunnitteludokumentit eivät ole perinteisesti suuressa osassa ketterän kehityksen näkökulmasta, saattaa suunnittelu- ja kehitystiimien välillä syntyä näkemyseroja niiden tarpeellisuudesta. Ongelmia voikin esiintyä, mikäli kaikkien projektin elementtien merkitystä tai tarpeellisuutta ei haluta tai pystytä syystä tai toisesta ymmärtämään. Tämä koskee niin projektitiimien kuin myös asiakkaiden työpanosta, sillä asiakkaiden aktiivinen osallistuminen projektissa on oleellisessa osassa sekä ketterän kehityksen että käyttäjäkeskeisen suunnittelun näkökulmasta. [Chamberlain, Sharp ja Maiden 2006]

Huono kommunikaatio saattaa pahimmillaan heijastua edellisessä alakohdassa esitettyihin roolien ja tehtävien epäselvyyksiin, mikä puolestaan voi johtaa ajankäytöllisiin haasteisiin projekteissa. Kommunikaation haasteina voidaan nähdä niin sidosryhmien tiukat aikataulut kuin myös kehitys- ja suunnittelutiimien perinteisten työskentelytapojen ja periaatteiden eroavaisuudet.

## 5.2 Hyödyt ja kehitysmahdollisuudet

Tässä kohdassa perehdytään erilaisiin Agile UXD:n hyötyihin ja kehitysmahdollisuuksiin, joita on esitetty kirjallisuuskatsaukseen valituissa tutkimuksissa. Kuten edellisestä kohdasta voidaan todeta, ketterän kehityksen ja käyttäjäkeskeisen suunnittelun yhdistämisessä voi ilmetä monenlaisia haasteita, joiden syyt liittyvät pääosin ajankäyttöön ja viestintään. Tämä on linjassa myös kuvassa 8 kuvattuihin kehitysmahdollisuuksiin, jotka on kuvattu niiden esiintymisten määrän mukaan samalla tavalla kuin kuvassa 7.



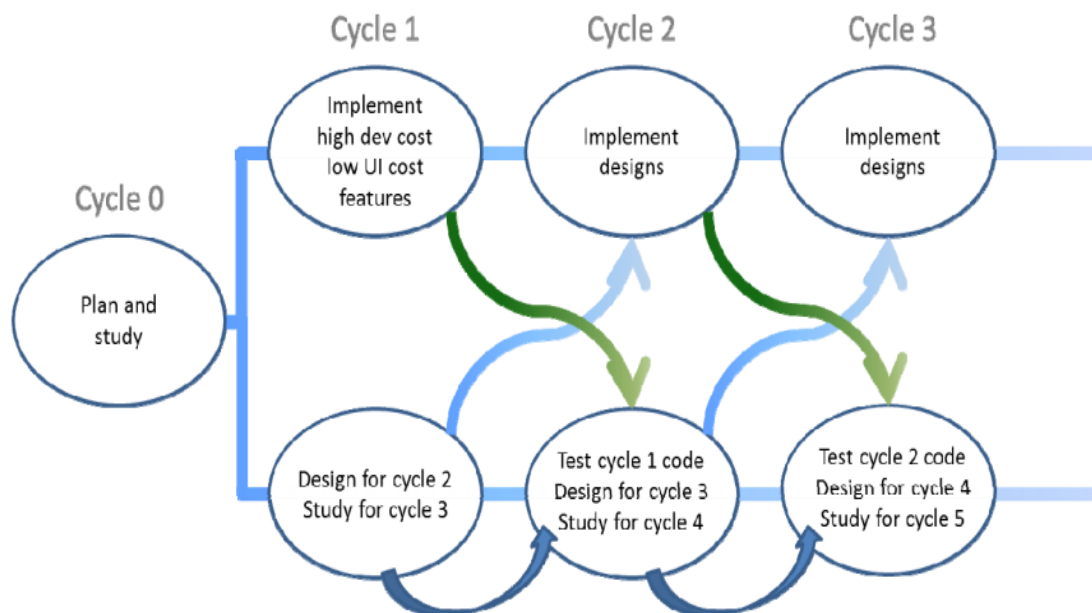
Kuva 8. Ehdotettujen kehitysmahdollisuuksien esiintyminen tutkimuksissa.

Kehitysmahdollisuuksia käsitellään seuraavissa kahdessa alakohdassa, jotka ovat etukäteissuunnittelu ja viestinnän kehittäminen. UX-työn selventäminen mainittiin neljässä eri tutkimuksessa kehitysmahdollisuutena, mutta sen katsottiin olevan osa viestinnän kehittämistä. Tästä syystä sitä käsitellään viestinnän kehittämisen kanssa samassa alakohdassa. Myös prosessien sulauttaminen mainittiin neljä kertaa, mutta sen katsottiin viittaavaan niin laajasti kaikkiin integraation eri osiin, että se pitää sisällään myös kaikki muut kuvassa 8 esiintyneet kehitysmahdollisuudet. Tästä syystä sitä ei päädytty käsittelemään erikseen omassa alakohdassaan.

### 5.2.1 Etukäteissuunnittelu

Kuten voitiin todeta, Agile UXD:ssa esiintyvät ajankäytölliset ongelmat liittyivät yleensä suunnittelu- ja kehitystiimien työskentelytapojen eroavaisuuteen. Siinä missä kehitystiimi pyrkii saamaan uusia implementaatioita valmiiksi lyhyissä aikatauluissa, yrittää suunnittelutiimi pysyä perässä tekemällä vaatimusmäärittelyjä ja jatkuvaa käyttäjätutkimusta. Tämän todettiin olevan suunnittelutiimille lähes mahdotonta ajankäytöllisestä näkökulmasta, sillä koodaustyö aloitetaan jo sprinttien alussa, jolloin suunnittelutyötä ei olla vielä edes ehditty aloittaa. [Gulliksen et al., 2016; Chamberlain et al., 2006; Sy, 2007]

Miller [2005] ja Sy [2007] esittivät ratkaisuna tähän työskentelytavan, johon esimerkiksi Kuusinen et al. [2012] ovat viittanneet myöhemmin termillä ”one sprint ahead”. Kirjallisuuskatsaukseen valittiin tarkasteltavaksi Syn [2007] uudempi tutkimus Millerin [2005] sijaan, koska se oli relevantimpi tutkielmalle.



Kuva 9. Syn [2007] malli Agile UXD -sprinteille. [Kuusinen et al., 2012]

Kuvassa 8 havainnollistettu Syn [2007] työskentelymalli pyrkii ratkaisemaan etukäteissuunnitteluun tarvittavan ajan ensimmäistä sprinttiä edeltävällä 4-6 viikon

mittaisella vaiheella, johon viitataan termillä ”Cycle Zero”. Tässä vaiheessa suunnittelutiimin on tarkoitus suorittaa alustavaa käyttäjätutkimusta, jonka avulla saadaan muodustettua parempi kuva käyttäjien tarpeista ja sitä kautta vaatimusmäärittelystä. Tämän alustavan suunnitteluvaiheen jälkeen aloitetaan varsinainen ensimmäinen sprintti, jolloin kehitystiimi alkaa työstää koodia kerätyn tiedon perusteella. Samaan aikaan suunnittelutiimi alkaa luomaan designeja seuraavaa sprinttiä varten ja tekemään tutkimusta kolmatta sprinttiä varten. Toisen sprintin aikana suunnittelutiimi jatkaa samaan tapaan, mutta tämän lisäksi tiimi testaa kehitystiimin implementaatiota, jonka kehitystiimi sai valmiiksi ensimmäisen sprintin päätteeksi. Toisinsanoen työskentelytavan tarkoituksena on luoda olosuhteet, joissa suunnittelu- ja kehitystiimi pystyvät työskentelemään rinnakkaisesti siten, että suunnittelulle ja käytettävyydetutkimukselle saadaan varattua aikaa koko projektin elinkaaren ajan. [Sy, 2007]

Vaikka suunnittelulle saatiinkin varattua kyseisen toimintamallin avulla enemmän aikaa, ilmeni käytettävyydetutkimusten tekemisessä silti kiirettä. Tähän pyrittiin vastaamaan jakamalla suuret designit pienempiin palasiin (engl. *design chunks*), joita lähestyttiin ketterän kehityksen henkisesti iteratiivisesti ja inkrementaalisesti. Syn mukaan tällaisessa työskentelytavassa oli tärkeää pystyä priorisoimaan designin osat ennen niiden toteuttamista. Tällä tavalla pystyttiin välttymään turhalta työltä, kuten käyttämättömien designien tekemiseltä, jolloin saatiin tehostettua myös ajankäyttöä. Kaiken kaikkiaan työskentelytavan koettiin tehokkaaksi ja sen avulla saatiin parannettua projektin työvaiheiden priorisointia ja tiimien välistä kommunikaatiota päivittäisillä keskusteluilla. Designien pilkkomisen vaarana nähtiin kuitenkin se, että suunnittelijoiden on vaikeampi hahmottaa lopullisen designin kokonaiskuvaa. [Sy, 2007]

Adikari et al., [2009] esittelivät myöhemmin hyvin samankaltaisen työskentelytavan, johon viitataan termillä Little Design Up Front (LDUF). Tätä termiä ovat käyttäneet omissa tutkimuksissaan muun muassa Brhel et al. [2015] ja Da Silva et al. [2012].

Brhel et al. [2015] toteavat kirjallisuuskatsauksessaan, että 50.6% tutkituista julkaisuista vahvistivat etukäteissuunnittelun tärkeyden ohjelmistotuotannossa. Vaikka perusteellisen etukäteistutkimuksen todetaan olevan ketterien menetelmien periaatteiden vastaista, LDUF:n kaltaisen maltillisen etukäteissuunnittelun myönnetään olevan tarpeellista onnistuneessa käyttäjäkeskeisessä suunnittelussa. Suoritettavan etukäteissuunnittelun



määrästä ei ole yksimielistä mielipidettä ja Brhel et al. [2015] suosittelivatkin, että aiheesta pitäisi suorittaa lisää empiiristä tutkimusta. LDUF:n todetaan olevan erityisen tarpeellinen metodi yhdenmukaisen designin saavuttamiseksi ja se mahdollistaa oikeanlaisen designin konseptoinnin heti projektin alusta lähtien. Alati muuttuvien käyttöliittymädesignien todetaan myös hankaloittavan käyttäjätutkimusten tekemistä, koska käyttäjien tulisi tällaisissa tapauksissa opetella järjestelmän toiminta joka kerta uudestaan. [Brhel et al., 2015]

Tätä löydöstä tukee myös Adikari et al. [2009] suorittama tutkimus, jossa verrattiin kahdella eri tavalla työskennellyttä ketterää projektitiimiä. Toinen tiimeistä toimi perinteisen ketterän ohjelmistokehityksen toimintatapojen mukaisesti ja toinen tiimi puolestaan hyödynsi työssään LDUF:n mukaista työskentelytapaa, jossa kaksi UX-suunnittelijaa saivat jatkuvasti työskennellä kehitystiimin kanssa. Projektien lopussa suoritetun käytettävyyssyytyväisyyskyselyn mukaan LDUF-menetelmällä työstetty järjestelmä oli helpommin opittava, sitä oli helpompi käyttää ja sen käyttö vaati vähemmän ylimääräistä tukea. [Adikari et al., 2009]

Chamberlain, Sharp ja Maiden [2006] tutkivat myös usean eri tavalla työskentelevän projektitiimin toimintatapoja suuressa organisaatiossa. Tutkimuksen perusteella todettiin, että LDUF:n tapaista metodia harjoittaneella tiimillä esiintyi kolmesta tutkitusta tiimistä vähiten ongelmia kehittäjiä ja suunnittelijoiden välisessä kommunikaatiossa. [Chamberlain, Sharp ja Maiden, 2006]

Myös Budwig et al. [2009] tukevat tutkimuksensa pohjalta työskentelytapaa, jossa suunnittelutiimille annetaan mahdollisuus työskennellä 1-2 sprinttiä kehitystiimiä edellä. Tämän todettiin vähentävän yleistä sekavuutta projektissa ja parantavan myös suunnittelijoiden tehokkuutta ja tyytyväisyyttä. [Budwig et al., 2009]

Vaikka etukäteissuunnittelu voidaan nähdä ketterän kehityksen periaatteiden vastaisena, voidaan LDUF:n kaltaisella mallilla vähentää niin ajankäytöllisiä kuin myös kommunikaatioon liittyviä haasteita. Sen todettiin lisäksi helpottavan yhtenäisempien designien luomista ja parantavan projektitiimin tehokkuutta.

### 5.2.2 Viestinnän kehittäminen

Kohdassa 5.1 kuvattiin, että ajankäytöllisten haasteiden lisäksi ongelmia Agile UXD -työskentelyssä aiheutti myös sidosryhmien puutteellinen ymmärrys ja kommunikaatio. Tämän todettiin johtavan muun muassa roolien ja päätöksenteon epäselvyyteen, yleisiin väärinkäsityksiin sekä ajantuhlaukseen. Todettiin myös, että viestinnälliset haasteet korreloivat suoraan muun muassa ajankäytön haasteiden kanssa. Kuten kuvasta 8 voidaan todeta, korostuu viestinnällisten haasteiden merkitys myös sillä, että niihin tarjottiin kirjallisuuskatsauksen tutkimuksissa eniten kehitysmahdollisuuksia etukäteissuunnittelun lisäksi.

Gulliksen et al. [2016] ehdottavat tutkimuksessaan, että käytettävyyden ja käyttökokemuksen merkitys ja siihen kuuluvat vastuut on selvitettävä kaikille projektin eri rooleille. Tämä pitää sisällään niin tiimin jäsenet, Scrum Masterit kuin myös tuotteenomistajat. Näin voidaan varmistua siitä, että kaikki sidosryhmät ymmärtävät suunnitteluun kuuluvien roolien merkityksen ja työtehtävät. Käytettävyydelle ja käyttökokemukselle täytyy myös asettaa selvät tavoitteet jo projektin aikaisessa vaiheissa. Näitä tavoitteita tulee referoida projektin edetessä ja tarpeen tullen niihin tulee myös tehdä muutoksia. [Gulliksen et al., 2016]

Tavoitteiden tulee olla mitattavissa ja niiden toteutumista tulee evaluoida säännöllisesti käyttäjien kanssa. Gulliksen et al. [2016] toteavatkin, että suunnittelutiimin jäsenien tulisi kommunikoida käyttäjien kanssa säännöllisesti kasvotusten, vähintään kerran sprintin aikana. Käyttäjärviointeja puolestaan tulisi järjestää vähintään joka toisen sprintin välein, jotta voidaan varmistua tuotteen vaatimusten täyttymisestä sekä käyttäjien tyytyväisyydestä. Arviointien tuloksia täytyy myös käydä läpi sisäisesti projektitiimin kesken, jotta kaikki ymmärtävät tulokset ja niiden merkityksen. Tutkimuksessa todetaan myös, että sprinttien retrospektiivisessä palaverissa olisi hyvä keskustella käytettävyyden ja käyttökokemuksen parantamisesta. [Gulliksen et al., 2016]

Brhel et al. [2015] toteavat kirjallisuuskatsauksessaan, että artefaktit ovat keskeinen tapa kommunikoida niin ketterässä ohjelmistotuotannossa kuin myös käyttäjäkeskeisen suunnittelun näkökulmasta. Siinä missä ketterä kehitys käyttää yleensä toimivia prototyyppejä, käyttää käyttäjäkeskeinen suunnittelu laajasti erilaista dokumentaatiota, kuten persoonia, designien kommunikoimiseen. [Brhel et al., 2015]

Myös Da Silva et al. [2012] ehdottavat ketterän kehityksen ja käyttäjäkeskeisen suunnittelun yhdistämiseksi viitekehystä, jossa käytettävyyssartefakteja käytetään kommunikaation ja designien välittämisen välineenä. Artefaktien käytön todetaan parantavan käytettävyyttä ja samalla vaikuttavan mahdollisimman minimaalisesti normaaleihin ketterän ohjelmistokehityksen käytäntöihin. Da Silva et al. [2012] mainitsevat erilaisista artefakteista prototyypit, ominaisuus-kortit (engl. *Feature Cards*), design-kortit (engl. *Design Cards*) sekä vika-kortit (engl. *Issue Cards*). Kyseisiä kortteja voidaan käyttää kanban-henkisesti käyttäjäkokemukselle suunnatulla visualisoidulla taululla, joka on kaikkien sidosryhmien nähtävillä. Tutkimuksessa painotetaan kuitenkin, että jokaisen yrityksen tulisi itse päättää toiminnalleen sopivat artefaktit. [Da Silva et al., 2012]

Da Silva et al. [2012] toteavat myös, että suunnittelutiimin tulisi työskennellä läheisesti kehitystiimin kanssa ja arvioida parhaansa mukaan sprinteissä toteutettuja implementaatioita. Koska meneillään olevan sprintin implementaatiota saattaa olla vaikea arvioida ennen sen valmistumista, voidaan arviointi suorittaa myös seuraavassa sprintissä. [Da Silva et al., 2012]

Budwig et al. [2009] suorittamassa tutkimuksessa UX-tiimillä oli ongelmia ymmärtää liiketoimintayksilöltä saatavia vaatimusmäärittelyjä ja etenemissuunnitelmia (engl. *roadmap*). Myös ketterän kehityksen nopeus aiheutti sekavuutta UX-tiimin prioriteetteihin. UX-tiimille päätettiin tästä syystä luoda oma scrumtiimi, jolla oli oma tuotteenomistaja ja oma tehtävälista. UX-tuotteenomistajan tarkoitus oli toimia UX-tiimin edustajana muiden yksikköjen ja tiimien välillä ja koordinoida näin UX-tiimin resursseja ja artefakteja yhdessä kehitystiimin kanssa. Idea herätti aluksi ristiriitaisia tunteita yrityksessä, mutta käytäntö johti lopulta kokonaisempaan ja tarkempaan UX-suunnitteluun ja auttoi UX-tiimiä keskittymään omaan työhönsä vaatimusmäärittelyn jatkuvan selvittelyn sijaan. [Budwig et al., 2009]

Kuten tutkimuksista voidaan todeta, vaatii viestinnän kehittäminen Agile UXD -projekteissa tiimin jäseniltä ensisijaisesti ymmärrystä toistensa rooleista ja tehtävistä. Tämä pätee myös asiakkaiden ja käyttäjien kanssa käytävään kommunikaatioon, sillä myös näiden osapuolien on hyvä ymmärtää esimerkiksi UX-suunnittelun merkitys projektin onnistumisen kannalta. Toisaalta myös kehitys- ja suunnittelutiimien on

ymmärrettävä yhtä lailla käyttäjien tarpeet, minkä takia designien ja implementaatioiden jatkuva evaluointi käyttäjien kanssa on äärimmäisen tärkeää.

## 6. Keskustelu

Analysoitujen tutkimusten perusteella Agile UXD vaikuttaa konseptilta, jolla on potentiaalia tuoda lisäarvoa perinteiseen ketterään ohjelmistotuotantoon oikeanlaisissa olosuhteissa. Kuten luvussa 5 kerrottiin, sisältyy ketterien menetelmien ja käyttäjäkeskeisen suunnittelun sulavaan yhdistämiseen merkittäviä haasteita, jotka johtuvat pitkälti niille ominaisten toimintatapojen keskinäisestä eroavaisuudesta. Tästä huolimatta kyseisten kehitystapojen integraatio on mahdollinen ja sillä voidaan saavuttaa hyviä lopputuloksia, kuten esimerkiksi Syn [2007] one step ahead -mallilla ja Adikarin et al. [2009] Little Design Up Front -mallilla. Tutkielman perusteella integraation merkittävimpiä hyötyinä voidaan nähdä esimerkiksi käytettävyyden ja käyttökokemuksen paraneminen, niiden jatkuva evaluointi kehitystyön ohessa sekä sidosryhmien välisen kommunikaation tehostuminen.

Integraation toimivuus ei ole itsestäänselvyys ja se saattaa vaatia uusien työskentelytapojen omaksumista yrityksen usealta eri taholta ja yksiköltä. Koska Agile UXD:n haasteet liittyvät usein kommunikaatioon, yleisiin epäselvyyksiin sekä ajankäytöllisiin haasteisiin, tulisikin huomiota kiinnittää potentiaalisesti enemmän niiden tahojen toimintaan, jotka ovat vastuussa tiimien koordinoinnista ja resurssien ohjaamisesta. Kuten Kuusinen ja Väänänen-Vainio-Mattila [2012] totesivat tutkimuksessaan, saattaa myynnin ja hallinnon ymmärtämättömyys käyttökokemuksesta heijastua negatiivisesti koko ohjelmistoprosessiin, mikäli esimerkiksi UX-suunnittelun merkitystä ei osata selventää asiakkaille jo palvelun myyntivaiheessa. Tämä voi aiheuttaa sen, ettei suunnittelutyölle ohjata tarpeeksi resursseja, mikä puolestaan näkyy suunnittelutiimin kiireenä ja sitä kautta heikompana työnlaatuna.

Ehdotetaan, että empiiristä lisätutkimusta Agile UXD:sta tulisi tehdä suunnittelu- ja kehitystiimien lisäksi muidenkin sidosryhmien näkökulmasta. Näitä sidosryhmiä voivat olla esimerkiksi hallinto ja myynti, joiden toiminnalla on suora vaikutus projektitiimien toimintaan [Kuusinen ja Väänänen-Vainio-Mattila, 2012]. Lisäksi ehdotetaan, että Agile UXD:n prosesseja tulisi tutkia myös enemmän asiakkaiden ja käyttäjien näkökulmasta, koska nämä kuuluvat niin ketterien menetelmien kuin myös käyttäjäkeskeisen suunnittelun periaatteiden mukaisesti tiiviiksi osaksi kehitysprosessia.

Brhel et al. [2015] totesivat tutkimuksessaan, että muiden tieteenalojen, kuten organisaatiotieteiden ja sosiologian näkökulmista saattaisi olla hyötyä parhaiden Agile UXD -menetelmien löytämisessä. Da Silva et al. [2018] totesivat puolestaan, että kaupalliset työkalut suunnittelu- ja kehitystyön vaatimien teknologioiden ja artefaktien integroimiseen ovat edelleen puutteelliset ja niiden kehitykseen suunnattua tutkimustyötä tulisi lisätä. Kaiken kaikkiaan vaikuttaisi siltä, että Agile UXD:n vaatimat metodit ja prosessit tarvitsevat vielä merkittävästi lisää empiiristä tutkimusta niin muiden sidosryhmien kuin myös eri prosessien sulauttamisen näkökulmasta ennen kuin niiden tueksi voidaan luoda yleispäteviä ja kestäviä periaatteita sekä ohjeistuksia.

Tutkielman tekeminen sujui pääpiirteittäin sujuvasti ilman suurempia ongelmia. Prosessin alkuvaiheilla tutkielman tarkka fokus ja näkökulma olivat vielä hieman epäselviä, mikä hankaloitti myös tutkimusaineiston keruuta ja sen karsimista. Aineiston keruuta haittasivat myös muutaman julkaisun kohdalla tietokantaan kohdistuvien käyttöoikeuksien puuttuminen, minkä takia muutamaan julkaisuun piti etsiä suora linkki Googlen avulla. Julkaisukanavien JUFO-luokitus yritettiin aina tarkastaa mahdollisuuksien mukaan, mutta usean kanavan kohdalla tämä luokitus puuttui kokonaan.

## 7. Yhteenveto

Tutkielman kirjallisuuskatsauksen perusteella Agile UXD on haasteellinen, mutta toteutettavissa oleva integraatio, jossa on mahdollista yhdistää sekä ketterien menetelmien että käyttäjäkeskeisen suunnittelun parhaat ominaisuudet toimivaksi kokonaisuudeksi. Sen suurimmat haasteet ovat sidosryhmien välisessä puutteellisessa kommunikaatiossa, minkä nähtiin johtavan ymmärtämättömyyteen rooleista, tehtävistä sekä vastuualueista. Tällainen puutteellinen ymmärrys voi puolestaan johtaa ajankäytöllisiin haasteisiin etenkin suunnittelutyössä ja sitä kautta myös esimerkiksi käyttäjätutkimuksessa. Puutteellisesti suoritettun suunnittelutyön todettiin myös korreloivan suoraan tuotteen lopullisen laadun kanssa.

Ajankäytön haasteisiin ehdotettiin ratkaisuiksi etukäteissuunnittelua esimerkiksi Little Design Up Front -menetelmällä, jotta suunnittelu- ja kehitystiimien aikataulut saadaan paremmin synkronoitua keskenään. Sidosryhmien välisen viestinnän kehittämiseksi ehdotettiin tiiviimpää yhteistyötä kaikkien sidosryhmien välillä sekä vastuualueiden selventämistä kaikille osapuolille. Ideoiden ja designien kommunikoimiseen suositeltiin artefaktien, kuten erilaisten prototyyppien, käyttöä. Lisäksi käytettävyydelle pitää pystyä määrittelemään selkeät ja mitattavissa olevat tavoitteet, joita pitää pystyä evaluimaan säännöllisesti oikeiden käyttäjien kanssa.

## Lähteet

Adikari, S., Campbell, J and McDonald, C. (2009). Little Design Up-Front: a design science approach to integrating usability into Agile requirements engineering. *Human-Computer Interaction - HCII*, 2009, 549-558

Agile. (2015). 8 benefits of Agile software development. Segue Technologies. <https://www.seguetech.com/8-benefits-of-agile-software-development/> (haettu 28.4.2019).

Agile Project Management. What is Agile project management? Pivotal Tracker. <https://www.pivotaltracker.com/agile/what-is-agile-project-management> (haettu 28.4.2019).

Balaji, S. and Murugaiyan, M. (2012). Waterfall vs V-model vs Agile: a comparative study on SDLC. *International Journal of Information Technology and Business Management*, Vol. 2 No. 1, 26-30.

Beck, K. and Andres, C. (2004). *Extreme Programming explained: embrace change, 2nd edition*. Pearson Education, Inc.

Beck, K. ja Beedle, M. and van Bennekum, A. et al. (2001). Agile manifesto. <http://agilemanifesto.org>. (haettu 28.4.2019).

Blomquist, Å. and Arvola, M. (2002). Personas in action: ethnography in an interaction design team. *NordiCHI '02 Proceedings of the second Nordic conference on Human-computer interaction*, 197-200

Brandenburg, L. (2018). What is a use case? Bridging the Gap. <https://www.bridging-the-gap.com/what-is-a-use-case/> (haettu 28.4.2019).

Brhel, M., Meth, Maedche, A. and Werder, K. (2015). Exploring principles of user-centered agile software development: A literature review. *Information and Software Technology*, Vol. 61, 163-181.

Bruun, A., Larusdottir, M.K., Nielsen, L., Nielsen, P.A. and Persson, J.S. (2018). The role of UX professionals in Agile development: a case study from industry. *NordiCHI '18 Proceedings of the 10<sup>th</sup> Nordic Conference on Human-Computer Interaction*, 352-363.



Budwig, M., Jeong, S. ja Kelkar, K. (2009). When user experience met Agile: a case study. *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, 3075-3084.

Cao, J. (2016). What is a prototype: a guide to functional UX? UXPin.  
<https://www.uxpin.com/studio/blog/what-is-a-prototype-a-guide-to-functional-ux/>  
 (haettu 28.4.2019).

Chamberlain, S., Sharp, H. and Maiden, N. (2006). Towards a framework for integrating Agile development and user-centered design. *XP'06 Proceedings of the 7th international conference on Extreme Programming and Agile Processes in Software Engineering*, 143-153.

Cockburn, A. and Highsmith, J. (2001). Agile software development – the people factor. Computer Science Department, University of Southern California.

Da Silva, T.S., Silveira, M.S., Maurer, F. and Hellmann, T. (2012). User Experience Design and Agile development: from theory to practice. *Journal of Software Engineering and Applications*, 2012, 743-751.

Da Silva, T.S., Silveira, M.S., Maurer, F. and Silveira F.F. (2018). The evolution of Agile UXD. *Information and Software Technology*, Vol. 102, 1-5.

Düchting, M., Zimmermann, D. and Nebe, K. (2007). Incorporating user centered requirement engineering into Agile software development. *HCI 2007: Human-Computer Interaction. Interaction Design and Usability*, 58-67.

Extreme Programming. Extreme Programming. Agile Alliance.  
<https://www.agilealliance.org/glossary/xp> (haettu 28.4.2019).

Garcia, A., Da Silva, T.S. and Silveira, M.S. (2017). Artifacts for Agile user-centered design: a systematic mapping. *Proceedings of the 50th Hawaii International Conference on System Sciences*, 5859-5868.

Gulliksen, J., Boivie, I., Persson, J., Hektor, A. and Herulf, L. (2014). Making a difference – a survey of the usability profession in Sweden. *NordiCHI '04 Proceedings of the third Nordic conference on Human-computer interaction*, 207-215.

Gulliksen, J., Larusdottir, M. and Cajander, Å. (2016). A license to kill – improving UCSD in agile development. *Journal of Systems and Software*, Vol. 123, 214-222.

- ISO 9241-11. (1998). ISO9241 Ergonomic, Part 11: Guidance on usability. International Organisation for Standardisation.
- ISO 2941-210. (2009). ISO 2941-210 Ergonomics of human system interaction – Part 210: Human-centered design for interactive systems. International Organisation for Standardisation.
- Jeremiah, J. (2017). Survey: Is Agile the new norm? Tech Beacon. <https://techbeacon.com/app-dev-testing/survey-agile-new-norm> (haettu 28.4.2019).
- Kanban. (2018). What is Kanban? Digite. <https://www.digite.com/kanban/what-is-kanban/> (haettu 28.4.2019).
- Kanban. (2016). Agile vs Scrum vs Waterfall vs Kanban. Smartsheet. <https://www.smartsheet.com/agile-vs-scrum-vs-waterfall-vs-kanban> (haettu 28.4.2019).
- Kitchenham, B. (2004). Procedures for performing systematic reviews. Software Engineering Group, Department of Computer Science, Keele University.
- Kukhnavets, P. (2018). Disadvantages and advantages of Extreme Programming. Hygger. <https://hygger.io/blog/disadvantages-and-advantages-of-extreme-programming/> (haettu 28.4.2019).
- Kuusinen, K., Mikkonen, T. and Pakarinen, S. (2012). Agile user experience development in a large software organization: good expertise but limited impact. *International Conference on Human-Centered Software Engineering, 2012*, 94-111.
- Kuusinen, K. and Väänänen-Vainio-Mattila, K. (2012). How to make agile UX work more efficient: management and sales perspectives. *NordiCHI '12 Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, 139-148.
- Lakeworks. (2009). The Scrum process. Wikipedia. [https://en.wikipedia.org/wiki/Scrum\\_\(software\\_development\)#/media/File:Scrum\\_process.svg](https://en.wikipedia.org/wiki/Scrum_(software_development)#/media/File:Scrum_process.svg) (haettu 28.4.2019).
- Losada, B., Urretavizcaya, M. and Fernández-Castro, I. (2012). A guide to Agile development of interactive software with a “user objectives” -driven methodology. *Science of Computer Programming, Vol. 78, Issue 11*, 2268-2281.

- Lotz, M. (2018). Waterfall vs. Agile: which is the right development methodology for your project? Segue Technologies. <https://www.seguetech.com/waterfall-vs-agile-methodology/> (haettu 28.4.2019).
- Maguire, M. (2001). Context of use within usability activities. *Int. J. Human-Computer Studies*, Vol. 55, Issue 4, 453-483.
- Matthews, T., Judge, T. and Whittaker, S. (2012). How do designers and user experience professionals actually perceive and use personas? *CHI '12 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1219-1228.
- McInerney, P. and Maurer, F. (2005). UCD in Agile projects: dream team or odd couple? *Interactions*, Vol. 12, Issue 6, 19-23.
- Miller, L. (2005). Case study of customer input for a successful product. *ADC '05 Proceedings of the Agile Development Conference*, 225-234.
- Nadikattu, S. (2016). Integrating User Experience (UX) development with Agile software development practices. Blekinge Institute of Technology, Faculty of Computing, Department of Software Engineering.
- Nedeltcheva, G. and Shoikova, E. (2017). Coupling design thinking, user experience design and Agile: towards cooperation framework. *BDIOT2017: Proceedings of the International Conference on Big Data and Internet of Thing*, 225-229.
- Nielsen, J. (1994). Ten usability heuristics for user interface design. Nielsen Norman Group. <https://www.nngroup.com/articles/ten-usability-heuristics/> (haettu 28.4.2019).
- Pereira, J.C. and Russo, Rosaria de F.S.M. (2018). Design thinking integrated in Agile software development: a systematic literature review. *Procedia Computer Science*, Vol. 138, 775-782.
- Pernice, K. (2016). UX prototypes: low fidelity vs. high fidelity. Nielsen Norman Group. <https://www.nngroup.com/articles/ux-prototype-hi-lo-fidelity/> (haettu 28.4.2019).
- Salah, D., Cairns, P. and Paige, R. (2015). Patterns for integrating Agile development processes and user-centered design. *EuroPLoP '15: Proceedings of the 20th European Conference on Pattern Languages of Programs*.

Schwaber, K. and Sutherland, J. (2016). The Scrum guide. Scrum.org.  
<https://www.scrum.org/resources/scrum-guide> (haettu 28.4.2019).

Sergeev, A. (2017). To Scrum or not to Scrum? Pros and cons to consider. Hygger.  
<https://hygger.io/blog/to-scrum-or-not-to-scrum-pros-and-cons-to-consider/> (haettu 28.4.2019).

Subbaiah, V. (2016). Why Agile is the best alternate methodology to waterfall? Asahi Technologies. <https://www.asahitechnologies.com/blog/why-agile-is-the-best-alternate-methodology-to-waterfall/> (haettu 28.4.2019).

Sy, D. (2007). Adapting usability investigations for Agile user-centered design. *Journal of Usability Studies*, Vol. 2, Issue 3, 112-132.

Takeuchi, H. and Nonaka, I. (1986). The new new product development game. Harvard Business School Publishing.

Use cases. Capturing requirements with use cases. University of Missouri Kansas City.  
<https://sce2.umkc.edu/BIT/burris/pl/usecasemodeling/> (haettu 28.4.2019).

Wells, D. (2009). Extreme Programming: A gentle introduction. Extreme Programming.  
<http://www.extremeprogramming.org> (haettu 28.4.2019).