Tampereen yliopisto

Aino Haikala

# KÄYTTÄJÄKOKEMUKSEEN VAIKUTTAVIEN RATKAISUJEN TEKEMINEN KETTERÄSSÄ PROJEKTISSA

– kehittäjän näkökulma

Informaatioteknologian ja viestinnän tiedekunta
Diplomityö
Maaliskuu 2019

# TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
**HAIKALA, AINO**: Käyttäjäkokemukseen vaikuttavien ratkaisujen tekeminen ketterässä projektissa – kehittäjän näkökulma
Diplomityö, 71 sivua, 4 liitesivua
Maaliskuu 2019
Pääaine: Pervasive systems
Tarkastaja: professori Hannu-Matti Järvinen, palvelumuotoilija Outi Kotala
Avainsanat: käyttäjäkokemus, ketterät menetelmät, käyttäjäkeskeiset menetelmät,

Käyttäjäkokemukseen (UX) vaikuttavat ratkaisut tehdään projekteissa siirtyen laajemmista kokonaisuuksista pienempiin yksityiskohtiin. Tämän työn tavoitteena oli ensinnäkin selvittää, kuinka käyttäjäkokemuksen suunnittelu on yhdistetty projekteissa ketteriin menetelmiin. Toisena tavoitteena oli selvittää, millaisia käyttäjäkokemukseen vaikuttavia ratkaisuja ohjelmistokehittäjät tekevät ja mikä on heidän kokemuksensa käyttäjäkokemukseen vaikuttavien ratkaisujen tekemisestä.

Työ koostuu teoriaosasta ja empiirisestä osasta. Teoriaosassa käydään läpi käyttäjäkokemukseen liittyvät peruskäsitteet ja tutustutaan ketteriin menetelmiin. Lisäksi tutustutaan siihen, millaisilla malleilla käyttäjäkeskeisen suunnittelun menetelmiä voidaan yhdistää ketteriin menetelmiin. Empiirisessä osassa toteutettiin haastattelututkimus, jossa haastateltiin henkilöitä seitsemästä eri ohjelmistoprojektista. Saatuja tuloksia verrattiin kirjallisuudesta löydettyihin tuloksiin.

Tutkimuksessa havaittiin, että kirjallisuudessa esiintyneitä malleja ei käytetty projekteissa. Lisäksi eri projektien tavat yhdistää käyttäjäkokemuksen suunnittelu ketteriin menetelmiin erosivat toisistaan. Kehittäjien havaittiin tekevän projekteissa käyttäjäkokemukseen vaikuttavia ratkaisuja, joiden laajuus ja laatu riippuivat projektin tilanteesta ja kehittäjän mielenkiinnonkohteista. Kehittäjien havaittiin tekevän itsenäisesti erityisesti pienempiä käyttöliittymäratkaisuja, toimivan yhtenä UX-suunnittelijan työn laatua arvioivana portaana ja kertovan teknisistä rajoitteista, jotka vaikuttivat UX-ratkaisujen tekemiseen.

# ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
HAIKALA AINO: Making Decisions Affecting User Experience in Agile Projects – Developer's Point of View
Master of Science Thesis, 71 pages, 4 Appendix pages
Joulukuu 2018
Major: Pervasive systems
Examiner: Professor Hannu-Matti Järvinen and Outi Kotala
Keywords: user-experience, agile, user-centered methods

Decisions affecting user experience (UX) are made top down in projects. The aim of this study was to research how UX design and agile methods are combined in projects. The second objective was to inspect the usability affecting decisions made by software developers and investigate the experience of making those decisions.

The fundamental concepts of user experience and agile methods are presented in the theorical part of this study as well as the models designed for integrating the human centered design into agile methods. The empirical part of this study consist of interviews of employees in seven software projects, the analysis of results and comparison of the results to studies published before.

It was found that projects did not use the models published in literature. Furthermore, the way of combining user experience design and agile methods differed between projects. Software developers were found to make independent user experience affectiong decisions. They make are especially smaller user interface solutions, evaluate the work of UX designers and share knowledge about technical constraints limiting the UX decisions.

# ALKUSANAT

Olen saanut työn tekemiseen paljon apua eri ihmisiltä. Olen onnekas, kun olen saanut olla tekemisissä kanssanne. Olen saanut teiltä uusia näkökulmia, tietoa, ideoita ja myös hellävaraista patistamista. Outi Kotala on kärsivällisesti ohjannut työtä ja avannut silmiäni maailmalle oman kuplani ulkopuolella. Kiitos myös arvokkaista neuvoista, joita olen saanut työn tarkastajalta, professori Hannu-Matti Järviseltä. Erityiskiitos teille, jotka suostuitte haastateltaviksi. Kiitokset myös Kirsi Mäkiselle, Jarmo Korhoselle, Suvi Leanderille ja Viljakaisa Aaltoselle.

Lisäksi tahdon kiittää Juho-Mikko Heinosta, joka on uhrannut monta iltaa erilaisten pohdintojen testikenttänä. Kiitokset myös Iirolle ja Ullalle sekä teille muille rakkaille, jotka teette tästä merkityksellistä.

# SISÄLLYS

# TERMIT JA NIIDEN MÄÄRITELMÄT

| | |
|---|---|
| Kanban-menetelmä | Kanban-menetelmä on inkrementaalinen ja evolutionäärinen Lean-filosofiaa tukeva kehitysmenetelmä, jossa kanban-taulua voidaan käyttää työkaluna (Al-Baik ja Miller, 2015; Ahmad *ym.*, 2018). |
| Ketterät menetelmät | Ketterän julistuksen periaatteita noudattavat mallit, joilla ohjelmistoja tehdään iteratiivisesti kehityssykleissä. |
| Käytettävyys | Vaikuttavuus, tehokkuus ja tyytyväisyys, joilla palvelulle määritellyt käyttäjät saavuttavat määritellyt tavoitteet tietyssä ympäristössä. |
| Käyttäjäkeskeinen suunnittelu (UCD = user-centered design) | Vuorovaikutteisten systeemien kehittämisen lähestymistapa, jonka tavoitteena on tehdä järjestelmistä käytettäviä ja hyödyllisiä keskittymällä käyttäjiin, heidän tarpeisiinsa ja vaatimuksiinsa (ISO 9241:210). |
| Käyttäjäkokemus | Henkilön käsitykset ja reaktiot, jotka ovat seurausta tuotteen, järjestelmän tai palvelun käytöstä tai odotetusta käytöstä |
| Lean | Tuotantofilosofia, jossa painotetaan erityisesti arvon luomista asiakkaalle ja hukan välttämistä. |
| Lean UX | Lean-periaatteista ja systeemiajattelusta vaikutteita saanut menetelmä UX-suunnittelun ja ketterän kehityksen yhdistämiseen. Painottaa tiimin yhteistyötä ja kehityksen iteratiivisuutta. |
| Palvelumuotoilu | Käyttäjän mukaan ottava suunnitteluprosessi, jonka tavoitteena on luoda mahdollisimman hyvä palvelukokemus optimoimalla palveluprosessi, työtavat, tilat ja vuorovaikutus sekä poistamalla palvelua häiritsevät asiat. |
| UX-suunnittelu l. käyttäjäkokemuksen suunnittelu | Suunnittelutoimet, jolla pyritään käyttäjän mahdollisimman hyvään kokonaiskokemukseen käytettäessä tuotetta, ennen käyttöä ja sen jälkeen. |
| Yksi sprintti edellä -malli | Malli, jossa yhdistetään käyttäjäkeskeiset menetelmät ketterään kehitykseen. Käytössä on normaalin kehityssprintin rinnalla kulkeva UX-sprintti, jossa suunnitellaan seuraavassa toteutussprintissä toteutettavia asioita. |

# 1 JOHDANTO

Tämä työ lähti liikkeelle sellaisen projektin seuraamisesta, jossa ei ollut UX-suunnittelijaa mukana. On ollut kiehtovaa seurata, miten sovellus on muuttunut vuosien mittaan. Sovellukseen on tullut uusia ominaisuuksia ja jokainen muutos on vaikuttanut käyttäjien kokemukseen sovelluksesta jollain tavalla. Välillä käyttäjien kokemusta on pohdittu muutoksia tehtäessä. Välillä muutokset on tehty nimenomaan sen takia, että asiat halutaan tehdä käyttäjälle helpommiksi tai selkeämmiksi. Joskus taas käyttäjäkokemuksen muuttumista ei ole sen suuremmin mietitty. Kun kukaan yksittäinen henkilö ei ollut vastuussa käyttäjäkokemuksen muuttumisesta, muutokset ovat jääneet sekä asiakkaan että yksittäisten kehittäjien osaamisen varaan.

Tätä opinnäytetyötä tehdessäni opin, että kyseinen projekti oli erilainen verrattuna moneen muuhun projekteihin. Kuitenkin se, mikä yhdistää sen kaikkiin muihin projekteihin oli, että nykyiset ohjelmistokehitysprojektit ovat ennen kaikkea tiimityötä. Ohjelmistot kehitetään tiimeissä, joissa on ihmisiä erilaisilla taidoilla ja toimenkuvilla. Koska kehitettävät ohjelmistot ja niitä kehittävät ihmiset vaihtuvat, ei ole kahta samanlaista ohjelmistokehitysprojektia. Lisäksi asiakkaat ovat erilaisia, puhumattakaan asiakkaan ja toimittajan organisaatiosta ja toimintakulttuurista. Vielä monimutkaisempaa on, jos asiakkaana onkin konsortio ja toimittajia on projektissa useita.

Kuitenkin niin kauan kuin ohjelmistokehitystä on ollut, on ollut tarve yrittää löytää käytäntöjä hallitsemaan sitä prosessia, joilla asiakkaan tarpeet saadaan muutettua ohjelmistoksi. Myöhemmin, kun hyvän käyttäjäkokemuksen arvo on alettu ymmärtää ja kun käyttäjäkeskeisen suunnittelun menetelmät ovat yleistyneet, on alettu etsiä hyviä tapoja yhdistää käyttäjäkeskeisen suunnittelun menetelmät ohjelmistokehityksen prosessimalleihin. Yleensä yhdistettäessä malleja toisiinsa on havaittu, että 1) käyttäjäkokemusta tulee suunnitella kokonaisuuden tasolla jo ennen koko toteutuksen aloitusta ja 2) yksittäiset ominaisuudet suunnitellaan jonkin verran ennen niiden koodaamista, usein yksi tai kaksi sprinttiä etukäteen. Käytännössä kuitenkin todettiin, että vaikka yksittäiset ominaisuudet pyrittäisiin suunnittelemaan etukäteen, ominaisuuksia suunnitellaan myös vasta silloin, kun on pakko.

Työn tekemisen antia on ollut ymmärtää paremmin kuinka laaja taitojen kirjo ohjelmistoprojektissa tarvitaan. Ohjelmistokehittäjien lisäksi tarvitaan, mainitakseni vain muutamia, palvelumuotoilijoita, projektipäällikköjä ja graafikkoja. Kaikilla näistä ihmisistä on oma laaja osaamisalueensa ja tapansa ajatella asioita.

## 1.1 Tutkimuskysymykset

Tässä diplomityössä on pyritty selvittämään, että miten käyttäjäkokemukseen vaikuttavat ratkaisut tehdään ketterässä projektissa. Ajallisesti tutkimuksessa keskitytään nimenomaan siihen vaiheeseen, kun ominaisuuksia ollaan toteuttamassa. Esiselvitysvaihe jätettiin tutkimuksen fokuksen ulkopuolelle.

Pääpaino tutkimuksessa on erityisesti kehittäjien tekemissä käyttäjäkokemukseen vaikuttavissa ratkaisuissa sekä kehittäjien ja UX-suunnittelijoiden vuorovaikutuksessa. Työssä pyrittiin vastaamaan seuraaviin tutkimuskysymyksiin:

1. Miten käyttäjäkokemuksen suunnittelu on liitetty mukaan ketterien projektien käytäntöihin?
2. Ketkä projekteissa tekevät käyttäjäkokemukseen vaikuttavia päätöksiä?
3. Millaisia käyttäjäkokemukseen vaikuttavia päätöksiä kehittäjät tekevät?
4. Miten kehittäjät kokevat käyttäjäkokemukseen vaikuttavien päätösten tekemisen projektissa?

Tavoitteena ei ole arvottaa parhaita ratkaisuja tai antaa neuvoja siitä, kuinka ohjelmistokehitys pitäisi yhdistää ihmiskeskeiseen suunnitteluun. Yhtä ainoaa mallia olisikin vaikea luoda. Työn aikana tuli hyvin selkeästi esille, että eri kokoiset ja eri vaiheessa olevat projektit vaativat erilaisia toimintotapoja. Samoin erilaiset ihmiset työskentelevät eri tavalla. Työn pyrkimyksenä on dokumentoida olemassa olevia käytäntöjä ja yrittää tehdä läpinäkyvämmäksi se, millä tavalla sovelluksen käyttäjäkokemus kehityksen aikana syntyy.

## 1.2 Tutkimusprosessi

Työhön toteutettiin sekä teoriaosa että empiirinen osa. Teoriaosassa käydään läpi työhön liittyviä peruskäsitteitä ja esitellään teoreettinen pohja käyttäjäkeskeisen suunnittelun ja ohjelmistosuunnittelun yhdistämiseen. Teoriaosan tavoitteena on pyrkiä luomaan pohja käyttäjäkokemuksen suunnittelun ja erilaisten projektikäytäntöjen ymmärtämiselle. Empiirisessä tutkimuksessa haastateltiin projektitiimejä ja yksittäisiä henkilöitä siitä, kuinka käyttäjäkokemukseen vaikuttavat ratkaisut projektissa käytännössä tehdään ja siitä, kuinka käyttäjäkeskeinen suunnittelu oli sovitettu projektimalliin.

Työn tekeminen aloitettiin tutkimalla aiheesta kirjoitettuja valmiita tutkimuksia ja muokkaamalla tutkimuksen aihetta niiden perusteella. Tutkimuskysymysten asettamisen jälkeen kirjoitettiin työn teoriaosuus. Tässä vaiheessa haastateltiin myös palvelumuotoilijaa siitä, miten palvelumuotoilijan työ ajoittuu ohjelmistokehitysprojektissa ja mitä siihen kuuluu. Näin pyrittiin saamaan parempi kokonaiskäsitys aihepiiristä. Kun tarpeeksi hyvä yleiskäsitys aiheesta oli hankittu, haastateltiin projektitiimejä ryhmähaastatteluissa

ja yksittäisiä henkilöitä. Saaduista tuloksista tehtiin yhteenveto ja saatuja tuloksia verrattiin kirjallisuudessa saatuihin tuloksiin.

## 1.3 Diplomityön rakenne

Diplomityö koostuu teoreettisesta osasta ja empiirisestä osasta. Teoreettinen osuus kattaa luvut 2 – 4. Luvussa kaksi kerrotaan siitä, mitä käyttäjäkokemus on ja siitä, miten käyttäjäkokemusta suunnitellaan. Luvussa kolme esitellään ohjelmistojen kehityksessä käytettyjä prosessimalleja. Painotus on erityisesti ketterissä, iteratiivisissa menetelmissä. Ainoa suunnittelupohjainen malli, joka esitellään, on vesiputousmalli. Vesiputousmalli haluttiin ottaa mukaan, koska se oli pitkään käytännössä standardin asemassa ja sillä on edelleen vahva asema etenkin tilaajien puolella. Luvussa neljä käydään läpi kirjallisuudessa aiemmin esitettyjä tapoja yhdistää käyttäjäkokemuksen suunnittelu ketteriin menetelmiin.
Empiirinen osa kattaa luvut 5 – 8. Luvussa viisi käydään läpi tutkimusmenetelmiä. Luvussa kuusi esitellään tutkimussuunnitelma. Luku seitsemän kattaa tulokset. Luvussa kahdeksan analysoidaan tutkimussuunnitelmaa. Luvussa yhdeksän pohditaan saatujen tuloksien suhdetta alalla julkaistuihin tutkimuksiin. Viimeinen luku sisältää loppusanat.

# 2 KÄYTTÄJÄKOKEMUS

Viesti puhelimella kavereille, pesukoneen ohjelman valinta, reseptin etsiminen verkkopalvelusta, aktiivisuusranneke mittaamassa hyvinvointia — tietokoneohjelmat ovat läsnä kaikkialla. Vuorovaikutuksemme tietokoneiden kanssa on niin arkista, ettemme välttämättä edes ymmärrä käyttävämme niitä. Muutama vuosikymmen sitten tietokoneet olivat harvojen koulutettujen spesialistien rajattuun tehtävään käyttämiä laitteita. Nykyään ne ovat muuttuneet suuren yleisön yleiskoneiksi, joita käytetään työkaluina niin töissä kuin vapaa-ajalla. Koska henkilön käyttämien tietokonesovellusten määrä ja tarjolla olevien sovellusten valikoima on lisääntynyt räjähdysmäisesti, ei yleensä voida olettaa, että käyttäjä näkee vaivaa opetellakseen käyttämään tiettyä sovellusta. Puhumattakaan siitä, että käyttäjä saisi koulutuksen kaikkiin käyttämiinsä sovelluksiin.

Ihmisen ja tietokoneen välisen vuorovaikutuksen lisääntyminen ja arkipäiväistyminen on tehnyt hyvästä suunnittelusta ja suunnittelumenetelmien käytöstä yhä tärkeämpää. Sinänsä ihmisen ja tietokoneen välisen vuorovaikutuksen tutkiminen ei ole uusi asia. Ihmisen ja tietokoneen välisen vuorovaikutuksen tutkimisen juuret juontavat tehdastyön tehokkuuden parantamisen tutkimiseen 1900-luvun alussa ja sen tutkimisen tarve alkoi kasvaa tietokoneiden siirtyessä 80-luvulla yhä enemmän suuren, kouluttamattoman yleisön käytettäväksi (Grudin, 2011).

ISO-standardeihin käyttäjäkokemus pääsi vuonna 2010. ISO-9241-210 standardin mukaan käyttäjäkokemus (*UX = user experience*) sisältää **henkilön käsitykset ja reaktiot, jotka ovat seurausta tuotteen, järjestelmän tai palvelun käytöstä tai odotetusta käytöstä** (SFS-EN ISO 9241-210, 2010). Käyttäjäkokemus koostuu käyttäjän tunteista, uskomuksista, mieltymyksistä, fyysisistä ja psyykkisistä vasteista, käyttäytymisestä ja aikaansaannoksista, jotka syntyvät ennen käyttöä, käytön aikana ja käytön jälkeen. Se, millaisen kokemuksen käyttäjä tuotteesta saa, riippuu sekä käyttäjästä, tuotteesta että käyttötilanteesta. Käyttäjäkokemus on siis hetkellinen ja henkilökohtainen, mikä asettaa haasteita käyttäjäkokemuksen suunnittelulle. Nykyisin käyttäjäkokemuksen suunnittelun tärkeys usein ymmärretään ohjelmistoja tuottavissa yrityksissä. Ymmärrystä alkaa olla myös yhä enemmän asiakkaiden puolella.

Käyttäjäkokemuksen tutkimus rajoittuu harmillisen usein siihen, millainen käyttäjäkokemus sovelluksella on silloin, kun käyttäjä aloittaa tuotteen käytön. Käyttäjäkokemus kestää kuitenkin koko tuotteen käyttämisen ajan ja voi muuttua käytön aikana alkuinnostuksesta tuskastumiseen tai päinvastoin. Toisaalta nykyisin myös tuotteista julkaistaan uusia versioita paljon useammin kuin ennen. Tyypillisesti julkaisujen välejä ei lasketa enää vuosissa vaan viikoissa. Käytettäessä jatkuvaa julkaisua julkaisusykli voi olla

vielä tiheämpi. Jos ajatellaan käyttäjän kokemukseen vaikuttavia tekijöitä, sekä käyttäjä, tuote että käyttötilanne muuttuvat ajan kuluessa. Eräällä tavalla tähdätään siis liikkuvaan maaliin.

## 2.1 Hassenzahlin malli

Hassenzahl (2003) on luonut mallin käyttäjäkokemuksesta. Hassenzalin mukaan tuotteella on tietyt ominaisuudet, jotka koostuvat sekä toiminnoista, sisällöstä, vuorovaikutustyylistä että esitystavasta. Tuotteen suunnittelijat pyrkivät välittämään valitsemillaan ominaisuuksilla tietynlaisen kuvan käyttäjälle. Kun käyttäjä kohtaa tuotteen, hän arvioi, millainen tuote on kyseessä. Jokainen käyttäjä muodostaa siis oman henkilökohtaisen kuvansa siitä, että millainen tuote on. Käyttäjän muodostama kuva eroaa vaihtelevissa määrissä tuotteen suunnittelijoiden tavoittelemasta kuvasta, sekä muiden käyttäjien muodostamasta kuvasta. Ero voi johtua esimerkiksi vastaavien tuotteiden vaikutuksesta käyttäjän kokemukseen. Toisaalta käsitys tuotteesta voi myös muuttua ajan kuluessa.

Hassenzahlin mallissa käyttäjän toiminta riippuu käyttäjän muodostamasta kuvasta ja siitä tilanteesta, missä käyttäjä on. Käyttäjän muodostettua kuvan tuotteesta hän muodostaa seuraavaksi mielipiteen sen houkuttelevuudesta. Lisäksi tuote aiheuttaa käyttäjässä mahdollisesti joitain tunnereaktioita. Se, millaista nämä ovat, riippuu tilanteesta. Esimerkiksi jossakin tilanteessa käyttäjä saattaa pitää tuotteen uutuutta mukavana asiana. Jossain toisessa tilanteessa sama ominaisuus koetaan ikävänä. Käyttäjän muodostamasta kuvasta riippuen voi seurata erilaisia muutoksia henkilön käyttäytymisessä. Seuraukset käyttäjien tunteissa ja käytöksessä vaihtelevat enemmän kuin käyttäjien muodostamat kuvat tuotteesta. Tämä johtuu siitä, että käyttötilanne vaikuttaa seurauksiin enemmän kuin muodostettavaan kuvaan. (Hassenzahl, 2003)

Käyttäjäkokemus voidaan jakaa osa-alueisiin sen mukaan millaiseen tarpeeseen osa-alue vastaa. Usein kirjallisuudessa esiintyvä jaottelu on jakaa käyttäjäkokemus käytännöllisiin tekijöihin ja nautinnollisiin tekijöihin. Nämä tekijät jaetaan sitten lähteestä riippuen edelleen tarkemmiksi osa-alueiksi. Käytännölliset tekijät liittyvät nimensä mukaisesti käyttäjän tarpeeseen suorittaa tietyt tehtävät. Ne jaotellaan tyypillisesti käytettävyyteen ja käyttökelpoisuuteen. Nautinnolliset tekijät ovat vaikeammin määriteltävissä ja niiden jaottelu vaihtelee enemmän. Nautinnollisia tekijöitä voivat olla esimerkiksi esteettisyys, innovatiivisuus ja omaleimaisuus (Hassenzahl, 2001; Kujala *ym.*, 2011). Hassenzahl (2004) jakaa ne identifikaatioon, stimulaatioon ja siihen miten hyvin tuote herättää muistoja. Nautinnollisilla tekijöillä ei ole varsinaisesti tekemistä tehtävän suorittamisen kanssa, vaan ne liittyvät käyttäjään itseensä ja tämän psykologiseen hyvinvointiin.

## 2.2 Käytännölliset tekijät

Jotta käyttäjäkokemus on käytännöllisellä tavalla hyvä, käyttäjän pitää pystyä toimimaan sovelluksessa tehtävän vaatimalla tavalla. Tuotteessa täytyy siis olla tarvittavat toiminnallisuudet ja käyttäjän pitää pystyä käyttämään niitä. Hassenzahl (2008) jakaa käytännölliset tekijät **käyttökelpoisuuteen** (*utility*) ja **käytettävyyteen** (*usability*) Käyttökelpoisuus tarkoittaa, että tuote tarjoaa oleelliset toiminnallisuudet, tehtävien suorittamiselle ja ympäristön muokkaamiselle. Käytettävyys on puolestaan kykyä käyttää näitä ominaisuuksia (Hassenzahl, 2003) .

### Käytettävyys

Käytettävyys on yksi ohjelmiston laadullinen ominaisuus samoin kuin toimivuus, tehokkuus tai ylläpidettävyys (Kujala *ym.*, 2011). Käytettävyys määritellään ISO-9241-standardin (1998) mukaan sinä "**vaikuttavuutena, tehokkuutena ja tyytyväisyytenä, joilla palvelulle määritellyt käyttäjät saavuttavat määritellyt tavoitteet tietyssä ympäristössä**". Vaikuttavuudella tarkoitetaan sitä tarkkuutta ja täydellisyyttä, millä käyttäjät saavuttavat määritellyt tavoitteet. Tehokkuudella tarkoitetaan sitä, kuinka paljon resursseja tavoitteiden saavuttamiseen joudutaan käyttämään. Tyytyväisyydellä tarkoitetaan käyttäjän tyytyväisyyttä laitteen tai järjestelmän käyttöön, vuorovaikutuksen sujuvuuteen ja saavutettuihin tuloksiin.

Toinen suosittu käytettävyyden määritelmä on alan pioneerin Jacob Nielsenin (1993) määritelmä, joka jakaa käytettävyyden viiteen eri komponenttiin:

1. **opittavuus**, eli kuinka helppoa käyttäjien on suorittaa tavallisia toimenpiteitä ensimmäisellä käyttökerralla,
2. **tehokkuus**, eli kuinka nopeasti käyttäjät, jotka ovat oppineet käyttämään järjestelmää, pystyvät suoriutumaan tehtävistään,
3. **muistettavuus,** eli kuinka nopeasti käyttäjät pystyvät käyttämään järjestelmää jälleen tehokkaasti tauon jälkeen,
4. **kuinka paljon virheitä** käyttäjät tekevät, kuinka vakavia nämä virheet ovat ja **kuinka helppo niistä on toipua**,
5. kuinka **miellyttävää** käyttö on.

Nielsen (1995) on myöhemmin täsmentänyt käytettävyysperiaatteitaan websovelluksille. Websovelluksen opittavuus on se helppous, jolla käyttäjät ymmärtävät, että mitä sisältöä ja millaisia palveluja sovellus tarjoaa. Opittavuudesta kertoo mm. se, että kuinka tietty informaatio löytyy sovelluksesta. Jokaisen näkymän tulisi olla tehty niin, että sen sisältö on helposti ymmärrettävissä ja näkymästä on helppo erottaa, miten sivulta pääsee navigoimaan toisille sivuille. Websovellus on puolestaan tehokas, jos kaikki

sovelluksen sisältö on käyttäjien helposti saavutettavissa. Tehokkuus tarkoittaa myös sitä, että käyttäjän saapuessa näkymään hän pystyy ymmärtämään, missä on ja miten sivu sijaitsee suhteessa siihen, mistä käyttäjä saapui sivulle.

Nielsenin mukaan muistettavuus websovelluksessa tarkoittaa sitä, että käyttäjä pystyy liikkumaan sovelluksessa sen jälkeen, kun käyttäjä ei ole vähään aikaan käyttänyt sovellusta. Virhetilanteissa, jos käyttäjä on siirtynyt vahingossa väärälle sivulle, hyvä käytettävyys edellyttää, että hän pystyy siirtymään edelliseen sijaintiinsa. Käyttäjän tyytyväisyys taas tarkoittaa sitä, kuinka hyvin käyttäjä tuntee hallitsevansa tilannetta sovelluksessa, ja kumpuaa siitä, kuinka hyvin käyttäjä ymmärtää tarjolla olevan sisällön ja toiminnot.

Tarkasteltaessa määritelmiä huomataan, että käytettävyyden ISO-standardin mukainen määritelmä ja Nielsenin määritelmä ovat laajempia kuin Hassenzahlin mallissaan antama määritelmä. Nielsenin määritelmä eroaa muista määritelmistä siinä, että sen käytettävyyskäsitteeseen on lisätty aikaulottuvuus. Nielsenin määritelmässä käytettävyys on suure, joka voi muuttua ajan kuluessa, kun käyttäjä oppii käyttämään järjestelmää tai pitää taukoa järjestelmän käytöstä. Toisaalta huomataan myös, että mitä tarkemmin määritelmä menee yksityiskohtiin, sitä huonommin se kestää aikaa. Nielsenin yleinen käytettävyyden määritelmä on säilyttänyt arvonsa paremmin kuin websovelluksille annetut tarkennukset. Esimerkiksi se, mitä vaaditaan websovellukselta, jotta se toimisi käytettävästi virhetilanteissa, on nykynäkökulmasta hyvin kapea määritelmä. Nielsenin määritelmän puutteissa näkyy, kuinka websovellukset ovat muuttuneet hypertekstisovelluksista interaktiivisemmiksi ja monimutkaisemmiksi kokonaisuuksiksi. Käyttäjän näkökulmasta websovelluksen erottaa natiiveista sovelluksista nykyään usein vain se, että niitä käytetään selaimen kautta. Jopa monet perinteiset työpöytäsovellukset, kuten tekstinkäsittelyohjelmat, ovat siirtyneet käytettäväksi myös websovelluksena.

## 2.3 Hedonistiset tekijät

Tyypillisiä käytännölliseen tuotteeseen liitettyjä ominaisuuksia ovat selkeys, hallittavuus ja hyödyllisyys. Kuitenkin käyttäjäkokemuksessa on aina myös piirteitä, jotka eivät ole sidoksissa tuotteella suoritettaviin tehtäviin. Tällaisia nautinnollisia ominaisuuksia ovat esimerkiksi esteettisyys, imagoarvo tai kyky sosiaaliseen kanssakäymiseen. Voidaan ajatella, että nautinnollisiin tekijöihin liittyvät ominaisuudet ovat kilpailuetu etenkin vapaa-aikana käytettävissä sovelluksissa. Kuitenkin on havaittu, että houkutteleva ulkonäkö vaikuttaa myös siihen, kuinka käytettäväksi tuote koetaan. Yllättävää kyllä, visuaalisesti houkuttelevalla tuotteella saadaan tehtävät myös nopeammin suoritettua, jolloin käyttö on tehokkaampaa (Sonderegger ja Sauer, 2010).

Hassenzahl (2004) jakaa nautinnolliset tekijät kolmeen kategoriaan:

1. **stimulaatioon** (*stimulation*),

2. **indentifikaatioon** (*identification*) ja
3. siihen **miten se herättää muistoja** (*evocation*).

Stimulaatiolla tarkoitetaan sitä, kuinka tuote pystyy vastaamaan ihmisten tarpeeseen kehittää itseään. Se mittaa siis sitä, kuinka hyvin tuote pystyy tarjoamaan uusia mielikuvia, mahdollisuuksia ja oivalluksia. Seuraava kategoria on identifikaatio. Ihmiset haluavat, että muut näkevät heidät tietyllä tavalla. Identifikaatiolla viitataan ihmisten tarpeeseen ilmaista itseään esineiden välityksellä. Se on siis puhtaasti sosiaalisuuteen liittyvä ominaisuus. Tuotteen kyky herättää muistoja kertoo sen kyvystä toimia menneisyyden symbolina. (Hassenzahl, 2003, 2004) Esimerkiksi posliinikissa takanreunalla saattaa olla muilla mittareilla arvioituna täysin tarpeeton esine, mutta herättää rakkaita muistoja jo edesmenneestä isotädistä.

Kuusisen ym. (2016) mukaan kehittäjät kykenevät ymmärtämään käyttäjäkokemuksen käytännöllisiä tekijöitä melko hyvin. Heillä on kuitenkin taipumus jättää huomiotta tekijöitä, jotka eivät vaikuta tehtävien suorittamiseen (Kuusinen *ym.*, 2016). Jos järjestelmän ja käyttäjien vuorovaikutukseen ei kiinnitetä huomiota, järjestelmät suunnitellaan tyypillisesti toteuttamaan kasa toimintoja, ja vaikka tämä lähestymistapa voi olla insinöörityön näkökulmasta tehokas, tehokkuus saavutetaan usein järjestelmän käyttäjien kustannuksella (Preece, Rogers ja Sharp, 2002). Tuotetta ajatellaan siis lähinnä käyttökelpoisuuden kautta. Kun toiminnallisuudet ja niihin liittyvät käsitteet pyritään mallintamaan loogisesti ja teknisesti järkevällä tavalla, saatetaan luoda käyttäjälle vieraita rinnastuksia. Esimerkiksi yrityksellä ja henkilöllä voi molemmilla olla nimi ja osoite, mutta käyttäjän mielessä näillä ei ole mitään tekemistä toistensa kanssa, kun taas järjestelmässä ne voivat olla periytetty samasta kantaluokasta. Toisaalta käyttäjän mielikuvissa lähellä toisiaan olevilla käsitteillä ei järjestelmän kannalta ole välttämättä muuta yhteistä kuin sijainti. Kuitenkin, kun tuotetta käytetään pitkään, nautinnolliset tekijät vaikuttavat olevan tärkeämpiä käyttäjäkokemuksen kannalta kuin käytännölliset tekijät (Kujala *ym.*, 2011).

## 2.4 Käyttäjäkokemuksen suunnittelu

Tuotteen toimittaja pystyy vaikuttamaan sovelluksen käyttäjäkokemukseen sillä, miten tuote suunnitellaan ja toteutetaan. Käyttäjän kokemus riippuu tuotteen lisäksi kuitenkin myös käyttäjästä ja käyttöympäristöstä. Käyttäjäkokemusta ei siis pysty koskaan täysin luomaan tuotteen ominaisuuksilla, vaikka käyttäjät ja käyttöympäristö pystytäänkin ottamaan huomioon suunnittelun aikana. Kun luodaan sovellusta, suunnittelua tehdään monilla eri tasoilla. Voidaan ajatella, että suunnittelussa käytetään ylhäältä alas -suunnittelustrategiaa (*top-down)* eli siirrytään abstraktimmalta tasolta yksityiskohtiin. Voidaan aloittaa suunnittelu liiketoimintamallin suunnittelusta, siirrytään siitä palvelupolun suunnitteluun ja tästä edelleen käyttäjäkokemuksen suunnittelun kautta käyttöliittymän

yksittäisten elementtien suunnitteluun. Suuri osa tästä suunnittelusta on ohjelmistokehittäjälle näkymätöntä, koska se tehdään jo paljon aikaisemmin ennen kuin tuotetta aletaan toteuttaa. Lisäksi suunnittelu ostetaan usein eri toimistolta kuin itse toteutus. Toteutustiimin onkin välillä haasteellista ymmärtää, mihin suunnittelun aikana on pyritty.

Suunnittelu (*design*) itsessään on terminä hankala. Sitä käytetään monissa eri yhteyksissä. Erilaisten suunnittelukäsitteiden termistö on sekavaa ja eri suunnittelukäsitteet ovat sisällöltään osin päällekkäisiä. Yksi syy on se, että saman suunnittelu-ylätermin alle on kertynyt suunnittelusuuntauksia, jotka on tarkoitettu eri asioiden suunnitteluun ja joilla on eri historia ja tausta. Ne voivat juontaa juurensa esimerkiksi ihmisen ja tietokoneen välisen vuorovaikutuksen tutkimisesta tai markkinoinnin tai johtamisen aloilta. Sekavuutta lisää, että englanninkielinen termi *design* suomennetaan välillä myös muotoiluksi.

Ohjelmistotuotteen kehitykseen liittyviä suunnittelu-termejä ovat mm. **käyttäjäkokemuksen suunnittelu** (*user-experience design*), **palvelumuotoilu** (*service design*), **ihmis- tai käyttäjäkeskeinen suunnittelu** (*human-centered-design, user-centered-design*), **käyttöliittymän suunnittelu** (*user interface design*), **vuorovaikutussuunnittelu** (*interaction design*) ja **graafinen suunnittelu** (*graphical design*). Kun ihmisen ja koneen vuorovaikutusta on alettu ymmärtää paremmin ja kun tietotekniikasta on tullut yhä suurempi osa elämäämme, on myös käyttäjän kokemukseen vaikuttavan suunnittelun tärkeys nähty yhä laajemmin. Käytettävyys pääsi ISO-standardeihin vuonna 1998 ja ihmiskeskeinen suunnittelu vuonna 2010. Tällä hetkellä voidaan nähdä trendinä, että koko palvelupolun suunnittelu on yhä kiinteämmin osa ohjelmistotuotteen suunnittelua. Trendi näkyy mm. siinä, miten ohjelmistotalot ovat ostaneet ahkerasti palvelumuotoiluosaamista viime vuosina.

Seuraavissa aliluvuissa avataan tarkemmin käsitteitä käyttäjäkokemuksen suunnittelu eli UX-suunnittelu, ihmiskeskeinen suunnittelu ja palvelumuotoilu. Työhön liittyvissä haastatteluissa käytettiin erityisesti termejä palvelumuotoilu ja UX-suunnittelu. Jonkin verran puhuttiin myös käyttöliittymäsuunnittelusta, visuaalisesta suunnittelusta ja käytettävyyden suunnittelusta. Termien eri määritelmät eroavat toisistaan ja sisältävät päällekkäisyyksiä, eikä tiedetä, että mitä termi kullekin haastateltavalle täsmälleen tarkoitti. Toisaalta, koska termistö ei ole vakiintunutta, tässä työssä ei yritetä antaa kattavaa selitystä termien sisällöstä ja suhteista toisiinsa.

### 2.4.1 UX-suunnittelu

UX-suunnittelun eli käyttäjäkokemuksen suunnittelun (*UX design* eli *user experience* design) alaisuuteen kuuluu kaikki se, mikä vaikuttaa käyttäjän kokonaiskokemukseen tuotteen kanssa. UX-suunnittelu sekoitetaan joskus UI-suunnittelun (*UI = user interface*) eli käyttöliittymäsuunnittelun kanssa. Käyttöliittymäsuunnittelu on rajatumpi termi. Se on nimensä mukaisesti vain laitteiden ja tietokonesovellusten käyttöliittymien suun-

nittelemista. Käyttöliittymäsuunnittelun tavoitteena on tehdä käyttäjän ja sovelluksen vuorovaikutuksesta mahdollisimman sujuvaa ja tehokasta (Wikipedia, 2018c) .

Käyttäjäkokemuksen suunnittelu kattaa alleen monen suunnittelun alan asioita. Joko osittain tai kokonaan käyttökokemuksen suunnittelun alle kuuluvaksi Saffer (2009) luokittelee mm. interaktioiden suunnittelun, visuaalisen suunnittelun, informaatioarkkitehtuurin, ohjelmistoarkkitehtuurin, teollisen suunnittelun, äänisuunnittelun sekä ihmisen ja tietokoneen välisen vuorovaikutuksen. Informaatioarkkitehtuuri määrää sisällön rakenteen, eli sen, että kuinka sisältö on nimetty ja organisoitu niin, että käyttäjä löytää etsimänsä. Visuaalinen suunnittelu kattaa mm. käytetyt fontit, värit ja komponenttien sijoittelun käyttöliittymässä, siis kaiken, mikä liittyy käyttäjän kanssa kommunikointiin visuaalisesti. Teollisessa suunnittelussa muotoillaan käytettävät tuotteet niin, että ne viestivät käyttötarkoituksestaan ja toimivat tehtävässä, johon ne on tarkoitettu. Safferin luokittelua voi kritisoida siitä, kuinka paljon ohjelmistoarkkitehtuurin suunnittelun ajatellaan vaikuttavan käyttäjän kokemukseen. On totta, että on arkkitehtuurisia ratkaisuja, jotka vaikuttavat käyttäjän kokemukseen. Voidaan kuitenkin ajatella, että ohjelmistoarkkitehtuurin suunnittelu tulisi tehdä sen jälkeen, kun palvelumuotoilun ja käyttäjäkokemuksen suunnittelulla on selvitetty, mitä käyttäjä tarvitsee.

### 2.4.2 Käyttäjäkeskeinen suunnittelu

Käyttäjäkeskeinen suunnittelu *(UCD = user-centered design)* on suunnittelufilosofia, jossa käyttäjä asetetaan suunnittelun keskiöön. Sitä voidaan käyttää lähestymistapana, kun tehdään UX-suunnittelua. Käyttäjäkeskeisen suunnittelun lisäksi käytetään myös sisällöltään lähes vastaavaa termiä ihmiskeskeinen suunnittelu. Termejä käytetään tässä työssä samassa merkityksessä.

Ihmiskeskeinen suunnittelu on määritelty ISO 9241:210 -standardissa vuodelta 2010, jossa on ohjeita ihmiskeskeisten suunnittelumenetelmien käyttöön tietokonepohjaisia vuorovaikutteisia järjestelmiä varten koko systeemin elinajalle. Standardi määrittelee ihmiskeskeisen suunnittelun **lähestymistavaksi, jolla voidaan kehittää vuorovaikutteisia systeemejä ja jonka tavoitteena on tehdä järjestelmistä käytettäviä ja hyödyllisiä keskittymällä käyttäjiin, heidän tarpeisiinsa ja vaatimuksiinsa**. Suunnittelussa otetaan huomioon inhimilliset tekijät, ergonomia, tietämys käytettävyydestä ja käytettävyystekniikat.

Standardi määrittelee kuusi yleistä ihmiskeskeisen suunnittelun periaatetta, jotka eivät liity mihinkään erityiseen kehityssyklin vaiheeseen,

1. suunnittelu pohjautuu **käyttäjien ja tehtävien vaatimuksien hyvään ymmärtämiseen,**
2. **käyttäjät osallistuvat** kehitykseen suunnittelun ja kehitystyön aikana,
3. **käyttäjäkeskeinen arviointi** ohjaa ja hioo suunnitteluratkaisuja,

4. suunnitteluratkaisuja **iteroidaan,**
5. suunnittelussa **huomioidaan koko käyttäjäkokemus** ja
6. suunnittelussa käytetään **poikkitieteellisesti** useiden eri sovellusalueiden menetelmiä ja näkökulmia.

Kuvassa 4.2.1 esitetään ihmiskeskeisen suunnittelun prosessi. Prosessi alkaa ihmiskeskeisen suunnittelun tarpeen tunnistamisesta. Tämän jälkeen siirrytään luomaan iteroimalla ratkaisua, joka tyydyttää sekä käyttäjän että organisaation vaatimukset. Iteraatiosyklissä on neljä vaihetta: 1) käyttökontekstin ymmärtäminen ja täsmentäminen, 2) vaatimuksien määritteleminen, 3) suunnitteluratkaisujen luominen ja 4) suunnitteluratkaisujen arviointi. Ensimmäiseksi pyritään ymmärtämään ja täsmentämään käyttökonteksti. Tämä tarkoittaa käyttäjän, käyttöympäristön ja käyttäjän tehtävien tuntemista. Oleellista tietoa käyttäjistä on esimerkiksi käyttäjien taidot, koulutus ja kokemus. Sovelluksesta riippuen myös käyttäjän fyysisillä ominaisuuksilla, tavoilla, mieltymyksillä ja muilla kyvyillä voi olla merkitystä. Jos on tarpeellista, käyttäjien ominaisuudet voidaan määritellä käyttäjätyypeittäin. Esimerkiksi sovelluksen ylläpitäjällä voi olla erilaiset taidot kuin sovelluksen loppukäyttäjällä. (Kuusinen, 2015a)Toinen osa käyttökontekstia on käyttöympäristö. Käyttöympäristöön kuuluu käytettyjen laitteiden ja ohjelmistojen lisäksi myös fyysinen ja sosiaalinen ympäristö, jossa tuotetta tullaan käyttämään. Käyttöympäristön kannalta oleellisia voivat olla myös tuotteen käyttöön liittyvät lait tai kulttuuriset tekijät. Kolmannen osan käyttökontekstista muodostaa käyttäjän tehtävien tunteminen tarpeeksi hyvin. Sen lisäksi mitä tehdään, tulee tietää myös tehtävän käyttäjien suorittamien tehtävien kesto ja kuinka usein niitä suoritetaan. Lisäksi kuvauksesta tulisi käydä ilmi käyttäjän lopulliset tavoitteet, eli se miksi käyttäjä käyttää järjestelmää.(SFS-EN ISO 9241-210, 2010)

Kun ymmärretään käyttökontekstia, määritellään seuraavaksi käyttäjän ja organisaation vaatimukset. Vaatimuksissa tulisi määritellä käyttäjät ja muut suunnitteluun oleellisesti liittyvät henkilöt, esittää selkeästi ihmiskeskeiset suunnittelutavoitteet ja priorisoida vaatimukset. Käyttäjien suorittamien tehtävien lisäksi tulee ottaa huomioon useita erityyppisiä asioita, kuten lainsäädännöstä tulevat vaatimukset, liiketoiminnan asettamat vaatimukset, ylläpidettävyys ja käyttäjien koulutus. Tuotteen käytettävyydelle määritellään vaatimusten pohjalta onnistumiskriteerit. Kriteerit määritellään tehtävittäin. Esimerkkinä voisi olla kriteeri, kuinka nopeasti tyypillinen käyttäjä pystyy tallentamaan lisäämänsä tiedot. Kriteerien tulee olla sellaisia, että niiden toteutumista voidaan mitata. Kerätyt vaatimukset ja niiden onnistumiskriteerit varmistetaan käyttäjiltä. Myös riittävästä dokumentaatiosta tulee huolehtia.(SFS-EN ISO 9241-210, 2010; Kuusinen, 2015)


***Kuva 4.2.1.****Käyttäjäkeskeinen suunnitteluprosessi*

Seuraavassa vaiheessa tuotetaan suunnitteluratkaisut kerätyn tiedon pohjalta. Viimeisessä iteraation vaiheessa suunnitteluratkaisut evaluoidaan luotuja vaatimuksia vasten. Jos suunnitteluratkaisu ei täytä vielä vaatimuksia, aloitetaan uusi iteraatiosykli. Iterointia jatketaan, kunnes vaatimukset on saavutettu.(SFS-EN ISO 9241-210, 2010)

Ihmiskeskeisen suunnittelun filosofia on yksinkertainen: käyttäjät tietävät parhaiten omat tarpeensa, tavoitteensa ja mieltymyksensä (Saffer, 2009). Suunnittelijan tehtäväksi jää selvittää ne ja suunnitella käyttäjien tarpeet täyttäviä järjestelmiä. Endsleyn (2016) mukaan tämä ei kuitenkaan tarkoita, että käyttäjiltä kysyttäisiin, mitä he tarvitsevat ja annettaisiin se käyttäjille. Ensinnäkin käyttäjät tietävät usein vain osittain, mitä he haluavat verrattuna nykytilanteeseen. Käyttäjillä on yleensä myös hyvin rajoittunut käsitys siitä, kuinka nämä tarpeet voidaan tehokkaasti toteuttaa käyttöliittymässä. Lisäksi Endsleyn mukaan useimmilla systeemeillä on useita eri käyttäjiä, joilla on kaikilla omat käsityksensä siitä, mitä he haluaisivat toteutettavan järjestelmään. Tuloksena on suunnitteluratkaisuja, jotka ovat tyypillisesti satunnaisia ja epäsäännönmukaisia. Lisäksi Endsley huomauttaa, että useita suunnitteluvirheitä ei edes huomata.

### 2.4.3 Palvelumuotoilu

Palvelun määrittely on vaikeaa. Oleellista kuitenkin on, että palvelussa keskeinen osa on aineeton ja se on prosessi, joka ratkaisee asiakkaan jonkin ongelman (Tuulaniemi, 2011). Kotler ja Keller (2012) määrittelevät palvelun miksi tahansa aineettomaksi aktiviteetiksi tai hyödyksi, jonka yksi osapuoli voi antaa toiselle ja jossa minkään tuotteen

omistajuus ei vaihdu. Palvelut on sidottu aikaan ja ne ovat vuorovaikutteisia systeemejä. Monista palveluista on tullut nykyisin virtuaalisia. Toisaalta palvelu kokonaisuudessaan voi tapahtua osin digitaalisesti, osin perinteisesti. Williams ym. (2010) määrittelevät digitaaliset palvelut palveluiksi, jotka saavutetaan digitaalisen transaktion välityksellä käyttäen internetprotokollaa. Digitaalisissa palveluissa käytetään siis vuorovaikutukseen rajoittuneempia ja ennaltamäärättympiä polkuja kuin perinteisissä palveluissa. Perinteisissä palveluissa voidaan soveltaa enemmän palvelupolkua ja muuttaa käytöstä paremmin tilanteeseen sopivaksi.

Palvelumuotoilun keskeinen käsite on asiakaskokemus. Tuulaniemen mukaan asiakas on aina osa palvelutapahtumaa ja hän muodostaa kokemuksensa palvelusta joka kerta palvelutapahtumassa henkilökohtaisesti uudestaan. Kokemus on subjektiivinen ja tapahtuu asiakkaan pään sisällä. Palvelumuotoilun tavoitteena on saada mahdollisimman hyvä palvelukokemus. Tämä saavutetaan optimoimalla palveluprosessi, työtavat, tilat ja vuorovaikutus sekä poistamalla palvelua häiritsevät asiat. Palvelumuotoilun tavoitteena on osallistaa prosessiin kaikki palvelussa mukana olevat osapuolet.

Palvelupolku on palvelunkokonaisuuden kuvaus. Se kuvaa sen, miten asiakas liikkuu palvelussa ja kokee palvelun aika-akselilla. Palvelupolku jaetaan edelleen palvelutuokioihin ja nämä jakautuvat edelleen palvelun kontaktipisteisiin. Kontaktipisteitä ovat ihmiset, ympäristöt, esineet ja toimintatavat; niiden kautta asiakas on kontaktissa palveluun kaikilla aisteillaan. (Office, 2011; Tuulaniemi, 2011)

Palvelumuotoilulle on ominaista muodostaa palveluun vaikuttavista tekijöistä kokonaiskuva, jonka mukaan palvelua kehitetään. Toisaalta ongelmanratkaisua tehdään jakamalla palvelu osakokonaisuuksiin ja edelleen pieniin, yksittäisiin elementteihin. Näin niitä pystytään hallitsemaan paremmin ja ne voidaan optimoida tavoitteiden mukaisiksi. Nämä optimoidut osakokonaisuudet yhdistetään kokonaisratkaisuksi. (Tuulaniemi, 2011) Periaatteessa siis palvelumuotoilun tavat ratkaista ongelmia vastaavat ohjelmistotekniikan vastaavia.

Palvelumuotoilu ja UX-suunnittelu ovat termejä, joita käytetään paljon ohjelmistoalalla. Ne asettavat molemmat ihmisen - käyttäjän tai palvelun asiakkaan - keskiöön. Molemmat ovat monialaisia ja iteratiivisia. Molemmat myös käyttävät työkaluinaan erilaisia visualisointeja ja haastattelumenetelmiä. Palvelumuotoilu toimii kuitenkin ylemmällä tasolla kuin UX-suunnittelu. Kun UX-suunnittelussa keskitytään tyypillisesti käyttäjän vuorovaikutukseen tietyn sovelluksen kanssa, palvelumuotoilu tarkastelee laajemmin käyttäjän kohtaamisia sovelluksen ja sovelluksen tarjoajan kanssa. Sovelluksen UX-suunnittelu siis tavallaan alkaa siitä pisteestä, missä palvelumuotoilija on saanut asiakkaan palvelupolun suunniteltua. Palvelumuotoilu alana on periaatteessa lähtöisin markkinoinnin ja muotoilun aloilta (Office, 2011), vaikka osa teknistaustaisista UX-suunnittelijoista onkin siirtynyt kutsumaan itseään palvelumuotoilijoiksi. UX-suunnittelu puolestaan on lähtenyt liikkeelle ihmisen ja tietokoneen välisen vuorovaikutuksen tutkimisesta. Tämä selittää osin lähestymistapojen eroja ja tarkoittaa toisaalta myös sitä,

että palvelumuotoilu ja UX-suunnittelu voidaan määritellä enemmän tai vähemmän päällekkäisiksi termeiksi.

# 3 SUUNNITTELUPOHJAISISTA MENETELMISTÄ KETTERIIN

Vähänkään suuremman kokonaisuuden toteuttamisessa tarve jollekin ohjelmistotuotannon prosessimallille on ilmeinen. Erilaisia malleja on nykyisin hyvin paljon. Yhteistä niiden käytännön toteutukselle on kuitenkin yleensä se, että niitä harvoin toteutetaan puhtaasti. Todellisuudessa projekteissa valitaan osia prosessimalleista ja yhdistetään niihin omia hyväksihavaittuja käytäntöjä. Arkikielenkäytössä vesiputousmalli ja Scrum ovat saavuttaneet vahvimman aseman. Ne ovat malleja, jotka kaikki alalla tuntevat ja niitä käytetään usein vertailukohtana, kun halutaan hahmottaa jotain toista mallia.

Ohjelmistotuotannon prosessimallit voidaan jakaa **suunnittelupohjaisiin** - eli prosessi-orientoituneisiin malleihin ja **ketteriin** malleihin. Suunnittelupohjaisien mallien perusajatuksena on, että ohjelmiston vaatimukset lyödään lukkoon täysin ennen kuin ohjelmiston suunnittelu ja kehitys aloitetaan (McCauley ja Renée, 2001). Abrahamsson ym. (2002) listaavat suunnittelupohjaisten mallien piirteeksi, että ohjelmistoprojekti nähdään hallittavana, toistettavana, universaalina prosessina. Heidän mukaansa prosessi on lineaarinen ja siinä edetään vaihe vaiheelta. (Abrahamsson *ym.*, 2002) Ketterissä malleissa puolestaan painotetaan iteratiivista kehitystä, jossa vaatimukset täydentyvät projektin edetessä. Aloitettaessa ketterää ohjelmistototeutusta ei siis tiedetä tarkalleen, mitä tullaan tekemään.

Prosessimalleista puhuttaessa törmätään usein termeihin iteratiivinen ja inkrementaalinen. **Iteratiivisissa** menetelmissä kehityksen voi ajatella kulkevan sykleissä. Yksittäinen ominaisuus tai kokonaisuus käy läpi useita versioita. Käytännössä tämä tehdään tyypillisesti niin, että käytetään lyhyitä, säännöllisen kokoisia kehityssyklejä. Jokaisen kehityssyklin lopputuloksena on testattava systeemi, jota voi käyttää. Iteratiivisuuden ideana on käyttää edellisen vaiheen palautetta, jotta saadaan ohjelmisto, joka sopii käyttäjien tarpeisiin. Ominaisuudesta voidaan esimerkiksi tehdä prototyyppi, josta kerätään palautetta käyttäjiltä ja prototyypin pohjalta tehdään parannettu versio.(C. Larman, 2004) **Inkrementaalinen** malli puolestaan edellyttää, että jokaisessa uudessa versiossa on joku uusi käyttäjälle näkyvä ja testattava uusi kokonaisuus (Subhajit, 2007). Eli inkrementaalisessa mallissa jokaisessa kehitystyön aikaisessa julkaisussa ohjelmiston ominaisuuksien määrän tulisi kasvaa. Monet mallit ovat sekä inkrementaalisia että iteratiivisia, mutta inkrementaalisuus ei edellytä iteratiivisuutta eikä päinvastoin.

## 3.1 Vesiputousmalli

Useimmat suunnittelupohjaisista malleista pohjautuvat Roycen vuonna 1970 artikkelissaan *Development of Large Software systems* esittelemään malliin. Julkaisu saattaa olla ohjelmistokehityksen historian väärinymmärretyin julkaisu. Roycen pojan mukaan hänen isänsä oli aina iteratiivisen kehityksen kannattaja ja että julkaisu kuvaa iteratiivista kehitystä 60–70-lukujen julkishallinnollisten projektien rajoitusten puitteissa (Larman ja Basili, 2003). Roycen julkaisu ei esitäkään mallia täydellisenä prosessimallina, vaan julkaisu sisältää myös kritiikkiä. Esimerkiksi Royce pitää ongelmana testausvaiheen jättämistä kehitystyön loppuun. Hän sanoo, että jos ohjelmisto ei täytä testausvaiheessa ulkoisia rajoituksia, ongelma täytyy korjata merkittävällä uudelleen suunnittelulla. Ohjelmiston kehitys täytyy käytännössä aloittaa alusta ja voidaan odottaa 100 prosentin ylitystä aikatauluihin ja/tai kuluihin. Roycen mallia on alettu myöhemmin kutsua vesiputousmalliksi. Yhdysvaltain puolustusvoimat otti 1985 vesiputousmallin standardiinsa DOD-STD-2167A, joka oli tarkoitettu ohjelmistotoimittajien kanssa työskentelyyn (Wikipedia, 2018d). Tämä lienee vaikuttanut siihen, että vesiputousmallista on tullut eräänlainen arkkityyppi, johon muita malleja verrataan.

Vesiputousmalli jakaa prosessin seitsemään eri vaiheeseen:

1. vaatimusmäärittelyyn,
2. suunnitteluun,
3. toteutukseen,
4. integraatioon,
5. testaukseen,
6. asennukseen ja
7. ylläpitoon.

Vaatimusten määrittelyvaiheessa **kaikki vaatimukset dokumentoidaan**. **Suunnitteluvaihe tuottaa sovelluksen arkkitehtuurin**. Huomattakoon, että myös käyttöliittymä määritellään tässä vaiheessa. Toteutusvaiheessa ohjelmisto toteutetaan ja toteutus testataan. Kun ohjelmisto on saatu toteutettua valmiiksi, ohjelmistokomponentit integroidaan valmiiksi järjestelmäksi, jolle tehdään järjestelmätestaus. Kun järjestelmä läpäisee testit, se voidaan asentaa ja voidaan siirtyä ylläpitovaiheeseen. (Royce, 1970)

Vesiputousmalli nähdään usein pelkkänä putkena, jonka läpi prosessi ohjataan, mutta todellisuudessa se sisältää myös iteratiivisia piirteitä. Royce itse näki tärkeäksi, että eri vaiheiden välillä voidaan tarvittaessa palata myös taaksepäin ja hän oli jopa sitä mieltä, että tehtäessä uutta ohjelmistoa vesiputousmalli tulisi käydä läpi kahdesti. Ensimmäinen iteraatio tuottaisi prototyypin ja seuraava iteraatio vasta valmiin tuotteen. Toisaalta vesiputousmallissa, toisin kuin myöhemmissä ketterissä menetelmissä, koko

sovellus määritellään ja suunnitellaan ennen toteutuksen aloittamista. Haikalan ja Mikkosen (2011) mukaan “sallimalla iterointi ja vaiheiden käynnistäminen jo ennen edellisen vaiheen loppua saadaan kuitenkin moneen tilanteeseen sopiva toimintamalli”.

Mitä myöhemmin virheet ohjelmistoissa havaitaan, sitä kalliimmiksi ne yleensä tulevat (Haikala ja Märijärvi, 2004). Vesiputousmallin vastaus tähän on suunnitella ohjelmisto hyvin valmiiksi ennen kuin siirrytään seuraaviin vaiheisiin. Ongelmana tässä lähestymistavassa on kuitenkin se, että tähdätään liikkuvaan maaliin. Asiakkaan ja käyttäjien tarpeet muuttuvat jatkuvasti. Voi olla, että sovellus on jo valmistuessaan vanhentunut. Yksi tapa ratkaista putkimaisten prosessimallien ongelmat oli siirtyä käyttämään inkrementaalisia malleja. Tälläinen malli on esimerkiksi inkrementaalinen vesiputousmalli, jossa toteutettavat vaatimukset jaetaan iteraatioihin. Nämä iteraatiot viedään läpi käyttämällä vesiputousmallia (Pressman, 2010). Iteraatioiden pituus voi olla esimerkiksi kuukauden mittainen ja asiakastoimitukset voidaan tehdä puolen vuoden välein (Haikala ja Märijärvi, 2004).

Costabile (2000) kuvaa etukäteissuunnitteluun perustuvien mallien ongelmia käytettävyyden suhteen. Costabile kuvaa malleja systeemikeskeisiksi. Hänen mukaansa vesiputousmallissa ei kiinnitetä käytettävyyteen huomiota. Lisäksi vesiputousmallissa on Costabilen mukaan käytettävyyden kannalta myös muita ongelmia. Esimerkiksi systeemitestaus tehdään vasta ohjelmistotuotantosyklin loppuvaiheessa. Silloin on liian myöhäistä tehdä suuria muutoksia ohjelmistoon. Toinen Costabilen esiin nostama ongelma on, että asiakkaat, joiden kanssa vaatimukset kerätään, eivät ole usein järjestelmän loppukäyttäjiä. Lisäksi hänen mukaansa vaatimukset painottuvat yleensä toiminnallisiin vaatimuksiin ja käytettävyyteen liittyvät vaatimukset jäävät huomiotta. Vaatimuksien puuttuessa käytettävyyttä ei myöskään testata järjestelmätestauksessa.

## 3.2 Ketterät menetelmät

Uusohjelmistokehitys muistuttaa enemmän perinteistä käsityöläisyyttä kuin teollista tuotantoa siinä, että ohjelmistot räätälöidään asiakkaiden kulloistenkin tarpeiden mukaisiksi. Toisaalta samoin kuin perinteisessä käsityöläisyydessä, myös ohjelmistokehityksessä asiakkaiden ongelmat toistuvat usein samantapaisina. Yksi ketterän kehityksen taustalla olevista ajatuksista on sen tunnustaminen, että ohjelmistokehitys on samankaltaista kuin uusien ja innovatiivisten tuotteiden luominen (Craig Larman, 2004). Düchtingin ym. (2007) mukaan on harvoin mahdollista kerätä kaikkia vaatimuksia etukäteen ja määritellä sekä aikatauluttaa kaikkia tehtäviä toimenpiteitä. Ketterissä menetelmissä muutokseen varaudutaan ja vaatimuksia tarkennetaan prosessin edetessä. Lean vie ajatuksen vielä pidemmälle, Lean-filosofian mukaan ratkaisut tehdään mahdollisimman myöhäisessä vaiheessa.

Ketterien menetelmien käytön tärkeänä merkkipaaluna pidetään vuonna 2001 julkaistua **Ketterän kehityksen julistusta** (*=agile manifest).* Tämä julistus määritteli ket-

terien menetelmien arvot ja antoi pohjan sen jälkeen kehitettyjen ketterien menelmien luomiseksi. Näissä arvoissa painotetaan 1) **yksilöitä ja kanssakäymistä** enemmän kuin prosesseja ja työkaluja, 2) **toimivaa ohjelmistoa** enemmän kuin kattavaa dokumentaatiota, 3) **yhteistyötä asiakkaan kanssa** enemmän kuin sopimusneuvotteluja ja 4) **muutokseen reagoimista** enemmän kuin valmiin suunnitelman seuraamista (Beck *ym.*, 2001). Kannattaa huomata, että tämä ei tarkoita sitä, että dokumentaatiota tai sopimusneuvotteluja ei tarvittaisi.

Arvot toteutuvat julistuksen kahtenatoista perusarvona, jotka tiivistettynä kuuluvat:

1. asiakas pidetään tyytyväisenä toimittamalla **toimivia versioita ohjelmistosta tasaisella tahdilla**,
2. **vaatimuksia voidaan muuttaa** missä vaiheessa projektia tahansa,
3. toimivia versioita toimitetaan asiakkaalle **muutaman viikon – muutaman kuukauden välein,**
4. projektissa olevien henkilöiden tulee **työskennellä päivittäin yhdessä** koko projektin ajan,
5. projekti rakennetaan **motivoituneiden yksilöiden ympärille**, joilla on **tarvittavat välineet, tuki ja luottamus**,
6. **keskustelu kasvokkain** on kaikkein tehokkain kommunikaatiomenetelmä,
7. **toimiva ohjelmisto** on tärkein mittari projektin edistymiselle,
8. **työkuorma pidetään tasaisena** ja ylitöitä ei tehdä,
9. kiinnitetään jatkuvaa huomiota **tekniseen erinomaisuuteen** ja **hyvään suunnitteluun**,
10. asiat tehdään mahdollisimman yksinkertaisesti **minimoiden turhat tehtävät**,
11. tiimien tulisi olla **itseorganisoituneita**,
12. prosessia ja **työtapoja kehitetään jatkuvasti** tehokkaammiksi. (Beck *ym.*, 2001)

Ketterissä menetelmissä siis sovellus kehitetään sykleissä, jotka tuottavat aina jotkin toiminnalliset kokonaisuudet. Näin päästään kehitystyön aikana hyötymään siitä, että sovellusalueen tuntemus kehittyy työn edetessä. Ketteriä menetelmiä on syntynyt erilaisia, joista tässä työssä kerrotaan tarkemmin Scrumista. Luvussa 3.3 käsiteltävät Lean-menetelmät lasketaan välillä ketteriin menetelmiin, samoin kappaleessa 3.4 käsiteltävä SAFe. Muita ketteriä menetelmiä ovat mm. XP (*XP = Extreme Programming*) ja RUP ( *RUP = Rational Unified Process* ). **XP** on käytännöiltään lähellä Scrumia. Molemmat menetelmät pohjautuvat ketterän kehityksen manifestiin ja jakavat näin paljon yhteisiä piirteitä,

kuten kehityksen iteraatioissa ja nopeat julkaisusyklit. Kuitenkin XP:n ytimessä on 12 toisiinsa liittyvää ohjelmistokehityksen toimintatapaa, kuten parikoodaus, testaus ennen toteutusta (*test driven development*) ja jatkuva integraatio (Agile Alliance, 2018). Scrum ei puolestaam ota kantaa ohjelmointitekniikoihin vaan on pikemminkin projektinhallinnallinen kannanotto. Partogin (2018) mukaan XP ja Scrum eivät sulje toisiaan pois, vaan XP:n käytännöt voi yhdistää Scrumiin.

### Scrum

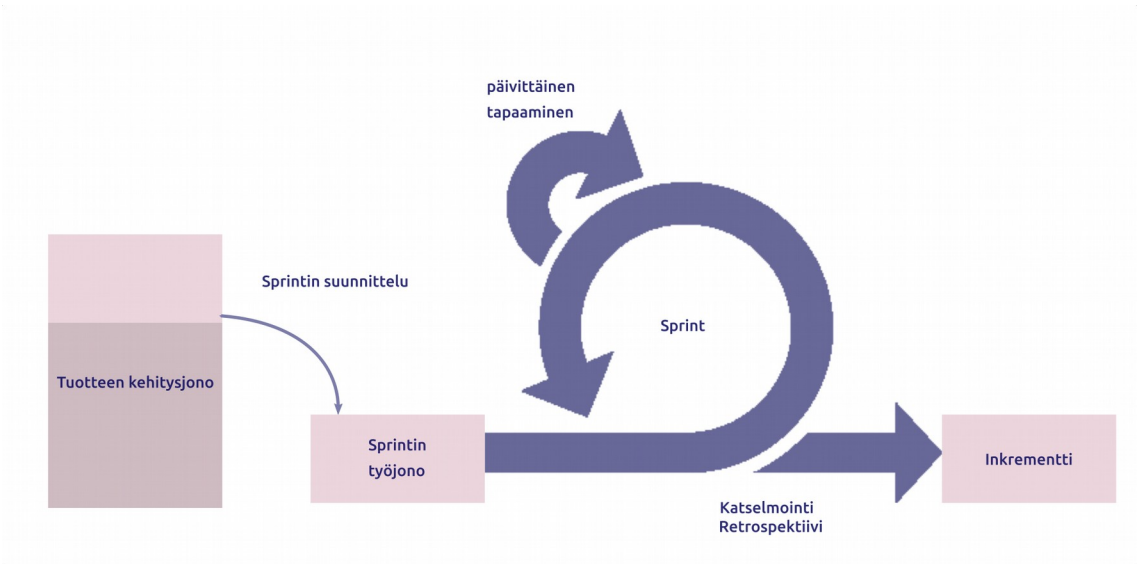
Scrum on suosittu ohjelmistotuotannon ketterä prosessikehys, joka on helppo ymmärtää, mutta vaikea hallita. Usein Scrumiksi kutsutaan myös sellaisia prosesseja, jotka toteuttavat vain osan Scrumissa määritellyistä asioista. Määritelmän mukaan Scrumin toteuttaminen vaatii kuitenkin kaikkien Scrumin periaatteiden noudattamisen.

Scrumin kolme peruspilaria ovat läpinäkyvyys, tarkastelu ja sopeuttaminen. **Läpinäkyvyys** tarkoittaa sitä, että kaikki henkilöt, jotka ovat prosessissa mukana, yksittäisestä kehittäjästä toimitusjohtajaan, ovat läpinäkyviä päivittäisessä kanssakäymisessään toisten kanssa. Kaikki luottavat toisiinsa ja pitävät toisensa ajan ajantasalla niin hyvistä kuin huonoista uutisista. **Tarkastelu** on toiminnan ja prosessissa syntyvien tuotteiden jatkuvaa avointa tarkastelua. **Sopeuttaminen** puolestaan on jatkuvaa toiminnan parantamista tarkastelemalla saatujen parannusehdotusten perusteella.(Scrum.org, 2017; Hiren, 2016)

Scrumin ydinroolit ovat 1) **tuoteomistaja**, 2) **kehitystiimi** eli Scrum-tiimi ja 3) **scrum-masteri**. Perusperiaate on, että tuoteomistaja huolehtii siitä, mitä tehdään, tiimi huolehtii siitä, kuinka tehdään ja Scrum-masteri on palveleva johtaja, joka huolehtii Scrumin toteutumisesta. Tuoteomistaja kerää järjestelmän vaatimukset kaikilta sidosryhmiltä. Hän on vastuussa tuotteesta ja siitä, että tehty työ tuottaa mahdollisimman paljon arvoa asiakkaalle. Scrumin oppien mukaan tuoteomistaja on aina yksittäinen henkilö, ei komitea. Tuoteomistajalla voi olla apunaan esimerkiksi kehitysryhmä, mutta vastuun tuotteen sisällöstä tulee säilyä tuoteomistajalla. Kehitystiimi on itseohjautuva ja itseorganisoituva. Sen jäsenillä on kaikki tarvittava osaaminen, joka tarvitaan tuotteen kehittämiseksi. Optimaalinen tiimin koko on 3-9 henkilöä. Se on tarpeeksi pieni määrä, jotta tiimi pysyisi ketteränä ja tarpeeksi suuri määrä, jotta merkityksellinen määrä työtä pystytään tekemään yhden sprintin aikana. Suuren tuotteen kehityksessä voi olla mukana useita kehitystiimejä. Scrum-masteri on kehitystiimin jäsen ja sen palveleva johtaja. Hän vastaa siitä, että kehitystiimi ja tuoteomistaja noudattavat Scrumin sääntöjä ja käytäntöjä. Scrum-masteri tuntee tuotteen ja iteraation vision ja kulloisenkin tavoitteen. Hänen tehtäviinsä kuuluu myös erilaisten kokousten organisointi ja organisaation valmentaminen scrumin käyttöön. (Scrum Guides.org, 2016)

Scrumissa työ tehdään sprinteissä, jotka kestävät korkeintaan 30 päivää. Tämä aika ajatellaan ylärajaksi sille, että sprintin aikana tehtävissä oleva työ olisi ennustettavissa. Scrum-sprintti on esitetty kuvassa 3.2.1. Kaikki ominaisuudet, toiminnallisuudet,

vaatimukset ja parannukset, mitä tuotteeseen saatettaisiin tarvita, on listattu **tuotteen kehitysjonoon** (*product backlog*), joka täydentyy ja täsmentyy jatkuvasti. Sprintin **suunnittelupalaverissa** (*sprint planning)* suunnitellaan sprintin aikana tehtävä työ. Suunnittelupalaveriin osallistuu koko scrum-tiimi ja tuoteomistaja. Ennen suunnittelupalaveria, missä tahansa sprintin vaiheessa, tuoteomistaja ja kehitystiimi käyvät läpi tuotteen kehitysjonon alkioita ja täsmentävät erityisesti korkealla tärkeysjärjestyksessä olevien alkioiden kuvauksia (*backlog refinement*). Vastuu tuotteen kehitysjonosta on kuitenkin tuoteomistajalla. Vain tuoteomistaja saa muuttaa kehitysjonon sisältöä ja järjestystä. (Scrum Guides.org, 2016; Flewelling, 2018)




*Kuva 3.2.1.Scrum-sprintin kulku*

Kehitystiimi valitsee sprintin suunnittelupalaverissa sopivan määrän tehtäviä tuotteen kehitysjonosta **sprintin työjonolle** (*sprint backlog*). Kehitystiimi päättää itse sen, mitä tiimi pystyy toteuttamaan. Tiimi päättää myös itse sen, miten valitut tehtävät toteutetaan. Kun tehtävä työ on määritelty, tiimi määrittelee yhdessä sprintin tavoitteen, jonka tehtävä on ohjata ja muistuttaa kehitystiimiä siitä, miksi se on kehittämässä tuoteversiota. Kun tavoitteet on lyöty lukkoon, niitä ei enää muuteta sprintin aikana. Tarvittaessa tuoteomistaja voi kuitenkin keskeyttää sprintin, jos havaitaan, että kehitettävä tuote on väärä. (Scrum Guides.org, 2016) Näin kehitystiimi saa työrauhan asiakkaan muuttuvilta vaatimuksilta sprintin ajaksi.

Sprintin aikana pidetään **päivittäinen Scrum-tapaaminen** (*daily scrum*), joka kestää korkeintaan 15 minuuttia. Siinä kehitystiimi kertoo toisilleen, mitä edellisen päivittäisen tapaamisen jälkeen on tehty, mitä tänään aiotaan tehdä ja miten nämä auttavat saavuttamaan sprintin tavoitteen. Tapaamisessa kerrotaan myös, jos on olemassa jotain esteitä, jotka estäisivät tiimiä saavuttamasta sprintin tavoitetta. (Scrum Guides.org,

2016) Päivittäisen tapaamisen etuna on se, että koko tiimi pysyy kärryillä sprintin etenemisestä ja mahdollisia ongelmakohtia pystytään pohtimaan yhdessä.

Kun sprintti on lopussa, pidetään **sprintin katselmointi (***sprint review*). Siinä kehitystiimi, tuoteomistaja ja tuoteomistajan kutsumat sidoshenkilöt tarkastelevat, mitä sprintin aikana saatiin aikaiseksi. Tutkitaan, mitä tehtäviä kehitysjonosta on saatu tehtyä ja osallistujat pohtivat, mitä seuraavaksi voitaisiin tehdä. Samoin kuin muut Scrumin tapaamiset, myös sprintin katselmoinnit ovat aikarajattuja. Ne saavat kestää maksimissaan neljä tuntia. Sprintin katselmoinnin lopputuotteena on uudistettu kehitysjono, jossa on todennäköiset tehtävät seuraavalle sprintille. (Scrum Guides.org, 2016)

**Sprintin retrospektiivi** (*retrospective*) on viimeinen kokous ennen sprintin loppumista. Se on tapaaminen, jossa kehitystiimi tarkastelee toimintaansa ja pohtii miten voisi parantaa sitä seuraavassa sprintissä. Tarkastelun kohteena on sprintin sujuminen, niin henkilöiden, ihmissuhteiden, prosessien ja kuin työkalujenkin suhteen. Lisäksi etsitään ja laitetaan tärkeysjärjestykseen asiat, jotka menivät hyvin ja asiat, joissa olisi ollut parantamisen varaa. Havaintojen pohjalta tehdään suunnitelma, mitä tiimin työssä tulisi parantaa. (Scrum Guides.org, 2016)

## 3.3 Lean

Lean ei ole oikeastaan menetelmä vaan ennemmin filosofia. Tai kuten Charette (2018) kuvaa: ”Se (*Lean*) on oikeastaan projektikäytäntöjen, periaatteiden ja filosofian synteesi, jonka avulla tuotetaan ohjelmistoja asiakkaiden käyttöön”. Lean-filosofiassa painotetaan erityisesti arvon luomista asiakkaalle ja hukan välttämistä. Painopisteen ei siis tulisi olla itse prosessissa ja tiettyjen metodien mukaisissa toimintatavoissa vaan siinä, miten ohjelmistoja voidaan lopulta käyttää arvon tuottamiseen asiakkaalle. Lean-ajattelua ohjaa viisi toisiinsa liittyvää käsitettä:

1. **Arvo**: Loppukäyttäjä määrittelee tuotteen arvon.
2. **Arvoketju**: Arvoketju näyttää prosessin jokaisen askeleen ja näyttää, kuinka paljon arvoa se lisää.
3. **Virta**: Tuottava työ virtaa jatkuvasti prosessissa.
4. **Imu:** Asiakkaan tilaukset saavat aikaan tuotteen tekemisen, mitään ei rakenneta ennen kuin sitä tarvitaan.
5. **Täydellisyys:** Pyritään täydellisyyteen parantamalla prosessia jatkuvasti identifioimalla ja poistamalla hukkaa. (Wang, Conboy ja Cawley, 2012)

Lean pohjautuu Toyotan toisen maailmansodan jälkeen kehittämään tuotantotapaan. Wardin mukaan perinteinen johtaminen perustuu kahteen 1600-luvulla kehitettyyn oletukseen: 1) **Ulkopuolinen viisaampi tekijä tarvitaan**, jotta järjestelmään voidaan

luoda järjestys ja 2) **järjestelmät ovat ennustettavissa**. Toisin sanoen johdon tehtävänä on kertoa työntekijöille, mitä tehdään ja tehdyn työn seurauksena saadaan oletettu lopputulos. Maailmansodan jälkeisessä Japanissa näiden periaatteiden noudattaminen ei ollut mahdollista, sillä Toyotalta puuttui sekä materiaaleja, osaavaa työvoimaa että taitoa rakentaa autoja. Toyotan tietämys autojen valmistuksesta rajoittui puolen vuoden kiertomatkaan Yhdysvalloissa. Ensimmäiset autonkorit Toyotalla tehtiinkin kaivamalla kuoppa maahan ja vasaroimalla metalli kuoppaan niin, että siitä saatiin ylösalaisen auton muotoinen. Koska autojen valmistuksesta tiedettiin niin vähän, Toyotalla ei kyetty antamaan täsmällisiä ohjeita ja jouduttiin luomaan uudenlaisia toimintamalleja. (Ward, 2007)

Toyotan kehittämä tuotantofilosofia alkoi kiinnostaa länsimaissa, kun havaittiin 1980-luvulla, että Toyotan kehittämillä menetelmillä autoja saatiin tuotettua suunnilleen puolella niistä työtunneista, joita kolme suurta yhdysvaltalaista autonvalmistajaa joutuivat käyttämään. Lean-nimeä käytettiin ensimmäisen kerran John Krafcikin artikkelissa Lean-tuotantojärjestelmän riemuvoitto. Lean-nimitys valittiin korostamaan sitä, kuinka Toyotan käyttämä tuotantotapa erosi Fordin käyttämästä puskuroidusta massatuotantotavasta. (Poppendieck ja Cusumano, 2012; Modig ja Åhlström, 2015)

Toyotalla kehitetyn järjestelmän ydinajatus on idea, että asiat tehdään vasta silloin, kun ne tarvitsee tehdä. Yhdysvaltalaisten autonvalmistajien tuotantoprosesseissa komponentteja tuotettiin varastoon valmiiksi tehtyjen tuotantosuunnitelmien mukaan. Toyotalla sen sijaan materiaalit ja komponentit imettiin tuotantojärjestelmän läpi jatkuvana virtana tarpeen mukaan. Edellisen vaiheen tuotantomäärät riippuvat siis seuraavan vaiheen tarpeesta. Tälläinen **imuohjaus** (*pull-manufacturing*) pienensi merkittävästi Toyotan väli- ja lopputuotevarastoja. Näin saatiin minimoitua varastot, joiden ajateltiin aiheuttavan kustannuksia ja piilottavan prosessien ongelmia. (Modig ja Åhlström, 2015)

Imuohjauksen käytännön toteutukseen käytetään "Kanban"-kortteja. Kukin Kanban-kortti antaa luvan siirtää tiettyä osaa tai tuotetta kortissa määritellyn määrän. Ilman korttia tuotetta ei ole lupa valmistaa. Korttien määrä on rajattu, eli se määrittää keskeneräisen tuotannon ja varastomäärien ylärajan. Imuohjaus on yksinkertaisinta toteuttaa silloin, kun tarve on kohtuullisen tasaista ja täydennykset ovat nopeita. Haasteellisemmaksi imuohjauksen käyttö tulee, jos kysynnän vaihtelu on suurta ja täydennysajat ovat pitkiä tai niitä ei voida ennustaa. (Logistiikan maailma, 2018)

Poppendieckin ja Cusumanon (2012) mukaan 90-luvun puolivälissä alettiin havaita samankaltaisuudet japanilaistyylisessä johtamisessa ja PC-tyylissä ohjelmistokehityksessä. Esimerkiksi Microsoftilla oli käytössä päivittäiset päivitykset, joissa kehittäjien täytyi pysähtyä korjaamaan virheitä päivittäin. Tämä oli samankaltaista Toyotan tavan kanssa pysäyttää tuotantolinjat heti, kun ongelmia havaittiin ja korjata ne välittömästi. Lean-ohjelmistokehitys kuitenkin esiteltiin ensimmäistä kertaa vasta Mary ja Tom Poppendieckin klassikkoteoksessa Lean Software Development: An Agile Toolkit (2003) Kirjassa esitettiin seitsemän Lean-ohjelmistokehityksen periaatetta. Poppendiec-

kit ovat tehneet listastaan myöhemmin päivitettyjä versioita saamiensa kokemusten perusteella ja listasta on olemassa ainakin kolme eri versiota (Poppendieck ja Poppendieck, 2003; Poppendieck ja Cusumano, 2012; Poppendieck.LLC, 2018). Tähän työhön valittiin versio, jota Poppendieck ja Cusumano käyttävät julkaisussaan. Verkkosivuilla olevan uusimman version katsottiin painottuvan vähemmän ohjelmistokehitykseen.

1. **Optimisoi kokonaisuus**
    - Kaikkien systeemin osien välinen synergia on avaintekijä koko systeemin onnistumiseen (Poppendieck.LLC, 2018). Lean-ohjelmistokehitys tulisi perustua erinomaiseen ymmärrykseen siitä, että asiakkaat tekevät ja miten tätä työtä voidaan auttaa ohjelmiston avulla (Poppendieck ja Cusumano, 2012).
2. **Eliminoi hukka**
    - Hukka on kaikki se, mikä ei joko tuota suoraan lisää arvoa asiakkaalle tai tietoa siitä, kuinka arvoa saataisiin tuotettua asiakkaalle nopeammin. Poppendieck ja Cusumano toteavat, että suurimpia syitä hukan syntymiselle ohjelmistokehityksessä ovat tarpeettomat ominaisuudet, menetetty tietämys, osittain tehty työ, siirtymiset ja useiden tehtävien tekeminen samaan aikaan (*multitasking*), puhumattakaan siitä, että 40 - 50 % ajasta käytetään virheiden etsimiseen ja korjaamiseen.
3. Rakenna **laatu tuotteen sisään**
    - Laadun rakentaminen liittyy hukan välttämiseen. Virheiden korjaaminen lisää aikaa ja saman koodin testaaminen uudelleen on tehotonta. Jokaisen inkrementin tulisi täyttää riittävät laatuvaatimukset.
4. **Opi jatkuvasti**
    - Loppujen lopuksi ohjelmistokehitys on tietämyksen luomista ja tämän tietämyksen jalostamista tuotteeksi. Leanissa tämä saadaan aikaiseksi kahdella eri tavalla. Ensinnäkin tulee harkita monia vaihtoehtoja hankalasti muutettavia päätöksiä tehtäessä. Tälläiset päätökset ovat esimerkiksi perusarkkitehtuuriin liittyvät valinnat ja ohjelmointikieli. Kriittisiä päätöksiä tulee viivyttää viimeiseen mahdolliseen hetkeen ja tehdä päätös sen hetkisen tiedon perusteella. Tätä kutsutaan usein myös "*opi ensin*" -periaatteeksi. Sellaisten ominaisuuksien suhteen, jotka eivät ole kriittisiä, käytetään toista lähestymistapaa. Hankitaan kykyjä ja tietoa juuri niin paljon, että voidaan aloittaa. Tämän jälkeen toimitetaan tuotteita usein ja käytetään oikeiden käyttäjien kokemuksia jatkokehityksen apuna. Tämä jatkuvan oppimisen taktiikka vähentää sellaisten ominaisuuksien tuottamista, joita asiakkaat eivät koe tarpeelliseksi.

5. **Toimita nopeasti**
    - Monissa Lean-projekteissa julkaisut tehdään säännöllisesti viikoittain, päivittäin tai jopa jatkuvasti. Näin vältetään mittavat regressiotestaukset, joita tarvittiin aikaisemmin julkaisujen yhteydessä. Nopeat julkaisut vaativat kuitenkin kunnolliset testit, joiden avulla huomataan suurin osa muutosten aiheuttamista virheistä. Nopeassa julkaisusyklissä ei enää ajatella ohjelmistokehitystä projektina, vaan on järkevämpää ajatella sitä jatkumona, jossa ohjelmisto suunnitellaan, kehitetään ja toimitetaan pienten muutosten jatkuvana virtana. Ohjelmisto ei siis tässä ajattelutavassa ole projekti, joka saatetaan valmiiksi tai edes sarja vuosittaisia versiopäivityksiä. Nopean toimituksen tulisi tapahtua koko tuotekehityksessä, ei pelkästään ohjelmiston kehittämisessä.
6. Sitouta **kaikki mukaan**
    - Lean-filosofia rohkaisee tekemään tiimityötä myös silloin, kun kyseessä on monitoimittajaprojekti. Kaikissa Lean-toteutuksissa ihmisille tulisi antaa valtuudet tehdä ratkaisuja, ryhmätyötä tulisi rohkaista ja päätösten tekeminen pitäisi siirtää mahdollisimman alhaiselle tasolle.
7. Tule **jatkuvasti paremmaksi**.
    - Lean-filosofiaan kuuluu ajatus, että vaikka mikä tahansa käytäntö näyttäisi toimivan hyvin toisissa tilanteissa, ne ovat harvoin parhaita ratkaisuja tämänhetkiseen ongelmaan. Lean rohkaiseekin käyttämään valmiita menetelmiä, kuten XP:tä ja Scrumia lähtökohtana ja muokkaamaan niistä omiin tarpeisiin toimivat versiot.

Taulukossa 3.3.1 näkyy kolme eri versiota. Käytännössä Poppendieckien Lean-filosofia on muuttunut vähemmän versiosta toiseen kuin miltä luetellut seitsemän periaatetta antavat ymmärtää. Joitain ajatuksia, jotka olivat aiemmin periaatteiden sisällä, on korostettu nostamalla ne periaatteeksi ja toisia periaatteita on vastaavasti sisällytetty muihin periaatteisiin. Lisäksi periaatteiden sisältöjä on ryhmitelty eri tavoin. Näyttää myös, että Poppendieckien web-sivuilla olevassa viimeisimmässä versiossa on haettu laajempaa näkökulmaa Lean-periaatteisiin, jotta ne toimisivat paremmin myös ohjelmistotuotannon ulkopuolella.

**Taulukko 3.3.1.** Lean-periaatteiden kolme eri versiota.

| **Poppendieck ja Poppendieck 2003)** | **Poppendieck ja Cusumo (2012)** | **Poppendieck.com (2018)** |
|---|---|---|
| Eliminoi jäte | Optimisoi kokonaisuus | Keskity asiakkaisiin |
| Edistä oppimista | Eliminoi hukka | Energisoi työntekijät |
| Päätä niin myöhään kuin mahdollista | Rakenna laatu tuotteen sisään | Poista kitkaa |
| Toimita tuote niin nopeasti kuin mahdollista | Opi jatkuvasti | Edistä oppimista |
| Anna tiimille valtaa | Toimi nopeasti | Lisää virtausta |
| Rakenna laatu tuotteen sisään | Sitouta kaikki mukaan | Rakenna laatu tuotteen sisään |
| Näe kokonaiskuva | Tule jatkuvasti paremmaksi | Tule jatkuvasti paremmaksi |

### 3.3.1 Ketterät menetelmät ja Lean

Voidaan kysyä, että mikä on Leanin ja ketterien menetelmien suhde toisiinsa. Joissakin lähteissä Lean luokitellaan ketterien menetelmien alaluokaksi. Toiset lähteet taas korostavat ketterien menetelmien ja Leanin eroja. Erityisesti ohjelmistokehityksen ulkopuolella ketterät toimintatavat ja Lean-tuotekehitys erotetaan toisistaan (Narasimhan, Swink ja Kim, 2006; Purvis, Gosling ja Naim, 2014) . Perinteisissä tuotelinjoissa ajatellaan, että Lean sopii hyvin tuotantoon, jonka kysyntä ja tarpeet ovat helposti ennustettavissa ja ketterät lähestymistavat tapauksiin, joissa kysyntää ja asiakkaiden tarpeita on vaikeampi selvittää (Naylor, Naim ja Berry, 1999). Täytyy muistaa myös, että ketterän kehityksen manifesti on alun perin ohjelmistokehitykseen suunniteltu ratkaisu, kun taas Lean-ohjelmistokehityksen juuret johtavat japanilaiseen autoteollisuuteen.

Kun verrataan ketterän manifestin kahtatoista periaatetta Poppendieckien seitsemään Leanin ohjelmistokehityksen periaatteeseen, huomataan, että ketterä kehitys on määritelmällisesti iteratiivista. Muutokset vaatimuksiin hyväksytään myös myöhään kehityksessä. Lean-periaatteet eivät vaadi iteratiivisuutta. Sen sijaan molemmat filosofiat ovat inkrementaalisia. Ketterän manifestin voidaan sanoa katsovan asioita enemmän ohjelmistokehittäjän näkökulmasta. Molemmat lähestymistavat pyrkivät antamaan tarpeelliset valtuudet kehittäjätiimille, välttämään hukkaa ja toimittamaan tuotteet mahdollisimman nopeasti.

Wangin ym. (2012) mukaan aluksi Lean käsitettiin yhtenä ketteränä menetelmänä. Hänen mukaansa Poppendieckit ajattelevat, että Lean tarjoaa ketterien menetelmien taustalla olevan teorian. Tämän perusteella voi ajatella, että Poppendieckit pitäisivät Leania korkeamman tason käsitteenä kuin ketterät menetelmät. Martin Fowler (2008) sanoo, että "Lean-teollisuuden ja ketterän ohjelmistokehityksen välillä oli yhteys

jo alun alkaen, koska monet ketterien menetelmien kehittäjät olivat saaneet vaikutteita Lean-tyylisestä valmistuksesta." Lisäksi kannattaa huomata, että Mary Poppendieck oli yksi Agile Alliancen perustajajäsenistä. Kun tutkitaan ketterän manifestin ja Poppendieckien Lean-periaatteita, huomataankin, että ketterä manifesti menee pidemmälle toteutusyksityiskohtiin esimerkiksi julistamalla, että ylitöitä ei pidä tehdä. Toisaalta ketterässä manifestissa on melkein tuplasti enemmän periaatteita kuin Poppendieckien Lean-periaatteissa, jolloin on tilaa mennä pidemmälle yksityiskohtiin. Toisaalta julkaisuissa (esimerkiksi Poppendieck ja Cusumano, 2012) Lean periaatteiden noudattamiseen annetaan hyvinkin yksityiskohtaisia ohjeita.

Termejä ketterät menetelmät ja Lean käytetään usein ristiriitaisesti ja sekavasti. Arkikielenkäytössä puhutaan usein Lean-menetelmistä ketterien menetelmien asemesta. Wangin ym. (2012) mukaan Lean nähdään kuitenkin yhä enemmän omana menetelmänään ja ohjelmistokehityksessä ollaan siirtymässä yhä enemmän "*ketterästä Leaniin*". Jotkut sanovat jopa, että ketterillä menetelmillä ei pystytä skaalaamaan ketteryyttä projekti- tai tiimitasolta organisaatiotasolle, vaan tähän tarvitaan siirtymistä Lean-menetelmiin (Smits, 2007, Wangin ym. (2012) mukaan). Toisaalta Lean niputetaan edelleen myös ketteräksi menetelmäksi. Jotta termistö ei menisi liian yksinkertaiseksi käytetään myös termiä lean/ketterät-menetelmät (*lean-agile methodology tai leagile methods*). Tämän työn tulosten tarkastelussa Lean-menetelmät niputettiin käytännön syistä kuulumaan ketteriin menetelmiin. Koska termien käyttö ei ollut haastatteluissa johdonmukaista, eikä voitu olla varmoja, laskivatko haastateltavat Lean-menetelmät ketterien menetelmien alle, tämä oli yksinkertaisin ratkaisu.

### 3.3.2 Kanban

Kanban on alunperin kortti, jolla saatiin säädettyä Toyotalla tuotannon määrää. Ohjelmistotuotannossa Kanban-kortteja on käytetty tehtävien tilan visualisointiin. Kanban-taulussa on omat sarakkeensa tehtävän eri valmiusasteille. Kanban-kortin paikka muuttuu Kanban-taulussa sen mukaan, miten siihen liittyvän tehtävän tila muuttuu. Nykyisin Kanbanilla voidaan tarkoittaa kuitenkin myös Leaniin pohjautuvaa prosessimallia. Al-Baikin (2015) mukaan David Anderson (2010) on muokannut vuodesta 2003 saakka Kanbanista ohjelmistokehityksen menetelmää, joka on Kanban-taulua laajempi käsite. Kanban-menetelmä on inkrementaalinen ja evolutionäärinen Lean-filosofiaa tukeva kehitysmenetelmä, jossa Kanban-taulua voidaan käyttää työkaluna (Al-Baik ja Miller, 2015; Ahmad *ym.*, 2018).

Anderson määrittelee Kanban-menetelmän onnistuneeseen käyttöön viisi eri elementtiä:

1. työn kulun visualisointi,
2. keskeneräisten tehtävien (*WIP = work in process*) määrän minimointi,

3. virtauksen hallinta,
4. täsmälliset menettelytavat ja
5. palautesyklien luominen (Andersson 2010 Al-Baikin ja Millerin 2015 mukaan).

Työn kulun visualisointi tarkoittaa sitä, että tehtäville annetaan tila. Tämä tila muuttuu työn edetessä ja tehtävän kulkiessa organisaatiossa. Yksinkertainen tapa visualisoida työn kulku on käyttää fyysistä tai virtuaalista taulua, Kanban-taulua, jossa tehtävät esitetään kortteina. (Ahmad, Markkula ja Oivo, 2013; Ahmad *ym.*, 2018)

Keskeneräisten tehtävien määrä tulee olla rajoitettu kaikissa mahdollisissa työn kulun vaiheissa. Anderson sanoo tiukasti, että jos tehtävien määrää eri vaiheissa ei ole rajoitettu ja jos seuraava vaihe ei viesti edelliselle vaiheelle imeäkseen uuden tehtävän työn alle, prosessi ei ole Kanban-menetelmän mukainen (Anderson, 2010). Käytännössä määrä voidaan rajoittaa niin, että annetaan Kanban-taulun jokaiselle sarakkeelle yläraja, kuinka monta korttia sarakkeessa voi olla.

Virtauksen hallintaan on olemassa erilaisia tekniikoita, kuten Kanban-taulu, kumulatiiviset vuokaaviot ja edistymiskäyriä. Virtauksen laatua mitataan neljällä eri metriikalla. Ensinnäkin on **jonon koko**: mitä suurempi jonon koko on, sitä suurempi todennäköisyys on, että syntyy hukkaa kontekstien vaihdoista, tehtävien siirtyessä henkilöiltä toisille jne. Toinen tekijä on **läpimenoaika**, joka syntyy siitä, kuinka nopeasti työ virtaa systeemin läpi. Pitkä läpimenoaika kertoo, että prosessissa on hidastavia tekijöitä, jotka vähentävät aikaansaatuja tuloksia. **Kiertoaika** kertoo, kuinka kauan yksittäisellä tehtävällä kestää kulkea tietyn vaiheen tai tiettyjen vaiheiden läpi. Lopuksi **toimitusaika** kertoo, kuinka kauan yksittäiseltä tehtävältä kestää siirtyä koko systeemin läpi pyynnöstä tai ideasta tuotantojulkaisuun asti. (Power ja Conboy, 2015; Ahmad *ym.*, 2018)

Täsmälliset menettelytavat tarkoittavat sitä, että tehtävälle on määritelty täsmälliset kriteerit, koska se saa poistua tietystä vaiheesta ja koska se saa siirtyä seuraavaan vaiheeseen. Täsmälliset menettelytavat auttavat organisaatioita tutkimaan muutosten vaikutuksia prosesseissa. Viimeinen Kanban-menetelmän elementti on palautesyklit. Palautesyklien luomisessa voidaan käyttää erilaisia jatkuvan parantamisen malleja, kuten **kapeikkoajattelua** (*TOC = theory of constraints*) tai **systeemiajattelua** (*systems thinking*) (Ahmad *ym.*, 2018). Kapeikkoajattelun ideana on tunnistaa prosessin pullonkauloja ja parantaa prosessia pyrkimällä poistamaan tai kiertämään näitä hidasteita (Wikipedia, 2018b). Systeemiajattelussa puolestaan keskitytään ymmärtämään monimutkaisia kokonaisuuksia sen kautta, että kuinka eri komponentit ja niiden ominaisuudet vaikuttavat koko systeemin toimintaan (Hyytiälä, 2011).

## 3.4 SAFe

Edellisissä kappaleissa prosessimalleja ja käyttäjäkokemuksen suunnittelua on tarkasteltu yhden projektitiimin tasolla. Esimerkiksi Scrum ei määrittele, kuinka sitä käytetään

suurissa projekteissa tai kuinka vaatimuksia hallitaan portfoliotasolla. Skaalautumiseen on kehitetty useita eri ratkaisuja. Ebert ja Paasivaara (2017) listaavat julkaisussaan viisi mallia, joita nykyisin käytetään viimeaikaisten kyselyiden ja heidän oman kokemuksensa perusteella: 1) **Scaled Agile Framework** eli *SAFe*, 2) **Scrum of scrums**, 3) **Large-Scale-Scrum** eli *LeSS*, 4) **Diciplined Agile Delivery** eli *DAD* ja 5) **Lean Scalable Agility for Engineering** eli *LeanSAFE*. Ne eroavat niin laajudeltaan ja sillä, vaativatko ne tietyn ketterän menetelmän käyttöä. Scrum of Scrums ei ole varsinainen viitekehys Scrumin skaalaamiseen vaan se on ylimääräinen päivittäinen tapaaminen, jossa useista Scrum-tiimeistä valitut edustajat keskittyvät koordinoimaan ja synkronoimaan eri tiimejä (Agile Alliance, 2018a).

SAFe on menetelmistä suosituin (Ebert ja Paasivaara, 2017). Se on kokonaisvaltainen viitekehys, joka soveltuu niin pienemmille 50-125 henkilön projekteille kuin projekteille, joissa työskentelee tuhansia ihmisiä. Se on suuri kokonaisuus, joka itsessään riittäisi yhden opinnäytetyön aiheeksi. Työn empiirisessä osassa havaittiin, että myös pienempiin, yhden tiimin projekteihin oli otettu mukaan ominaisuuksia SAFesta, joten SAFe päätettiin käsitellä tässä pintapuolisesti.

SAFe määrittelee projektin toiminnot neljällä eri hierarkiatasolla, jotka ovat 1) **portfolio**, 2) **arvovirrat**, 3) **hanke** ja 4) **tiimi**. Portfoliotasolla hallitaan pitkän aikavälin suunnitelmaa sekä arvovirtoja, joista koostetaan liiketoiminta- ja arkkitehtuuriteemat ohjaamaan pitkän aikavälin rahankäyttöä. Arvovirrat ovat pitkän aikavälin strategisia teemoja, jotka otetaan käyttöön vasta todella isoissa, satojen ihmisten hankkeissa. Hanketasolla ohjataan hankkeiden kehitystä portfoliotasolta tulevan priorisoinnin mukaan. Tiimitasolla seurataan ketterän kehityksen toimintatapoja ja tuotetaan säännöllisesti niin tiimitasoisia kuin tiimien työt yhdistäviä julkaisuja.(Hietaniemi, 2016) SAFesta on saatavissa neljä erilaista konfiguraatiota, jotka vaihtelevat laajuudeltaan. Eri konfiguraatioihin otetaan mukaan SAFe-mallin eri hierarkiatasoja. Esimerkksi SAFen yksinkertaisimpaan versioon kuuluvat vain tiimi ja tuotetasot. Portfoliotasoon kuuluu puolestaan tiimi- ja tuotetasojen lisäksi hierarkiataso.

Koko SAFe tähtää siihen, että kehitystiimit pystyvät toimittamaan jatkuvasti ratkaisuja, jotka tuottavat arvoa asiakkaalle ja lopulta auttavat asiakasta saavuttamaan tavoitteensa. Ratkaisut toimitetaan arvovirtojen avulla. **Arvovirrat** käynnistyvät jostain merkittävästä tapahtumasta, esimerkiksi asiakkaan tekemästä tilauksesta ja ne päättyvät siihen, kun arvo on toimitettu asiakkaalle. Arvovirta on ne toimenpiteet, jotka toimittajan organisaatiossa tehdään, jotta ratkaisut saadaan tuotettua asiakkaalle. Niitä käytetään määrittelemään portfoliotason liiketoimintatavoitteet ja siihen, että saadaan organisoitua tiimien tiimit eli toteutusjunat (*ART = Agile Release Trains* ) tuottamaan arvoa nopeammin. (Scaled Agile, 2018)

**Toteutusjunat** organisoivat tiimit toimimaan yhteisen bisnes- ja teknologiatavoitteen eteen. Jokainen toteutusjuna määrittelee ja ylläpitää kehitysputkea mahdollisimman itsenäisesti. Kehitysputken toimintaan kuuluu jatkuva tutkiminen, jatkuva integraatio ja

jatkuva toimitus. Jatkuvaan tutkimiseen kuuluu markkinan ja käyttäjän tarpeiden jatkuva tutkiminen ja vision, *roadmap*-suunnitelman ja tarvittavien ominaisuuksien määrittäminen. Jatkuva integraatio on puolestaan kehitysjonosta otettavien ominaisuuksien kehitystä, testausta, integraatiota ja validaatiota. Jatkuvassa toimituksessa testiympäristössä validoidut ominaisuudet viedään tuotantoympäristöön. **Hankkeen inkrementti** (*PI = program increment*) on aikaikkuna, jonka aikana toteutusjuna tuottaa suurehkon toimivan ja testatun kokonaisuuden. Hankkeen inkrementit aloitetaan PI-suunnittelusessiolla. Sen jälkeen aloitetaan iteraatiot. Tyypillisessä PI:ssä on neljä kehityssprinttiä ja sen jälkeen pidetään vielä yksi innovaatio- ja suunnitteluiteraatio. Hankkeen inkrementti on siis vastaava käsite toteutusjunalle kuin sprintti on Scrum-tiimille. (Scaled Agile, 2018)

SAFen korkeimmalla abstraktiotasolla oleva työjono on portfolio-työjono. Se sisältää **kehitysaihioita** eli niin kutsuttuja **eeppisiä ominaisuuksia** (*epic*). Asiakkaan visiot tai tilaamat ratkaisut jaetaan pienempiin osiin toteuttamista varten. Eeppiset ominaisuudet ovat kokonaisuuksia, jotka ovat tarpeeksi isoja vaatiakseen analyysiä, pienimmän toimivan tuotteen (*MVP = minimum viable product*) määrittelemistä ja taloudellisen hyväksynnän ennen toteutusta. Eeppisten ominaisuuksien toteutus venyy useammalle hankkeen inkrementille ja ne toteutetaan noudattaen Lean startup –tyylistä kehitä-mittaa-opi–sykliä. Eeppisiä ominaisuuksia hallitaan portfolion Kanban-taululla. (Nitor, 2018; Scaled Agile, 2018)

Eeppiset ominaisuudet pilkotaan edelleen toiminnallisuuksiksi ja kyvykkyyksiksi. **Toiminnallisuudet** ovat niin pieniä, että yksi toteutusjuna pystyy tuottamaan sen yhdessä hankkeen inkrementissä. **Kyvykkyydet** ovat isompia kokonaisuuksia, joiden toteuttamiseen tarvitaan tyypillisesti useampi toteutusjuna ja ne pilkotaan edelleen ominaisuuksiksi. Ominaisuudet ja kyvykkyydet priorisoidaan omissa hanketason kehitysjonoissaan. (Nitor, 2018; Scaled Agile, 2018)

Kehitystiimit ovat ketterän kehityksen mukaisia monialaisia tiimejä, jotka saavat järjestää työskentelynsä ketterän kehityksen puitteissa vapaasti. Ne toimivat kahden viikon mittaisissa sprinteissä, jotka tuottavat toimivia lisäarvoatuottavia ominaisuuksia ohjelmistoon. Sprintit kootaan puolestaan hankkeen inkrementeiksi, kuten edellä on kerrottu. Tiimit käyttävät omaa työjonoaan, joka on tuoteomistajan vastuulla. Työjonossa on ominaisuuksista pilkottuja käyttäjätarinoita ja mahdollistajia. **Käyttäjätarinat** ovat yksinkertaisia kuvauksia pienestä osasta haluttua toiminnallisuutta. Ne kerrotaan yleensä käyttäjän näkökulmasta ja käyttäjän kielellä. **Mahdollistajat** puolestaan kuvaavat kehitystyötä, jota tarvitaan mahdollistamaan tulevien liiketoiminnan toiminnallisuuksien kehittämisen. Ne voivat olla esimerkiksi arkkitehtuurisia parannuksia. (Hietaniemi, 2016; Nitor, 2018; Scaled Agile, 2018)

Se, että SAFen ominaisuuksia on otettu mukaan myös pienempiin projekteihin, kertoo siitä, että pelkästään ketterien tiimien toimintaan keskittyvät menetelmät, kuten Scrum, eivät määrittele tarpeeksi asioita vastatakseen projekti- tai hanketason kysymyksiin. Toisaalta SAFea on kritisoitu liiasta raskaudesta ja monimutkaisuudesta. Se yrittää

sisällyttää itseensä kaikki hyvät käytännöt (Ebert ja Paasivaara, 2017). Tosin kannattaa huomata, että SAFe ei ota kantaa palvelumuotoilun toteuttamiseen ja LeanUX lisättiin vasta SAFe-versioon 4.5 (tuorein versio kirjoitushetkellä 4.6) (Scaled Agile, 2017). Ebertin ja Paasivaaran mukaan jotkut jopa sanovat, että SaFe lisää byrokratiaa muodostuen uudeksi vesiputoukseksi. LeSS ja LeanSAFE määrittelevät SaFeen verrattuna paljon vähemmän ja jättävät enemmän tilaa omille ratkaisuille.

# 4 KÄYTTÄJÄKESKEISYYDEN YHDISTÄMINEN KETTERIIN MENETELMIIN

Ketterä kehitys ja ihmiskeskeinen suunnittelu ovat eri lähtökohdista lähteviä lähestymistapoja ohjelmistokehitykseen. Niillä on yhteisiä periaatteita, kuten iteratiivinen suunnittelu, käyttäjien osallistuttaminen, jatkuva testaus ja prototypointi (Ardito *ym.*, 2017). Kuitenkin ketterissä menelmissä lähdetään ajatuksesta, että ohjelmiston arvo luodaan asiakkaalle toiminnallisuuksilla (Brhel *ym.*, 2015). Kehityksessä ei välttämättä huomioida sitä, että kuinka helposti ohjelmistoa voidaan käyttää. Käyttäjäkeskeisessä suunnittelussa sen sijaan tämä on luonnollisesti toimintatapojen keskeisenä ideana. Ketterien menetelmien ja käyttäjäkeskeisten menetelmien erona on myös se, että ketterät menetelmät keskittyvät tiimin toimintaan. Käyttäjäkeskeisten menetelmien keskiössä on käyttäjän tarpeiden selvittäminen. Ehkä se, että menetelmien fokus on hieman erilainen, on vaikeuttanut menetelmien yhdistämistä toisiinsa. Ketterissä menetelmissä ei oteta yleensä kantaa siihen, kuinka käyttäjäkokemus tulisi suunnitella käytettäessä menetelmää. Reilun viimeisen vuosikymmenen aikana onkin tehty ahkerasti tutkimusta siitä, kuinka nämä kaksi metodologiaa tulisi yhdistää toisiinsa.

## 4.1 Havaitut haasteet

Düchting ym. (2007) tarkastelivat teoriatasolla kuinka hyvin ketterät menetelmät ovat yhdistettävissä käyttäjäkeskeisiin menetelmiin. He jakoivat ihmiskeskeisen suunnittelun vaatimukset kolmeen eri kategoriaan Zimmermanin ja Grötzbachin (2007) luoman jaottelun mukaan. Tämän jälkeen he analysoivat, miten hyvin Scrum ja XP-menetelmissä huomioidaan käytettävyyteen, työn kulkuun ja käyttöliittymään liittyvien vaatimusten luominen ja niiden toteutumisen arviointi. Düchting ym. havaitsivat, että Scrumissa ja XP:ssä oli puutteita kaikissa näissä osa-alueissa. He nostavat esille muun muassa sen, että oikeat käyttäjät eivät välttämättä pääse osallistumaan prosessiin, koska asiakkaan vastuut on määritelty Scrumissa ja XP:ssä niin laajasti. Asiakkaan tehtäviin kuuluu ketterissä menetelmissä, muiden tehtävien ohella, vaatimusten toimittaminen, käyttäjätarinoiden kirjoittamiseen osallistuminen ja priorisointi, sekä ohjelmiston testaus (Sohaib ja Khan, 2010). Toinen Düchtingin ym oleellinen havainto on, että ketterissä menetelmissä on vaarana kokonaiskuvan menettäminen. Näitä samoja havaintoja on tehty myöhemmin empiirisistä tutkimuksista.

Sohaib ja Khan (2010) keskittyvät katsausartikkelissaan analysoimaan erityisesti ketterien menetelmien ja käyttäjäkeskeisten menetelmien välisiä jännitteitä. He jakoivat kirjallisuudesta löytämänsä ongelma-alueet neljään kategoriaan: 1) **keskittyminen asiakkaaseen vai keskittyminen käyttäjään**, 2) **toimiva sovellus vai käytettävä sovellus**, 3) **suunnittelu vasta tarvittaessa vai etukäteissuunnittelu**, 4) **yksikkötestaus vai käytettävyystestaus**. Kuusisen (2015a) väitöskirjan mukaan kolme selkeästi useimmin esiintyvää ongelmaa ketterässä UX-suunnittelussa olivat 1) **asiakkaan ja käyttäjän tarpeiden ymmärtäminen ja täyttäminen**, 2) **vision säilyttäminen** ja 3) **UX-suunnittelijoiden ja kehittäjien erilaisten toimintatapojen koordinoiminen**.

Salah ym. (2014) katsausartikkelissa on tehty samansuuntaisia havaintoja. Salah ym. mukaan ketterät menetelmät keskittyvät pelkästään tuotteen toiminnallisuuksiin, mistä seuraa, että suunnittelulle, käyttäjätutkimukselle ja suunnitelman viestimiseen tarvittavien dokumenttien tekemiselle ei varata aikaa projektissa. Vastaava ongelma havaittiin ominaisuuksien priorisoinnissa työjonossa. Huomio kiinnittyi herkästi toimivien kokonaisuuksien valmiiksi saamiseen ja käytettävyyteen. Käyttäjäkokemukseen liittyvät ominaisuudet jäivät toteuttamatta. Lisäksi ketterän kehityksen periaate pyrkiä maksimoimaan se työ, jota ei ole tehty, ei aina sovi yhteen toimivan käyttöliittymän suunnittelun kanssa.

Vaikka käyttäjäkokemuksen merkitys olisikin ymmärretty projektissa, tarpeeksi resursseja sen suunnitteluun ei oltu välttämättä varattu. Salah ym. nostaa esille, että etenkin pienemmissä organisaatioissa UX-asiantuntijat työskentelevät yleensä useammassa projektissa tai osa-aikaisesti. UX-suunnittelijoiden saattoi olla vaikea ehtiä osallistua menetelmään kuuluviin tapaamisiin, eikä heillä ollut välttämättä tarpeeksi aikaa saattaa suunnittelutyötä loppuun. Lisäksi jatkuvat kontekstinvaihdokset aiheuttivat merkittävän laskun tuottavuudessa. Se, että ketterissä menetelmissä tuote tehdään pala palalta, voi johtaa muutenkin käyttäjäkokemuksen kannalta epäyhtenäiseen tilkkutäkkiin, josta puuttuu kokonaisvaltainen näkemys ja rakenne. Salah ym. mukaan on myös haasteellista paloitella suunniteltavat kokonaisuudet sopiviksi paloiksi. Palojen täytyy olla oikean kokoisia, jotta ne voidaan suunnitella hyvin iteraation aikana. Lisäksi palat täytyy suunnitella loogisessa järjestyksessä, koska ne ovat riippuvaisia toisistaan. Useat projektit vaikeuttavat edelleen tämän kokonaiskuvan luomista. Memmelin ym. (2007) mukaan iteratiivinen suunnittelu voikin johtaa käyttöliittymään, josta puuttuu yhdenmukaisuus sekä yhdenmukainen käsitteellinen, navigationaalinen ja sisällöllinen malli.

Kehittäjät ja UX-suunnittelijat työskentelevät tyypillisesti rinnakkaisissa sprinteissä (*parallel sprint*). Vaarana on, että kehittäjät tekevät itsenäisesti ratkaisuja, jotka heikentävät käyttöliittymän yhdenmukaisuutta (Kuusinen, 2015b). Yhdistettäessä ketterää kehitystä käyttäjäkeskeisiin menetelmiin onkin tärkeä huolehtia jatkuvasta kommunikaatiosta kehittäjien ja UX-asiantuntijoiden välillä niin, että tieto kulkee sujuvasti. Näin

vältetään myös viivästyksiä ja pullonkauloja kehitysprojektissa. Toisaalta Salahin ym. mukaan ketterän kehityksen tiukka aikataulu saattaa aiheuttaa ongelmia käyttäjätestien toteutukselle. Heidän mukaansa yksi ongelma on se, että käyttäjien värväämineen testeihin oikean aikaan saattaa olla haastavaa. Myös käyttäjätestien tuloksien vaatimia muutoksia voi olla vaikeaa ajoittaa tuleville sprinteille. Uskon, että paine lyhentää edelleen julkaisusyklejä todennäköisesti tulee hankaloittamaan edelleen käyttäjätestien toteuttamista.

Myöskään etukäteen tehtävälle suunnittelulle ei välttämättä varata tarpeeksi aikaa (Salah, Paige ja Cairns, 2014). Kuitenkin Da Silvan ym. (2011) aineistotutkimuksessa miltein kaikki läpi käydyt tutkimukset suosittelivat vain pientä määrää etukäteissuunnittelua. Kuusisen (2015a) mukaan tämä saattoi kuitenkin johtua siitä, että etukäteissuunnittelun suuren määrän katsottiin olevan ketterien periaatteiden vastaisia, ei niinkään siitä, että pieni määrä etukäteissuunnittelua olisi optimaalinen tapa yhdistettäessä käyttäjäkeskeisiä menetelmiä ketteriin.

Sinänsä monet kysymykset, joista keskustellaan, kun mietitään käyttäjäkeskeisten menetelmien yhdistämistä ketteriin, ovat samoja kuin ne, jotka ovat aiheuttaneet päänvaivaa siirryttäessä suunnittelupohjaisista malleista ketteriin. Miten hallita kokonaiskuvaa, kun käsitys suunniteltavasta ohjelmistosta täydentyy ja muuttuu jatkuvasti? Miten yhdistää testaus ketterän kehityksen iteratiiviseen ja inkrementaaliseen luonteeseen? Esimerkkinä voidaan käyttää etukäteissuunnittelun määrää. Käytettäessä ketteriä menetelmiä yritetään tasapainoilla niin, että päätökset tehdään mahdollisimman myöhään, mutta tarpeeksi aikaisin.

Esimerkiksi Waterman ym. (2015) ovat tutkineet ohjelmistoarkkitehtuurien suunnittelua ketterässä ympäristössä. Jos etukäteissuunnittelua ei tehdä tarpeeksi, kokonaisuutta on vaikeampi hallita. Seurauksena päädytään joko koodaamaan uudelleen paljon toiminnallisuuksia, jotka on jo toteutettu tai vaihtoehtoisesti kokonaisuudesta ei tule yhtenäistä. Jos suunnittelua tehdään liikaa etukäteen, päädytään samoihin ongelmiin, joiden takia vesiputousmallista on haluttu siirtyä ketteriin ja menetetään se hyöty, joka saadaan ketteristä menetelmistä. Täysin vastaaviin haasteisiin törmätään pohdittaessa käyttäjän kokemukseen vaikuttavien ratkaisujen tekemistä.

Vaikka haasteita on, Brhelin ym. (2015) mukaan ketterillä menetelmillä ja ihmiskeskeisellä suunnittelulla on kuitenkin yhteisiä piirteitä, jotka auttavat näiden menetelmien yhdistämistä toisiinsa. Yhdistämisessä on heidän mukaansa käytetty erilaisia lähestymistapoja. On yhdistetty **käytettävyyteen liittyviä menetelmiä ketteriin menetelmiin**, **ketterien menetelmien piirteitä ihmiskeskeiseen suunnitteluun** ja lisäksi on ketterät menetelmät ja käyttäjäkeskeisen suunnittelun yhdistäviä **hybridimalleja**, joita kutsutaan käyttäjäkeskeisen ketterän ohjelmistokehityksen menetelmiksi. (Brhel *ym.*, 2015) .

## 4.2 Parhaaksi havaitut käytännöt

Brhel ym. (2015) analysoivat laajassa katsausartikkelissaan käyttäjäkeskeisen suunnittelun yhdistämistä ketteriin menetelmiin. He päätyvät suosittelemaan viittä periaatetta:

1. **erillinen** tuotteen **suunnittelu ja luominen**,
2. **iteratiivinen ja inkrementaalinen** suunnittelu ja kehitys,
3. ominaisuuksien luominen **rinnakkaisilla** poluilla,
4. **sidosryhmien jatkuva osallistuminen** kehitykseen ja
5. kommunikaatio erilaisten **tuotoksien (artefaktien)** avulla.

Ensinnäkin he ovat sitä mieltä, että tuotteella tulee olla erillinen, rajoitettu suunnitteluvaihe ennen tuotteen toteuttamista. Se on tärkeää, jotta saadaan luotua yhdenmukainen ja johdonmukaisesti toimiva käyttöliittymä sekä navigaatiorakenne. Brhel ym nostavat esille myös sen, että ketterät menetelmät, toisin kuin ihmiskeskeinen suunnittelu, eivät anna eväitä siihen, millainen tuote kokonaisuudessaan tulisi rakentaa. (Brhel *ym.*, 2015) Yhdistämällä siis käyttäjäkeskeistä suunnittelua ketteriin menetelmiin saadaan työkaluja tuotteen suunnitteluvaiheeseen. Samalla saadaan laajennettua sitä osaa, jonka menetelmä ohjelmistotuotantoprojektista kattaa.

Seuraava Bhrelin ym. (2015) periaate on, että kehityksen tulee olla inkrementaalista ja iteratiivista. Ohjelmiston kehittäminen iteratiivisesti luo mahdollisuuden arvioida kokonaisia toiminnallisuuksia ja saada niistä palautetta nopeasti. Palautetta pystytään käyttämään ominaisuuden parantamiseen seuraavassa iteraatiossa. Toiminnon iteratiivista parantamista ja lopputuloksen arvioimista jatketaan, kunnes käyttäjän tarpeet saavutetaan. (Brhel *ym.*, 2015) Inkrementaalinen ja iteratiivinen kehitys mahdollistaa tuotteen testaamisen kehityksen aikana samoin kuin jatkuvan käyttäjätestauksen ja käytettävyysparannusten tekemisen (Lee, McCrickard ja Stevens, 2009) .

Seuraava ketterän kehityksen ja ihmiskeskeisen suunnittelun yhdistävien mallien periaate on Brhelin ym. (2015) mukaan suunnittelun ja toteutuksen tekeminen erillisissä rinnakkaisissa säikeissä. Käytännössä siis sprintillä toteutettavat toiminnallisuudet on suunniteltu aikaisemmalla sprintillä. Samalla, kun suunnitellaan seuraavalla sprintillä toteutettavia toiminnallisuuksia, testataan myös edellisen sprintin aikana toteutettujen asioiden toimivuutta. **Suunnittelu ja toteutus voidaan tehdä käytännössä täysin erillisissä tiimeissä**. **Vaihtoehtoisesti voidaan käyttää myös yhtä tiimiä**, jossa on osaamista molemmista osa-alueista. (Brhel *ym.*, 2015) Jälkimmäinen tapa on enemmän ketterän kehityksen periaatteiden mukainen. Esimerkiksi Scrumia noudatettaessa kaiken ohjelmiston kehityksessä tarvittavan osaamisen tulisi olla itse kehitystiimissä.

On olemassa useita erilaisia versioita tästä "yksi sprintti edellä" -mallista, kuten Millerin (2005), Syn (2007), Fox ym. (2008) ja da Silvan (2012) versiot. Syn malli, josta puhutaan lisää seuraavaksi, lienee malleista tunnetuin ja on toiminut pohjana useille myöhemmille versioille mallista. On olemassa myös julkaisuja, joissa ehdotetaan, että UX-työ voitaisiin tehdä myös kaksi (Budwig, Jeong ja Kelkar, 2009) tai kolme (Illmensee ja Muff, 2009) sprinttiä ennen toteutusta.

Neljäs Brhelin ym. esittämä periaate on, että sidosryhmät ovat mukana kehityksen joka vaiheessa aktiivisesti. Sidosryhmien keskeinen merkitys onkin yhteistä ketterän kehityksen menetelmille ja käyttäjäkeskeiselle suunnittelulle. Ongelmana on ollut lähinnä se, että ketterissä menetelmissä asiakasta ja loppukäyttäjää ei ole selkeästi erotettu toisistaan. Ei ole tavatonta, että yksi henkilö, joka voi olla esimerkiksi asiakkaan tuotepäällikkö tai sovellusalueen asiantuntija, on kehitysprojektissa sekä asiakkaan että käyttäjän roolissa. Todelliset loppukäyttäjät eivät näin pääse osallistumaan prosessiin. Kuitenkin käyttäjäkeskeisissä menetelmissä erilaisten loppukäyttäjäryhmien kuuleminen on hyvin tärkeässä roolissa. (Brhel *ym.*, 2015)

Viides ja viimeinen periaate on se, että projektia dokumentoidaan mahdollisimman vähän. Dokumentaation sijaan apuna käytetään erilaisia tuotoksia eli artifakteja. Suunnittelun edetessä siirrytään tyypillisesti karkeammista mallinnoksista, kuten rautalankamalleista (*wireframes*), käyttäjäkertomuksiin ja edelleen tarkempiin ja lähempänä lopputulosta oleviin prototyyppeihin. Nämä tuotokset tukevat luovaa prosessia, mutta ne ovat tärkeitä myös kommunikaatiolle niin tiimin sisällä kuin muiden sidosryhmien kanssa. (Brhel *ym.*, 2015)

Myös Larusdottir ym. (2017) antavat tuoreessa katsausartikkelissaan neuvoja siihen, miten ketterät menetelmät ja käyttäjäkeskeinen suunnittelu tulisi yhdistää. Ensinnäkin Larusdottirin ym. mukaan kaikille projektin rooleille tulee määritellä, mitkä heidän vastuunsa ovat käytettävyyden ja käyttäjäkokemuksen suhteen. Vastuu käyttäjäkokemuksesta ei siis heidän mukaansa ole välttämättä pelkästään UX-suunnittelijoilla, vaan myös kehittäjillä, Scrum-masterilla ja tuoteomistajalla. Niiden henkilöiden, jotka vastaavat käyttäjäkokemuksesta tulisi tavata oikeita käyttäjiä säännöllisesti, vähintään kerran sprintin aikana. Kehitystiimin tulisi myös käyttää useita eri kommunikaatiovälineitä yhteydenpitoon käyttäjien kanssa. Käyttäjiltä tulevat muutospyynnöt tulisi priorisoida. Näin saadaan nostettua käyttäjien näkökulman painoarvoa suhteessa muihin näkökulmiin.

Toinen teema, jonka Larusdottir ym. nostavat esille on toimivan lopputuotteen tekeminen. Varhaisessa vaiheessa tulee luoda visio tuotteen käytettävyydestä ja käyttäjäkokemuksesta. Visiota tulee tarkistaa työn edetessä. Lisäksi käytettävyyden ja käyttäjäkokemuksen toteutumiselle määritetään mitattavat tavoitteet. Käyttäjien kanssa tulee arvioida säännöllisesti, että täyttääkö ohjelmisto käyttäjien vaatimukset ja kuinka arvokas ohjelmisto on heille. Tämä arviointi tehdään vähintään joka toisessa sprintissä. Larus-

dottir ym. esittää lisäksi, että tuoteomistajan, scrum-masterin ja tiimin tulee suunnitella, kuinka arviointien tuloksia tutkitaan. Saatujen tulosten tulee johtaa seuraamuksiin ja arviointien perusteella tehtyjen muutosten toteutumista pitää seurata. Larusdottir ym. suosittelevat vielä, että retrospektiiveihin valitaan teemat ja että käytettävyyden ja käyttäjäkokemuksen parantamisen tulisi olla yksi näistä teemoista. (Larusdottir, Gulliksen ja Cajander, 2017)

## 4.3 Suunnittelu yksi sprintti edellä

Øvad (2014) listaa neljä mallia käyttäjäkeskeisen suunnittelun ja ketterän kehityken yhdistämiseen: 1) **suunnittelu yksi sprintti edellä**, 2) **satelliittilähestymistapa**, 3) **uScrum** ja 4) **LeanUX**. Suunnittelua yksi sprintti edellä käydään läpi tässä aliluvussa ja LeanUX aliluvussa 4.4. Satelliittilähestymistavassa käytetään UX-asiantuntijaa, joka toimii välimiehenä kehitystiimin ja UX-tiimin välillä. Satelliittihenkilö keskittyy UX-tiimin ja kehitystiimin väliseen suhteeseen. Loput UX-tiimistä tekevät normaalia UX-työtä, kuten käyttäjätestejä ja näyttökuvia tai rautalankamalleja. UScrum-mallissa kehittäjät ja UX-suunnittelija ovat samassa Scrum-tiimissä. Scrumiin kuitenkin lisätään tavallisen tuoteomistajan lisäksi käytettävyys-tuoteomistaja, joka on vastuussa koko tuotteen käyttäjäkokemusvisiosta.

”*Yksi sprintti edellä*” -malli on tapa yhdistää ketterä kehitys käyttäjäkeskeiseen suunnitteluun. Palvelumuotoilun liitettämiseen mukaan projektin käytäntöihin malli ei ota kantaa. ”Yksi sprintti edellä” -mallissa ohjelmistokehitys ja UX-suunnittelu tehdään toisistaan erillään erillisissä säikeissä. Idea on, että sprintissä toteutettavat ominaisuudet suunnitellaan yhtä sprinttiä aikaisemmin. Sprintissä toteutetut ominaisuudet testataan seuraavassa UX-sprintissä. Sprintin aikana voidaan tehdä myös esitutkimusta tulevia sprinttejä varten. Käytännössä UX-asiantuntija tekee kerrallaan neljään eri sprinttiin kuuluvia asioita. Sprintillä 1) **testataan edellisellä sprintillä toteutetut asiat**, 2) **autetaan kuluvan sprintin toteutuksessa**, 3) **tehdään suunnitelmat seuraavaa sprinttiä varten** ja 4) **tehdään esitutkimusta** seuraavan sprintin jälkeistä sprinttiä varten. UX-suunnittelijat voivat olla joko osa kehitystiimiä tai muodostaa oman UX-tiiminsä. Ei ole ratkaisevaa, onko UX-suunnittelija sijoitettu organisatorisesti eri tiimiin vai samaan tiimiin. Oleellista on se, että kuinka kiinteästi projektin henkilöt tekevät yhteistyötä. Esimerkiksi Da Silva (2012) määrittää, että erillisen UX-tiimin tulisi työskennellä kiinteästi kehitystiimin kanssa. Gothelf (2013) puolestaan toteaa, että monet tiimit ovat tulkinneet väärin ”yksi sprintti edellä” -mallin ja muokanneet siitä jonkinlaisen minivesiputouksen, jossa kehittäjät ja UX-suunnittelijat kommunikoivat tuotosten välityksellä.

Syn (2007) malli käyttäjäkeskeiseen suunnitteluun aloitetaan niin kutsutulla nollasprintillä, jossa tehdään etukäteissuunnittelua ja kerätään tietoa tulevista käyttäjistä. Nollasprintillä käytetyt menetelmät riippuvat siitä, ollaanko kehittämässä uutta tuotetta vai olemassa olevan tuotteen uutta versiota, mutta työkaluina voidaan käyttää esimer-

kiksi haastatteluja tai toimintaympäristökartoitusta (*contextual inquiry*). Nollasprintillä tehdään hyvin samanlaisia asioita kuin vesiputousmallin suunnittelu- ja analyysivaiheen aikana. Erona kuitenkin on se, että vaihe kestää useiden kuukausien sijaan muutamia viikkoja ja työtä jatketaan edelleen seuraavien sprinttien aikana. Nollasprinttiä pidetään kuitenkin tärkeänä sen takia, että voidaan muodostaa projektista kokonaiskuva, jonka pohjalta työtä voidaan myöhemmin jatkaa. Koska sekä suunnittelu, että toteutus tehdään inkrementaalisesti ja iteratiivisesti, ketterä kehitys asettaa haasteita kokonaiskuvan hallinnalle (Da Silva, 2012). Da Silvan (2012) mukaan on tärkeää, että suunnittelijoita käytetään projektin suunnitteluvaiheessa, koska ohjelmistokehittäjillä on taipumus kuunnella sitä, mitä asiakkaat haluavat eikä pohtia sitä, mitä he oikeasti tarvitsevat.

Ensimmäisessä sprintissä toteutetaan ominaisuudet, joissa on paljon teknistä toteutettavaa, mutta vain vähän työtä UX-suunnittelijalle. Näin paljon UX-työtä vaativat ominaisuudet voidaan suunnitella jo nollasprintin aikana. Nollasprintin jälkeen siirrytään tekemään UX-työtä yhden sprintin verran itse toteutuksen edellä. Ensimmäisen sprintin aikana UX-suunnittelija aloittaa seuraavassa sprintissä toteutettavien ominaisuuksien suunnittelun ja tätä tapaa jatketaan projektin edetessä. Osa suunnittelutyöstä voi olla niin monimutkaista, että sitä ei ehditä toteuttamaan yhden sprintin aikana. Sy ratkaisee ongelman siten, että suuret kokonaisuudet pilkotaan pienempiin, sprinttiin sopiviin paloihin ja nämä palat lisätään inkrementaalisesti kokonaisuuteen. Tapa on sinänsä hyvin tyypillinen ketterille menetelmille ja ohjelmistotuotannolle yleensä. Se, millaista testausta UX-suunnittelijat tekevät tehdyille ominaisuuksille riippuu siitä, missä vaiheessa projekti on.

## 4.4 Lean UX

Lean UX on menetelmä, josta on vielä kohtalaisen vähän akateemista tutkimusta, mutta se on herättänyt mielenkiintoa yritysmaailmassa. Se on hyvin nopea käyttäjäkeskeinen ohjelmistokehityksen menetelmä, joka on suunnattu erityisesti uuden tuotteen kehitykseen start up -yrityksissä. Lean UX on saanut vaikutteita suunnitteluajattelusta (*design thinking*), Lean startup -metodista ja ketterästä ohjelmistokehityksestä. Lean UX:n tavoite on tuottaa tuote, joka täyttää asiakkaan tarpeet niin nopeasti kuin mahdollista ja mahdollisimman vähäisillä resursseilla. Tämä saavutetaan aloittamalla nopeasti sellaisten prototyyppien tekeminen, joita voidaan testata käyttäjillä. Työ yritetään tehdä mahdollisimman tehokkaasti, mahdollisimman pienellä hukalla. Metatason spesifikaatiot ja suunnittelussa syntyvät tuotokset pyritään pitämään mahdollisimman keveinä. Samoin tiimit yritetään pitää mahdollisimman pieninä. Hukkaa vältetään myös sillä, että tiimit ovat **monitoiminnallisia** ja sijaitsevat **yhdessä paikassa**. Menetelmässä käyttäjät osallistuvat jatkuvasti kehitykseen. Lean UX on iteratiivinen menetelmä. Jokaisen kehityssprintin tuloksena tulisi syntyä pienin toimiva tuote (*MVP = minimum viable product*), jota voidaan testata. Testeistä saadaan tietoa suunnitteluratkaisuista ja toteutukses-

ta. Lean UX:ssä suunnittelu tehdään yhteistyössä. (Gothelf, 2013) Menetelmä jakaa siis tietoisesti suunnittelutyötä myös muulle kehitystiimille, toisin kuin monet muut ketteriä ja käyttäjäkeskeisiä menetelmiä yhdistävät mallit. Pidän lähestymistavan hyötynä sitä, että näin saadaan omistajuutta ja tietoisuutta käyttäjäkokemuksesta siirrettyä myös muulle kehitystiimille.

Lean UX -kirjassaan Gothelf (2013) määrittelee Lean UX -menetelmän 15 periaatteen avulla. Liikkanen ym. (2014) käyttävät konferenssijulkaisussaan internet-yhteisön näistä periaatteista tiivistämää Lean UX -manifestia:

1. Asiakas **validoi** tuotteen **aikaisin**. Tuotetta ei julkaista niin, että sen arvoa asiakkaalle ei tunneta.
2. **Suunnittelu tehdään yhteistyössä ja monialaisesti**. Ei käytetä yksinäisiä "sankarisuunnittelijoita".
3. **Ratkaistaan asiakkaan ongelmia** eikä keskitytä hienolta näyttävien ominaisuuksien lisäämiseen.
4. **Mitataan** olennaisimpia tuottavuuteen vaikuttavia tekijöitä (*KPI = key performance indicators*). Ei jätetä määrittelemättä onnistumiskriteerejä.
5. Valitaan **joustavasti sopivat työkalut**. Ei seurata orjallisesti jäykkää metodologiaa.
6. **Suunnitellaan kevyesti**. Ei käytetä raskaita rautalankamalleja tai spesifikaatioita.

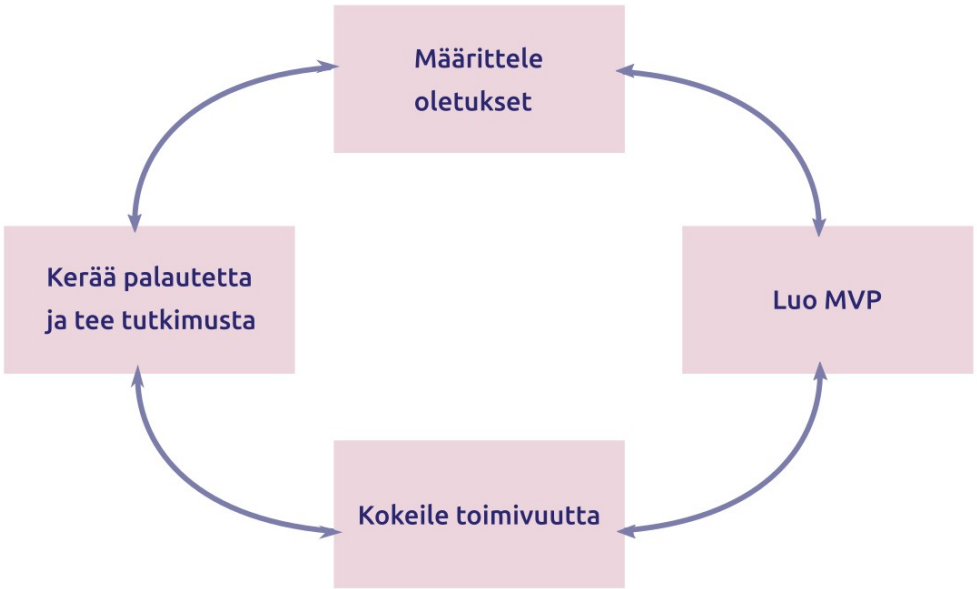
Lean UX -periaatteet ovat hyvin lähellä Lean-filosofian periaatteita. Pääpaino on spesifikaatioiden sijaan lopputuotteissa. Kun asiakas validoi tuotteen aikaisin, kyetään epäonnistumaan mahdollisimman aikaisin ja vältytään toteuttamasta ominaisuuksia, jotka eivät ole tarpeellisia. Toisaalta yritetään välttää hukkaa myös suunnittelun lopputuotteita tehtäessä. Se, että suunnittelu ja kehitys tehdään yhdessä tiimissä, vähentää suunnitteluratkaisujen dokumentoinnin tarvetta. Kun suunnittelija toimii samassa tiimissä kehittäjien kanssa, mahdolliset ongelmat on helppo käydä läpi sanallisesti ja suunnitteluratkaisuista on helppo keskustella. Onnistumista mittaamalla pystytään parantamaan jatkuvasti omaa toimintaa. (Liikkanen *ym.*, 2014)

Menetelmä koostuu neljästä vaiheesta, joka esitetään kuvassa 4.4.1: 1) oletuksien havainnoinnista ja priorisoinnista, 2) pienimmän toimivan tuotteen tekemisestä, 3) tuotteen kokeilemisesta ja 4) palautteesta ja tutkimisesta. Vaiheita toistetaan sykleissä ja edelliseen vaiheeseen on mahdollista tarvittaessa palata. **Oletuksien havainnoinnissa** kootaan oletukset, joita tiimillä on ja kaikki tiimin jäsenet pääsevät kertomaan mielipiteensä siitä, miten ongelma tulisi ratkaista. Tämän jälkeen tehdään priorisointi niin, että korkeimmalla prioriteetilla ovat sellaiset oletukset, joissa olisi pahinta olla väärässä ja

jotka ovat aiheesta, josta tiedetään vähiten. Tämän jälkeen oletuksista luodaan prioriteettijärjestyksessä **testattavia hypoteeseja**. Hyvä hypoteesi kertoo, että

1. kun **tehdään** tälläinen ominaisuus tai ratkaisu
2. tälle **käyttäjäryhmälle,**
3. siitä seuraa tälläinen **lopputulos** ja
4. **tiedämme, että hypoteesi on totta**, silloin kun voimme todeta tapahtuvan näin.

Kun hypoteesit on muodostettu, tehdään **pienin mahdollinen tuote**, jolla saadaan testattua niitä. MVP:n avulla pyritään selvittämään, onko suunnitellulle ratkaisulle tarvetta, onko sillä arvoa asiakkaalle ja onko ratkaisu käytettävä. MVP voi olla koodattu tai esimerkiksi paperiprototyyppi. Tämän jälkeen siirrytään **keräämään palautetta ja tekemään tutkimusta**. Tutkimuksen tekeminen on Lean UX:ssä jatkuvaa ja se tehdään yhteistyössä. Tämä tarkoittaa sitä, että tutkimusaktiviteetteja on joka sprintin aikana ja koko kehitystiimi osallistuu tutkimuksen tekemiseen. Tiimi päättää, mitä sen täytyy oppia, luo suunnitelman tämän tiedon saamiseen ja jalkautuu tapaamaan pareittain asiakkaita tai käyttäjiä. Tutkimisen tarkoituksena on testata vanhoja hypoteeseja ja kerätä tietoa, jotta voidaan tehdä paremmin uusia. (Gothelf, 2013)




***Kuva 4.4.1.****LeanUX prosessi*

Gothelf ehdottaa myös tapoja, kuinka Lean UX saadaan integroitua Scrumiin. Hänen mielestään kehitys on hyvä tehdä teemoissa. **Teema** ulottuu useammalle, esimerkiksi kolmelle, sprintille ja sisältää UX-suunnittelulle järkevän kokonaisuuden. Jokainen teema tulisi aloittaa **aloituspalaverilla** (*kickoff*), jossa käytetään erilaisia aivoriihi- ja validaatiotekniikoita. Tämän aloituspalaverin tarkoitus on saada **koko tiimi** luonnostelemaan ja ideoimaan yhdessä. Aloituspalaverin tuloksena saadaan lista (*backlog*) ideoita, joita voidaan testata ja joista voidaan oppia. Tämä lista muodostaa rungon sprinttien aikana tehtävälle suunnittelutyölle. Tarvittaessa joka sprintin alussa voidaan pitää lyhyempiä **aivoriihiä**, jossa käsitellään uusia ideoita ja muokataan suunnitelmia uudistuneen tietämyksen pohjalta. *Kickoff*-sessiota seuraa **iteraation suunnittelupalaveri**, jossa kickoff-sessiossa tehtyjen tuotoksien pohjalta tehdään käyttäjätarinat, arvioidaan ne ja priorisoidaan tarinat tärkeysjärjestykseen. Lisäksi Gothelfin mukaan tarvitaan **viikoittainen tapaaminen asiakkaiden kanssa**. Tässä tapaamisessa arvioidaan toimivatko tehdyt suunnitelmat ja tuotteet. (Gothelf, 2013)

# 5 TUTKIMUSMENETELMÄT

Tässä luvussa käydään läpi työssä käytettyjen tutkimusmenetelmien teoriaa lyhyesti. Aliluvussa 5.1 käsitellään laadullisen tutkimuksen piirteitä ja verrataan sitä määrälliseen tutkimukseen. Aliluku 5.2 kertoo teemahaastattelusta tutkimusmenetelmänä ja aliluvussa 5.3 tutustutaan laadullisen tutkimuksen laadun arviointiin.

## 5.1 Laadullinen eli kvalitatiivinen tutkimus

Yksi tapa luokitella tutkimusmenetelmiä on jakaa ne kvantitatiiviseen eli määrälliseen tutkimukseen ja kvalitatiiviseen eli laadulliseen tutkimukseen. Määrällinen tutkimus perustuu kohteen kuvaamiseen ja tulkitsemiseen tilastojen ja numeroiden avulla (Koppa, 2015). Laadullinen tutkimus puolestaan pyrkii ymmärtämään kohteen laatua, ominaisuuksia ja merkityksiä kokonaisvaltaisesti.

Jeff Sauro (2015) jakaa MeasuringU-internetsivustolla laadullisen tutkimuksen edelleen viiteen osa-alueeseen, 1) etnografiseen, 2) narratiiviseen, 3) fenomenologiseen, 4) ankkuroituun teoriaan ja 5) tapaustutkimukseen. **Etnografinen tutkimus** tähtää tutkimuskohteen kokonaisvaltaiseen ymmärtämiseen ja kuvaamiseen. Sen tavoitteena on kuvata ja ymmärtää ihmisten toimintaa omassa ympäristössään tai ryhmän jäsenten tulkintoja ja käsityksiä ympäristöstään ja toiminnastaan. Havainnoija on yleensä fyysisesti läsnä tutkittavien ympäristössä ja suorassa vuorovaikutuksessa tutkittaviensa kanssa. **Narratiivisessa tutkimuksessa** ollaan kiinnostuneita siitä, millaisia kertomuksia tutkimuskohteesta kerrotaan tai miten kohde nähdään kertomusten kautta kulttuurissa tai yhteiskunnassa. **Fenomenologinen tutkimus** perustuu välittömien havaintojen tekemiseen tutkimuskohteesta ilman ennalta määrättyjä oletuksia tai teoreettista viitekehystä. **Ankkuroitu teoria** ei hämäävästä nimestään huolimatta ole varsinainen teoria, vaan se on käsite, jota käytetään kuvaamaan aineistolähtöistä tutkimusta. Siinä pyritään käsitteellistämään ja luomaan suhteita dataan liittyvien käsitteiden välille. **Tapaustutkimuksessa** tutkitaan yksittäistä organisaatiota, käsitettä, yritystä tai tapausta. Tapaustutkimuksessa pyritään ymmärtämään kohdetta syvällisesti useiden datalähteiden kautta. (Koppa, 2014b; Sauro, 2015)

Määrällinen ja laadullinen tutkimus sopivat eri käyttötarkoituksiin. Määrällisellä tutkimuksella saadaan vertailukelpoista tietoa, jota voidaan analysoida helposti tilastotieteen keinoin. Se sopii erityisesti suurille kohdejoukoille ja silloin, kun tutkimuskysymykset pystytään pilkkomaan niin pieniksi ja täsmällisiksi osiksi, että niihin on järkevää vastata numeroin. Laadullinen tutkimus antaa puolestaan vastauksia sellaisiin kysymyk-

siin, joita ei voi mitata. Määrällisessä tutkimuksessa ei saada tietoa ilmiön monimuotoisuudesta, eikä siinä pystytä ottamaan huomioon kaikkia sen erityispiirteitä. (Flick, 2006) Määrällistä tutkimusta käytetään erityisesti luonnontieteellisessä tutkimuksessa, jossa ympäristön ilmiöitä puetaan matemaattiseen muotoon käyttäen apuna ongelmien pilkkomista pieniin osiin ja suuria määriä toistoja. Laadullista tutkimusta käytetään puolestaan erityisesti humanistisissa tieteissä.

Pitkittäistutkimuksesta ja poikittaistutkimuksesta puhutaan sen mukaan, miten tutkimus sijoittuu aika-akselille. Pitkittäistutkimuksessa tutkitaan muutosta ja ilmiön kehittymistä ajan suhteen. Samaa tutkimuskohdetta seurataan pidemmän aikaa, jotta saadaan havaittua tapahtuvat muutokset ja pystytään analysoimaan muutokseen vaikuttaneita tekijöitä ja niiden aikaansaamia seurauksia. Poikittaistutkimuksessa puolestaan ollaan kiinnostuneita tilanteista ja samantapaisten ilmiöiden ilmenemismuodoista tiettynä ajankohtana. Poikkileikkausaineistolla tehtävillä analyyseilla voidaan lähinnä kuvailla eri ilmiöitä. Sillä ei pystytä tutkimaan asioiden syy-seuraussuhteita. (Koppa, 2014b; Valli, 2018)

## 5.2 Teemahaastattelu

Kun tutkimuksen tavoitteena on tuottaa sellaista tietoa, joka koskee esimerkiksi mielipiteitä, käsityksiä, havaintoja, asenteita, arvoja tai kokemuksia, tutkimusaineisto on mielekästä koota haastattelemalla. Haastattelussa tutkija osallistuu vuorovaikutteisesti tutkimustiedon tuottamiseen haastateltavan tai haastateltavien kanssa. Se on monipuolinen tutkimusmenetelmä, jolla voidaan kerätä tietoa hyvin erilaisista aiheesta. Haastattelun perusajatus on sinänsä yksinkertainen: jos halutaan tietää, mitä joku ajattelee jostakin, kysytään sitä häneltä. Haastattelu eroaa kuitenkin arkisestaa keskustelusta siten, että tutkimushaastattelulla on selkeä päämäärä. Haastattelun avulla saatavaa aineistoa on tarkoitus analysoida ja tulkita, jotta saataisiin vastattua tutkimuskysymyksiin. (Saaranen-Puusniekka ja Kauppinen, 2006; Koppa, 2014a)

Haastattelut voidaan luokitella muunmuassa sillä perusteella, kuinka tarkkaan haastattelun kulku on suunniteltu etukäteen ja kuinka paljon haastattelun annetaan vaellella aiheesta toiseen. Strukturoidussa haastattelussa käytetään lomaketta, jossa on valmiita kysymyksiä ja valmiita vastausvaihtoehtoja. Kysymykset esitetään aina samassa järjestyksessä ja haastateltavan tulee valita valmiista vastausvaihtoehdoista. Puolistrukturoitu haastattelu etenee puolestaan niin, että kaikille haastateltaville esitetään samat tai lähes samat kysymykset samassa järjestyksessä. Avoimessa haastattelussa keskustelun etenemistä ei ole lyöty lukkoon, vaan se etenee tietyn aihepiirin sisällä vapaasti ja paljolti haastateltavan ehdoilla. (Saaranen-Puusniekka ja Kauppinen, 2006)

Teemahaastattelu on puolistrukturoidun haastattelun muoto, jossa haastattelun aihepiirit eli teemat on määrätty jo ennalta. Kysymyksien tarkkaa muotoa ja järjestystä ei ole kuitenkaan määritelty, kuten strukturoidussa haastattelussa tehdään. Vaikka aihepii-

rit ovat kaikille haastateltaville samoja, niissä liikutaan joustavasti ilman tiukkaa etenemisreittiä. Teemahaastattelussa ihmisten vapaalle puheelle annetaan tilaa, ennalta päätetyt teemat pyritään kuitenkin käymään läpi kaikkien tutkittavien kanssa. Kaikkien haastateltavien kanssa ei tarvitse käydä läpi kaikkia teemoja samassa laajuudessa. Teemahaastattelu edellyttää aihepiiriin perehtymistä huolellisesti ja haastateltavien tilanteen tuntemista. Näin haastattelu voidaan kohdistaa juuri tiettyihin teemoihin. Haastattelussa voidaan haastatella yhtä henkilöä kerrallaan, jolloin puhutaan yksilöhaastattelusta. Jos haastateltavia henkilöitä on useita, puhutaan ryhmähaastattelusta. (Saaranen-Puusniekka ja Kauppinen, 2006; Valli, 2018)

## 5.3 Laadullisen tutkimuksen laadun arviointi

Laadullisen tutkimuksen menetelmien ja tulosten laadun arviointiin on kehitetty useita erilaisia tapoja. Klassiset kriteerit laadullisen tutkimuksen luotettavuuden arviointiin ovat 1) **luotettavuus** (*reliability*), 2) **oikeellisuus** (*validity)*, ja 3) **objektiivisuus** *(objectivity)*.

Flickin (2006) mukaan Kirk ja Miller (1986) esittävät, että laadullisen tutkimuksen luotettavuutta voidaan arvioida kolmella eri tavalla. Ensinnäkin on se, kuinka helppo tulokset on toistaa tietyllä menetelmällä. Kirkin ja Millerin mukaan tämä on epäluotettava tapa tarkastella luotettavuutta. Erityisesti kenttätutkimuksissa tämä tapa johtaa siihen, että tutkimuksessa yritetään sovittaa tuloksia odotettuun malliin, eikä tarkastella sitä, mitä oikeasti saadaan selville. Toinen tapa on se, että kuinka pysyviä mittaustulokset tai havainnot ovat aika-akselilla. Tämän näkökulman ongelma on se, että tutkimuksen kohteena oleva ilmiö voi muuttua. Kolmas tapa on se, että kuinka pysyviä tulokset ovat, kun ne saadaan samalla hetkellä mutta käyttäen eri työkaluja.

Toisaalta Flickin (2006) mukaan etnografisen tutkimuksen laatua voidaan arvioida sillä, kuinka hyvin tutkimushaastattelut on nauhoitettu ja dokumentoitu. Kirk ja Miller suosittelevat toistamaan muistiinpanojen teossa tiettyjä käytäntöjä, jotta muistiinpanojen vertailtavuus paranee etenkin silloin, kun haastattelijoita tai tarkkailijoita on useita. Datan tulkinnassa voidaan käyttää esimerkiksi avointa koodausta tulosten tulkinnan laadun parantamiseen. Ylipäätään tutkimuksen luotettavuuden voidaan sanoa olevan sitä parempi, mitä paremmin koko tutkimusprosessi on dokumentoitu (Flick, 2006).

Oikeellisuus voidaan tiivistää siihen, vastaako todellisuus tutkimuksessa esitettyjä tuloksia. Oikeellisuuden kannalta tutkimuksessa voidaan tehdä kolmenlaisia virheitä: 1) nähdään korrelaatio tai periaate silloin, kun sitä ei ole, 2) ei löydetä sitä silloin kun se on tai 3) kysytään vääriä kysymyksiä. Tyypillinen kysymys tutkimuksen oikeellisuuden arvioinnissa on, miten tutkimuksen tekeminen ja itse tutkija vaikuttavat saatuihin tuloksiin. Tavoitteena on, että tutkimuksen kohde on todellisuudessa mahdollisimman samanlainen kuin millaisena se on esitetty tutkimuksessa. Lisäksi tutkimuksen tulee olla tehty niin läpinäkyvästi, että tutkimuksen lukijat pystyvät arvioimaan kuinka hyvin se on to-

teutettu. Oates (2006) jakaa oikeellisuuden sisäiseen oikeellisuuteen ja ulkoiseen oikeellisuuteen. Sisäisellä oikeellisuudella tarkoitetaan, sitä, kuinka hyvin tutkija onnistuu tulkitsemaan todellisuutta. Sitä voidaan parantaa triangulaatiomenetelmien avulla, jotka perustuvat siihen, että dataa kerätään usealla eri tavalla, tai siten, että haastateltavat tarkistavat raakaversion tuloksista. Ulkoinen oikeellisuus kertoo siitä, kuinka hyvin löydökset pystytään siirtämään toisiin tapauksiin ja onko tutkimus ja tulokset kuvattu tarpeeksi laajasti, niin että lukijat voivat arvioida onko heidän tuloksissaan samanlaisia piirteitä. (Flick, 2006; Oates, 2006)

Flick (2006) kertoo, että Maddill ym. (2000) mukaan objektiivisuuden arviointi soveltuu vain, kun tutkimus tehdään realististisessa viitekehyksessä. Realistisessa tutkimussuuntauksessa ajatellaan, että ilmiöiden tarkka havainnoiminen mahdollistaa tutkijan pääsyn yhä lähemmäksi tutkijasta, näkökulmasta ja ympäristöstä riippumatonta totuutta ja tietoa (Koppa, 2014b). Objektiivisuus tarkoittaa silloin sitä, että eri tutkijat päätyvät samoihin lopputuloksiin saman datan pohjalta. Oates (2006) neuvoo arvioimaan objektiivisuutta sen perusteella, yrittääkö tutkija tavoitella jotain tiettyä lopputulosta ja vaikuttaako tutkimuksen rahoittaja jollain tavoin tutkimuksen lopputulokseen.

# 6 TUTKIMUSSUUNNITELMA

Tutkimuksen tavoitteena oli muodostaa kokonaiskuva siitä, miten käyttäjäkokemuksen suunnittelu on huomioitu projekteissa. Lisäksi oltiin kiinnostuneita erityisesti kehittäjien roolista käyttäjäkokemuksen muodostajina ja kehittäjien kokemuksista käyttäjäkokemuksen suunnittelusta. Tässä luvussa käydään läpi tutkimukselle asetetut tavoitteet ja tavoitteiden saavuttamiseksi käytetyt menetelmät.

## 6.1 Tutkimuskysymykset ja aiheen rajaus

Työn tavoitteena oli selvittää sitä, miten käyttäjäkokemukseen vaikuttavia ratkaisuja tehdään Tampereella toimivissa ohjelmistoyrityksissä. Tutkimuskysymykset muotoiltiin seuraavasti:

1. Miten käyttäjäkokemuksen suunnittelu on liitetty mukaan ketterien projektien käytäntöihin?
2. Ketkä projekteissa tekevät käyttäjäkokemukseen vaikuttavia päätöksiä?
3. Millaisia käyttäjäkokemukseen vaikuttavia päätöksiä kehittäjät tekevät?
4. Miten kehittäjän kokevat käyttäjäkokemukseen vaikuttavien päätösten tekemisen projektissa?

Ensimmäisen kahden tutkimuskysymyksen tarkoituksena on luoda pohjatietämys, jotta kysymyksiin 3 ja 4 pystyttäisiin vastaamaan. Kysymykset kolme ja neljä pyrkivät kartoittamaan, millainen rooli kehittäjillä on käyttäjäkokemuksen suunnittelussa ja mikä heidän kokemuksensa siitä on. Alun perin työssä oli vielä yksi tutkimuskysymys: ”Ovatko käyttäjäkokemukseen vaikuttavat päätökset tietoisia vai tiedostamattomia?” Tämän selvittäminen kuitenkin todettiin vaikeaksi haastattelukysymyksillä, joten se päätettiin pudottaa pois tutkimuksesta. Tutkimuskysymysten asettelu työssä on melko laaja. Tämä katsottiin järkeväksi, koska työ on opinnäytetyö, jonka tavoitteena on tuottaa tekijälleen mahdollisimman hyvä yleiskäsitys aihepiiristä. Jos tavoite olisi ollut julkaista aiheesta alan julkaisussa, tutkimuskysymyksiä olisi rajattu selkeämmin kapeammalle alalle, jotta oltaisiin saatu syvällisempää tietoa.

Mukaan tutkimukseen otettiin vain projekteja, jotka täyttivät seuraavat ehdot:

1. projektissa käytettiin jotakin ketterää menetelmää,

2. lopputuotteessa oli käyttöliittymä, jota ihminen käytti,
3. projektissa toteutettiin uusia ominaisuuksia tuotteeseen,
4. ohjelmistosta oli julkaistu jo vähintään yksi versio (tai julkaisu oli hyvin lähellä),
5. projektissa oli vain yksi toimittaja.

Rajauksella pyrittiin saamaan tutkimukseen projekteja, joissa kehitystyö ja käytettävät menetelmät olisivat vakiintuneet. Tämä tutkimus ei kata toteutustyötä edeltävää selvitysvaihetta, jossa tehdyt ratkaisut vaikuttavat luonnollisesti paljon siihen, millaiseksi käyttäjäkokemus muodostuu. Rajaus tehtiin sen vuoksi, että oltiin kiinnostuneita nimenomaan kehittäjien arjesta toteutusvaiheessa. Tämän vuoksi myös käyttäjätestausten suorittaminen rajattiin tutkimuksen ulkopuolelle. Käyttäjätestaukset ovat laaja aihe ja kehittäjät eivät Kuusisen (2015b) havaintojen mukaan osallistu käyttäjätutkimuksien toteuttamiseen. Monitoimittajaprojektit rajattiin tutkimuksen ulkopuolelle, koska oletettiin, että niiden vaatimat prosessit ovat erilaisia ja monimutkaisempia kuin yhden toimittajan projekteissa.

## 6.2 Työhön valitut tutkimusmenetelmät

Työn tutkimuskysymyksiin ja tavoitteisiin sopii hyvin laadullinen analyysi. Laadullinen tutkimusstrategia sopii kokemusten ja havaintojen keräämiseen. Tässä tapauksessa myöskään tutkimusaihetta ei tunnettu niin hyvin, että määrällistä tutkimusta olisi voinut käyttää ainakaan ainoana menetelmänä. Tutkimusstrategiaksi valittiin poikittainen tutkimus. Työtä varten haastateltiin henkilöitä useista eri projekteista yhden kerran. Haastattelumuodoksi valittiin teemahaastattelu. Haastattelulle laadittiin runko teemoittain (liite A), johon koottiin haastattelussa esille nostettavia aiheita. Haastattelukysymykset jätettiin haastattelurunkoon melko korkealle abstraktiotasolle. Haastattelut nauhoitettiin äänitallenteina haastateltavien suostumuksella (liite B).

Haastattelutilanteet kestivät puolesta tunnista reiluun tuntiin. Haastattelujen kesto riippui lähinnä haastateltavien aikatauluista. Haastattelurunkoa ei seurattu järjestelmällisesti, vaan keskustelun annettiin kulkea mahdollisimman luontevasti. Haastattelun lisäksi haastateltavilta kysyttiin erillisellä lomakkeella muutamia kysymyksiä (liite C). Kysymyksillä pyrittiin kartoittamaan, millainen rooli haastateltavalla oli projektissa ja selvitettiin, mitä haastateltava ajattelee käyttäjäkokemuksen olevan. Jos haastattelun jälkeen havaittiin, että projektin käytännöissä oli jäänyt jotain epäselväksi, haastateltavilta kysyttiin tarkennuksia pikaviestisovelluksella. Kokemuksista pikaviestimillä ei enää kysytty. Jotkut haastateltavat jakoivat kuitenkin siinä vaiheessa vielä kokemuksiaan pyytämättä. Niitä käytettiin myös työn aineistona.

Haastattelun jälkeen nauhoite litteroitiin. Tässä tapauksessa litterointi tarkoitti vain sen kirjoittamista, mitä haastattelussa oli sanottu ja kuka sen oli sanonut. Haastatel-

tavien pitämiä taukoja tai äänenpainoja ei pyritty analysoimaan. Aineistoa alettiin analysoida alustavasti jo sinä aikana, kun aineiston kerääminen oli vielä kesken. Sen ansiosta pystyttiin tekemään havaintoja aineiston kyllääntymisestä eli saturaatiosta. Alilukua 7.2 varten aineistoon tehtiin avoin koodaus. Haastattelut pilkottiin ja haastatteluissa esille tulleet asiat lajiteltiin teemoista johdettuihin osa-alueisiin.

Haastatteluja tehtiin sekä yksilöhaastatteluina että ryhmähaastatteluina. Yksilöhaastattelut olivat kevyempi tapa kerätä aineistoa. Ryhmähaastattelut olivat mielenkiintoisia, koska niissä näkyi tiimin sisäinen dynamiikka. Lisäksi ajateltiin, että projektitiimit voivat hyötyä hetkestä, jolloin pysähdytään pohtimaan projektin käytäntöjä. Aikataulujen sopiminen ryhmähaastatteluihin osoittautui kuitenkin hankalaksi. Yksilöhaastatteluja käyttämällä saatiin laajempi otos erilaisista projekteista. Toisaalta ryhmähaastattelut ja yksilöhaastattelut tuottavat hieman erilaista tutkimusdataa. Ryhmässä haastateltavat saattavat sanoa vuorovaikutuksen ansiosta jotain, mitä eivät muuten muistaisi (Saaranen-Puusniekka ja Kauppinen, 2006) tai olisi tulleet ajatelleeksi. Toisaalta ryhmä voi myös rajoittaa tai muuten vaikuttaa ihmisten sanomisia. Esimerkiksi yksilöhaastatteluissa olisi voinut tulla esille, että UX-suunnittelijoihin suhtaudutaan negatiivisesti ja asia olisi voinut jäädä ryhmähaastatteluissa piiloon. Projekteissa työskentelevien henkilöiden lisäksi päätettiin haastatella myös projektipäällikköä ja (palvelu)muotoilijaa, jotta saataisiin laajempi kokonaiskuva erilaisissa projekteissa käytössä olevista käytännöistä.

# 7 TULOKSET

Aliluvut 7.1 ja 7.2 vastaavat tutkimuskysymyksiin 1 ja 2. Luku 7.3 vastaa tutkimuskysymyksiin 3 ja 4. Työstä saadut tulokset on jaettu projektien käytäntöihin, jotka esitellään luvussa 7.1. Kehittäjien tekemiä käyttäjäkokemukseen vaikuttavia ratkaisuja tarkastellaan kehittäjien näkökulmasta luvussa 7.3.

Työtä varten haastateltiin kehittäjiä seitsemästä eri projektitiimistä. Neljä näistä projekteista olivat yrityksestä 1 ja muista yrityksistä oli tarkasteltavana yksi projekti yritystä kohti. Kehittäjiä oli haastateltavana yhteensä 12 ja heistä yhdeksän olivat yrityksen 1 projekteissa. Osa näistä haastatteluista tehtiin yksilöhaastatteluina ja osa ryhmähaastatteluina, joissa saattoi olla myös muiden roolien edustajia.

Yrityksen 1 käytäntöjä tarkasteltiin vielä laajemmin eri roolien näkökulmasta tapaustutkimuksen omaisesti. Nämä tulokset esitellään aliluvussa 7.2. Aliluvussa käsitellään tuloksia, jotka saatiin haastattelemalla projekteittain ryhmähaastatteluissa kehittäjiä, projektipäällikköä/palveluarkkitehtia, teknistä projektipäällikkö-Scrum-masteria, palvelumuotoilija-UX-suunnittelijaa, UX-suunnittelijaa ja teknistä projektipäällikköä. Lisäksi haastateltiin yksilöhaastattelussa kokenutta teknistä projektipäällikköä ja kokenutta (palvelu)muotoilijaa. Haastatteluja työtä varten tehtiin Tampereella ja Helsingissä.

Projektien käytännöt kuvataan tuloksissa projekteittain. Näin pyritään saamaan käsitys eri projektien eroavaisuuksista. Sen sijaan haastateltujen henkilöiden mielipiteet ja kokemukset haluttiin jättää erilleen projektista ja yrityksestä, jotta saatiin suojeltua haastateltujen anonymiteettia. Lisäksi haastatteluissa nousi esiin myös kehittäjien kokemuksia aiemmista projekteista, joita hyödynnettiin myös tämän työn tuloksissa.

## 7.1 Käyttäjäkokemuksen suunnittelun yhdistäminen ketteriin menetelmiin projekteissa

Tässä aliluvussa kerrotaan käytetyistä projektimalleista ja miten käyttäjäkokemuksen suunnittelu oli toteutettu projektissa. Projektien käytäntöjä ei saatu haastatteluissa aina tarkasti selvitettyä. Epäselvissä tapauksissa projektien käytäntöjä tarkennettiin vielä jälkeenpäin.

### 7.1.1 Projekti 1, yritys 1

**Projekti**: Ensimmäinen versio tuotteesta julkaistu vuonna 2017. Sovellukseen kehitetään edelleen aktiivisesti uusia ominaisuuksia. Asiakkaana on julkishallinnon organisaa-

tio. Haastateltavana kaksi kehittäjää, projektipäällikkö-palveluarkkitehti, tekninen projektipäällikkö-Scrum master ja palvelumuotoilija-UX-suunnittelija.
**Projektimalli:** SAFe
**Projektin käytännöt:** Sprintit suunnitellaan 4 sprinttiä kerrallaan hankeinkrementteinä, jotka kestävät 10 viikkoa. Viimeinen sprintti on demoamis-, kokeilu- ja oppimisiteraatio. Sprintit ovat kahden viikon pituisia. Asiakkaan kanssa käydään läpi tehdyt muutokset sprintin demopalaverissa. Asiakas hyväksymistestaa tämän jälkeen muutokset. Tiimissä on mukana testaaja, joka testaa myös muutokset.
Palvelumuotoilua ja UX-suunnittelua tehdään jatkuvasti. Tavoitteena on, että palvelumuotoilua tehtäisiin konseptityyppisenä jo ennen kuin SAFen eeppisiä ominaisuuksia kirjoitetaan eli siis paljon ennen, kuin ne on pilkottu toiminnallisuuksiksi ja toiminnallisuudet ovat päätyneet työjonolle.
Palvelumuotoilijat kuuluvat omien tiimiensä lisäksi asiakaskokemustiimiin, joka pyrkii ennakoimaan käyttäjätarvetutkimus- ja palvelumuotoilutarpeet etukäteen niin, että edellisen hankeinkrementin aikana tehdään tarvittavat käyttäjätarvetutkimukset ja alustava palvelukonsepti siihen. Käytännössä tämä vaatii kuitenkin sen, että työ on suunniteltu tarpeeksi hyvin hanketasolla etukäteen.
Tuoteomistaja ja välillä myös Scrum-masteri täsmentävät ja jakavat asiakkaan eeppiset visiot työjonon ominaisuuksiksi. UX-suunnittelija täydentää yleensä vielä alustavaa ominaisuutta työjonossa. Esimerkiksi UX-suunnittelija voi tehdä yhdestä kokonaisesta eeppisen tason ominaisuudesta rautalankaprototyypin. Sen jälkeen, kun eeppisen tason ominaisuudet on jaettu yksittäisiksi ominaisuuksiksi, UX-suunnittelija suunnittelee tarkemmin niiden toiminnallisuuden. Tämä tapahtuu 2-3 sprinttiä ennen, kuin ominaisuus tulee työjonossa toteutukseen. Graafinen spesifikaatio tulee yleensä myös UX-suunnittelijalta näiden alustavien kuvausten perusteella. Aiemmin on pidetty myös UX-päiviä, jossa on suunniteltu ominaisuuksia tiimin, asiakkaiden, palvelumuotoilijan ja UX-suunnittelijan kanssa. Retrospektiiveissä käyvät ne projektitiimin jäsenet, jotka ovat resursoituna projektiin yli 50 prosenttisesti. Lisäksi on ollut käytössä sprinttien suunnittelupalaverit, johon on osallistunut myös asiakkaan edustaja tai edustajia. Kehitysjonon työstöpalavereissa on täsmennetty työjonoon kirjattuja ominaisuuksia, mutta myös formalisoitu UX-työtä.

Tässä projektissa oli muista projekteista poiketen oma työjononsa myös käyttäjätutkimuksille, johon on kirjattu tehtäviä jo monen sprintin päähän. Tämä johtuu siitä, että käyttäjätutkimus vaatii viikkojen valmistelun, kun pitää hankkia oikeita loppukäyttäjiä testaukseen, järjestää työpaja, dokumentoida tulokset ja muodostaa niden pohjalta ehdotus palvelun toiminnaksi.

### 7.1.2 Projekti 2, yritys 2

**Projekti:** Julkishallinon projekti, joka on ollut tuotannossa jo pitkään. Kuitenkin edelleen kehitetään myös uusia ominaisuuksia. Tehty noin vuosi sitten isompi kokonaisuus

käytettävyys- ja käyttöliittymäparannuksia (haastateltava käytti termiä *”face lift”*). Haastateltavana oli yksi kehittäjä.
**Projektimalli:** SAFe
**Projektin käytännöt**: Sprintit kestävät kaksi viikkoa. Sprinteistä koostetaan hankeinkrementtejä, joissa on neljä kehityssprinttiä ja sen jälkeen sprintti, jossa viimeistellään ja mennään tuotantoon. Ennen inkrementtiä suunnitellaan sprinttien sisältöä etukäteen sellaisessa version suunnittelupäivässä. Muuten mennään periaatteessa Scrum-mallilla. Scrumin tapaamisista on käytössä sprinttien suunnittelupalaverit, sprintin katselmointi, retrospektiivit joka toisen sprintin jälkeen ja päivittäin pidetään Scrum-tapaaminen. Kehitysjonon työstöpalavereja (*backlog refinement*) pidetään tarvittaessa eli suunnilleen kerran viikossa.

Asiakkaan puolella on kehitysryhmä, jossa on muutamia loppukäyttäjiä, jotka hallinnoivat muilta loppukäyttäjiltä tulevia kehitysideoita ja ehdotuksia. Asiakkaan puolelta tulee puolestaan isompia kehitysideoita liittyen esimerkiksi tietoturvavaatimuksiin. Kehitystiimi määrittelee kehitysryhmän ja asiakkaan kanssa tarkemmin toteutettavat ominaisuudet. Projektille ollaan nimetty UX-asiantuntija, joka ei ole kuitenkaan projektin arjessa mukana. Jos ollaan tekemässä isompaa kokonaisuutta, kehitystiimi pyytää UX-asiantuntijan apua. UX-asiantuntija ja kehitystiimi kommunikoivat käyttöliittymäkuvien avulla ja suullisesti. Välillä pidetään yhteinen määrittelypalaveri käyttäjien, UX-asiantuntijan ja kehitystiimin kesken. Yleensä kuitenkin toimitaan niin, että UX-asiantuntija on tekemisissä vain kehittäjien ja projektipäällikön kanssa.

### 7.1.3 Projekti 3, yritys 1

**Projekti**: Projektissa kehitetään yrityksen omaa tuotetta. Haastattelussa kehittäjä, UX-suunnittelija ja koodaava tekninen projektipäällikkö.
**Projektimalli**: Kanban
**Projektin käytännöt:** Tehtäville, jotka vaativat käyttäjäkokemuksen suunnittelua on oma sarake kanban-taulussa. Käytännössä kuitenkin isommat kokonaisuudet tulevat UX-suunnitteluun tuoteomistajan kautta. Jos on pienempiä asioita, ne tulevat käytännössä UX-suunnitteluun projektipäällikön kautta. Ennen toteutusta toiminnallisuuksiin tehdään perusmäärittely, käyttöliittymäsuunnittelu ja näyttökuva. Vähän viimeistelyä vailla oleva näyttökuva lähtee toteutukseen ja toteutuksen edetessä näyttökuvaa voidaan vielä viimeistellä. Viimeistelyn syynä voi olla esimerkiksi se, että halutaan testata vielä jotain tai toteutusta halutaankin vielä viilata suuntaan tai toiseen. Projektissa on päivittäiset scrum-tapaamiset ja periaatteessa myös viikoittain palaveri tuoteomistajan kanssa. Pääsääntöisesti UX-suunnittelija osallistuu kaikkiin projektin tapaamisiin. Periaatteessa tuoteomistaja priorisoi toteutettavat ominaisuudet ja projektipäällikkö kirjoittaa ne sitten omiksi tehtävikseen ja määrittelee niitä samalla lisää. Se, että kuinka toteutetut ominaisuudet katselmoidaan, vaihtelee.

### 7.1.4 Projekti 4, yritys 3

**Projektin kuvaus:** Ylläpitovaiheessa oleva projekti, jossa asiakkaana on teollisuusyritys. Suurin osa tehtävistä ylläpitoa, mutta edelleen kehitetään myös uusia ominaisuuksia. Projektissa työskentelee 70 henkilöä, jotka on jaettu kymmeneen tiimiin. Haastateltavana oli yksi kehittäjä.
**Projektimalli:** Kanban
**Projektin käytännöt**: Eri tiimeillä on omat käytäntönsä. Osassa tiimeistä pidetään päivittäiset palaverit, joissain tiimeissä pidetään palaverit kaksi kertaa viikossa ja joissakin vain kerran viikossa. Retrospektiivejä pidetään hyvin harvoin. Sprintin suunnittelupalaverejä ei joko pidetty tai niihin osallistui vain muutama kehitystiimistä. Kerran viikossa pidettiin koko projektin palaveri, jossa jokainen kehitystiimi kertoi, että mitä oli saanut tehtyä kuluneen viikon aikana. Käytössä ei ollut määritellyn mittaisia sprinttejä vaan julkaisut tehtiin kanban-periaatteiden mukaan.
Projektissa ei ollut omaa UX-asiantuntijaa, mutta yrityksen UX-suunnittelija oli tarpeen mukaan käytössä. Jos kehittäjä arvioi, että tehtävään tarvittiin UX-suunnittelua, kehittäjä saattoi pyytää apua UX-suunnittelijalta joko pohtiessaan toteutusta tai toteutuksen ollessa jo käynnossä. Joissain tilanteissa UX-suunnittelija piirsi kuvan malliksi, joissain tilanteissa pohdittiin asiaa suullisesti tai keskusteltiin, onko kehittäjän jo tekemä toteutus tarpeeksi hyvä. Kehitystiimit olivat aika vähän tekemisissä asiakkaan yhteyshenkilöiden kanssa. Kehittäjät päättivät aika suurelta osin itse, miten toiminnallisuudet toteutetaan.

### 7.1.5 Projekti 5, yritys 1

**Kuvaus projektista**: Projekti on aktiivisessa kehityksessä. Asiakas on julkishallinnosta. Projektissa oli 3 kehittäjää, lisäksi projektiin on nimetty UX-suunnittelija. Osa-aikaisesti projektissa on mukana myös tekninen projektipäällikkö ja devops-asiantuntija. Haastateltavana oli kaksi kehittäjää.
**Projektimalli:** Scrum
**Projektin käytännöt:** Projektissa on käytössä kahden viikon sprintit. Uusi versio viedään tuotantoon kahden viikon välein. Kahden viikon välein käydään myös asiakaspalaverissa läpi tehdyt muutokset ja keskustellaan seuraavan sprintin aikana tehtävistä asioista. Projektin työjonoa hallitaan kanban-taulun avulla. Asiakas tai tekninen projektipäällikkö lisää tehtävät ominaisuudet taululle ja niitä pilkotaan tarvittaessa pienemmiksi kokonaisuuksiksi. Periaatteessa projektissa on käytössä päivittäiset Scrum-tapaamiset, mutta käytännössä niitä ei ole aina pidetty. UX-suunnittelija ei osallistu säännöllisesti mihinkään tapaamisiin. UX-suunnittelijaa pyydetään mukaan silloin, kun tarvitaan suurempien kokonaisuuksien suunnittelua tai ongelmatilanteissa. UX-suunnittelija tuottaa toiminnallisuudesta näyttökuvan tarvittaessa. Tarvittaessa ongelmasta keskustellaan yrityksen käyttämässä pikaviestipalvelussa.

### 7.1.6 Projekti 6, yritys 1

**Projektin kuvaus**: Ylläpitovaiheessa oleva projekti, johon kehitetään edelleen uusia ominaisuuksia. Asiakas on julkishallinnosta. Haastateltavana oli kolme kehittäjää ja ohjelmistoarkkitehti-kehittäjä.
**Ohjelmistokehitysmalli:** Scrum
**Projektin käytännöt:** Uudet ominaisuudet ja korjaukset tehdään kolmen viikon sprinteissä. Uudet ominaisuudet julkaistaan neljä kertaa vuodessa uudessa sovellusversiossa. Käytössä ovat viikottaiset Scrum-tyyppiset palaverit. Asiakas selvittää tarvittavat uudet ominaisuudet ja jakaa ne toteutettaviksi osiksi. Kehitystiimi kommentoi asiakkaan ehdotuksia ja antaa toteutettaville kokonaisuuksille työmääräarviot. Asiakas on kehitystyössä aktiivisesti mukana verkon välityksellä, kasvotusten tavataan muutaman kuukauden välein.
Projektissa ei ole mukana UX-suunnittelijaa, mutta yrityksen UX-suunnittelijoilta saa periaatteessa apua. Käytännössä UX-suunnittelijoiden apua ei pyydetä. Epäselvissä tilanteissa käyttäjien tarpeita pohditaan asiakkaan ja toisten kehittäjien kanssa. Asiakas katselmoi ja testaa tehdyt muutokset ennen tuotantoon vientiä. Joskus asiakas haluaa käytettävyyteen liittyviä muutoksia tehtäviin ennen tuotantoon vientiä. Asiakas huolehtii yhteydenpidosta loppukäyttäjiin ja käyttäjien tarpeiden selvittämisestä.

### 7.1.7 Projekti 7, yritys 4

**Projekti:** Iso julkishallinnon projekti, jossa tuotantoon ollaan menossa alle kuukauden päästä. Haastateltavana oli yksi kehittäjä. Lisätietoja käytännöistä on saatu tekstimuodossa UX-suunnittelijalta.
**Projektimalli:** Scrum
**Projektin käytännöt:** Kehittämässä on tällä hetkellä viisi kehitystiimiä. Lisäksi on yksi tukitiimi, joka vastaa devops-tehtävistä, kuten erilaisten ympäristöjen pystytyksestä. Varsinaista graafikkoa ei ole kehityksessä mukana eikä palvelumuotoilijoita. Projektissa on käytössä kahden viikon sprintit. Suuremmat kokonaisuudet on kuvattu eeppisinä tai feature-tason ominaisuuksina. Ne yleensä on pilkottu valmiiksi käyttäjätarinoiksi, joista voidaan valita sprintille sopivat tehtävät toteutettavaksi. Joskus työjonoon päätyy feature-tason ominaisuuksia, jotka tiimi pilkkoo itse ja määrittelee tarkemmin. Tuoteomistaja tuottaa raakaversion kehitystiimin työjonosta, jossa käyttäjätarinat ovat priorisoituna. Työjonoa tarkistetaan kehitysjonon työstöpalavereissa, jotta työjonon järjestys pysyy ajan tasalla ja jotta työjonossa on tarpeeksi sopivan kokoisia käyttäjätarinoita. Joskus työjonon ominaisuuksia täytyy pilkkoa tai tarkentaa. Tuoteomistaja on mukana kehitysjonon työstöpalavereissa, jolloin saadaan varmistettua, että tiimi ymmärtää oikein työjonolla olevat käyttäjätarinat. Tiimiin kuuluu yksi testaaja ja kuusi kehittäjää. Scrum-master tekee myös testausta, jos jää ylimääräistä aikaa.

Projektiin on resursoitu kaksi UX-henkilöä, jotka tekevät tiivistä yhteistyötä tuoteomistajan kanssa omassa erillisessä tiimissään. Lisäksi joka Scrum-tiimistä on valittu yksi UX-lähettiläs. Joka sprintissä on yhteispalaveri UX-henkilöiden ja UX-lähettiläiden välillä. UX-lähettiläinä heidän roolinsa on pitää UX:ä mukana tiimin työskentelyssä. Heidät on koulutettu esimerkiksi tekemään heuristista asiantuntija-arviointeja ja piirtämään rautalankoja. UX-lähettiläiden ei ole tarkoitus korvata UX-suunnittelijoita vaan helpottaa UX-suunnittelijoiden aiheuttamaa pullonkaulaa projektityössä. UX-lähettiläiden tehtävänä on myös varmistaa, että UX-suunnittelijoita, jotka omistavat käyttöliittymän, konsultoidaan tarvittaessa. Näin UX saadaan sovitettua luontevasti osaksi jokapäiväistä kehitystyötä.

Käytössä ei ole mitään varsinaista UX-mallia, mutta käytetyt periaatteet on suunniteltu Lean UX:n pohjalta. Ne ovat varhainen asiakkaan validointi, osallistava poikkitieteellinen design, käyttäjien ongelmien ratkaisu, avainmittareiden (KPI) havainnointi, työkalujen joustava sovellutus ja ketterä design. UX-suunnittelussa pyritään myös opi-rakenna-testaa–sykliin. UX-tiimin työ rytmittyy muiden kehitystiimien kanssa samoihin kahden viikon sprintteihin. Sprintin ensimmäinen maanantai on suunnittelupäivä, jolloin UX-tiimi suunnittelee oman sprinttinsä samaan aikaan, kun kehitystiimit suunnittelevat omat sprinttinsä. Suunnitteluiden jälkeen UX-tiimi käy jokaisen tiimin UX-lähettilään kanssa läpi, mitä työtä tiimeille on suunniteltu. Jos kehitystiimien työjonoon on suunniteltu asioita, joihin tarvitaan UX-tiimiltä työtä, lisätään nämä UX-tiimin sprintille. UX-työlle on kokeiltu tehdä omat tikettinsä, mutta ratkaisua ei saatu toimimaan, vaan UX-työ suunnitellaan suuremmissa kokonaisuuksissa.

Tiimi tuottaa itse omia rautalankamallejaan, joko UX-tiimin kanssa yhteistyössä tai itsenäisesti. UX-suunnittelija kuitenkin katselmoi rautalankamallit. Kun tiimi aloittaa itse toteutusvaiheen, pidetään UX-suunnittelijoiden kanssa lyhyitä pikademoja. Toisin sanoen tiimi esittelee, miltä kehitteillä oleva ominaisuus näyttää ja sen jälkeen mietitään, onko suunta oikea. Kaikkia uusia ominaisuuksia näytetään loppukäyttäjille ja heiltä kerätään palautetta. Parhaimmassa tapauksessa jo rautalankoja on käyty testaamassa loppukäyttäjillä.

Toisaalta tiimi ei tuota kaikkia rautalankamalleja vaan osa rautalangoista tulee tiketin mukana. Jos malli jostain syystä puuttuu, kehitystiimi saattaa pyytää sitä UX-henkilöltä. Samoin UX-henkilö hoitaa tarvittaessa puuttuvat ikonit.

Projektissa on käytössä päivittäiset Scrum-tapaamiset ja kehitysjonon työstöpalavereja kerran sprintissä. Sprintin alussa pidetään pikainen kaikkien kehitystiimien yhteinen suunnittelupalaveri, jossa koordinoidaan eri kehitystiimien toimintaa. Retrospektiivit pidetään lähes joka sprintin jälkeen. Sprintin lopussa eri kehitystiimit esittelevät projektin väelle, että mitä on saatu aikaiseksi. Paikalla ovat kaikki kehitystiimit ja UX-henkilöt. Asiakkaalle pidetään demo joka toisessa sprintissä.

### 7.1.8 Yhteenveto eri projektien käytännöistä

Kahdessa projektissa oli käytössä SAFe, kahdessa Kanban ja kolmessa projektissa oli käytössä Scrum. Käytännössä jokaisella projektilla oli kuitenkin oma sovellettu tapansa käyttää projektimalleja. Yksi haastateltava totesi, että heidän projektissaan on otettu mukaan jotain SAFesta, mutta muuten mennään ihan Scrumin mukaisesti. Toisaalta toinen projekti, sisälsi piirteitä SAFesta, vaikka kehitystiimissä puhutaan Scrumista. Tutkimuksen projektit oli rajattu ketterää kehitystä noudattavaksi. Haastateltavia etsittäessä tämä ei ollut ongelma. Vaikuttaa siltä, että ketterä kehitys on nykyään normi, toteutettuna ainakin jollain tasolla. Yksi kehittäjä kuitenkin totesi, että heidän projektityönsä on järjestelty Scrumin mukaan, mutta vaatimusmäärittely ja sopimus on tehty vesiputousmallin mukaisesti. Tästä oli seurannut välillä käytännön ongelmia, koska asiakkaan kehityssyklissä hyväksymät ominaisuudet eivät olleetkaan välttämättä sopimuksessa kirjatun mukaisia.

Haastateltavista projektitiimeistä kahdessa oli mukana kiinteästi UX-suunnittelija. Näissä projekteissa UX-suunnittelija oli kiinteässä vuorovaikutuksessa niin tuoteomistajaan kuin kehittäjiin. Yhdessä projektissa UX-suunnittelu tehtiin omassa tiimissään. Se, kuinka kiinteästi UX-suunnittelija osallistui kehitystiimien toimintaan, riippui tiimistä. Kahdessa projektitiimissä oli nimetty UX-suunnittelija, mutta UX-suunnittelija ei ollut kiinteästi osa kehitystiimin toimintaa. Näissä tapauksissa kehitystiimi tai projektipäällikkö pyysi UX-suunnittelijan apua tarvittaessa. Yhdessä projektissa käytettiin koko yrityksen UX-suunnittelijaa. Tässä projektissa kehittäjät ratkoivat yleensä esiin tulevat ongelmat keskenään tai arkkitehdin kanssa. Joskus isompiin käyttöliittymämuutoksiin haettiin apua myös UX-suunnittelijalta. Yhdessä projektissa ei ollut ollenkaan UX-suunnittelijaa mukana. Tässä projektissa kehittäjä teki yleensä ratkaisut itsenäisesti. Mielipidettä kysyttiin asiakkaalta tai toisilta kehittäjiltä.

Vain yhdessä projekteista tehtiin jatkuvasti palvelumuotoilua säännönmukaisesti. Toisessa projektissa UX-suunnittelija teki myös silloin tällöin jonkin verran palvelumuotoilua, mutta sen määrä oli jäänyt hänen mielestään vähäiseksi. Kolmannessa projektissa UX-suunnittelijat tekivät palvelumuotoilua tarpeen vaatiessa. Yhdessä projektitiimissä oli ollut aikaisemmin UX-suunnittelijan lisäksi myös graafikko. Jos projektissa oli UX-suunnittelija, graafinen suunnittelu oli usein UX-suunnittelijan vastuulla.

Käytetyllä projektimallilla ei havaittu yhteyttä siihen, oliko projektissa mukana UX-suunnittelijaa. Tapa, jolla UX-suunnittelu oli yhdistetty ketterään kehitykseen, vaihteli projekteittain. Työn teoriaosuudessa esitettiin malli, jossa UX-suunnittelu integroitiin ketterään kehitykseen rinnakkaisten UX-sprinttien avulla. Vain yhdessä tutkittavista projekteista oli käytössä erilliset UX-sprintit. Yhdessä projektissa UX-sprinttiä oli kokeiltu, mutta siitä oli luovuttu. Pääsääntöisesti UX-suunnittelu pyrittiin tekemään ennen toteutuksen aloittamista. Tähän ei kuitenkaan aina päästä. Haastateltavan sanoin: **“Käytännössä pamua (palvelumuotoilua) ja UX:ää tehdään myös (tyyliin) piti olla eilen speksi, tänään alkaa toteutus. “** Tarkkoja sprinttimääriä, kuinka paljon aikaisemmin

jonkin tehtävän suunnittelu aloitetaan, ei ollut. Haastatteluissa kävi myös ilmi, että välillä tehtävän laajuus tulee yllätyksenä vasta toteutuksen alkaessa. Tällöin myös UX-suunnittelu aloitetaan vasta jälkijunassa. Toisaalta haastateltavat kertoivat myös tapauksista, jossa UX-suunnittelijan mielipidettä oli kysytty vasta valmiista toteutuksesta.

## 7.2 Käyttäjäkokemuksen suunnittelu yrityksessä 1

Yritys on keskisuuri suomalainen ohjelmistoyritys, jolla on toimintaa myös ulkomailla. Yritys tarjoaa palveluja digitaalisen muutoksen johtamiseen, palvelujen muotoiluun, toteutukseen ja pilvipalveluja. Yrityksellä on vahva osaaminen palvelumuotoilusta ja UX-suunnittelusta.

### 7.2.1 Kokemuksen suunnittelu on kerroksittaista

Käyttäjän kokemus rakentuu useissa eri kerroksissa tehdyistä suunnitteluratkaisuista, joissa siirrytään suuremmista kokonaisuuksista pienempiin yksityiskohtiin. Eräs haastateltava kertoi kuinka muotoilussa siirrytään suuremmista kokonaisuuksista suunnittelemaan pienempiä yksityiskohtia. Liiketoiminnan muotoilu on arvolupauksen muodostamista ja arvoketjujen muodostamista liiketoiminnan näkökulmasta. Siinä mietitään, mistä yrityksen tulovirrat ja arvo muodostuvat. Asiakaskokemusta mietitään jo liiketoiminnan muotoilun aikana, mutta sitä mietitään enemmän toimijaverkoston kautta, ketä toimijoita on jo ja ketä toimijoita tarvitaan, jotta kokonaisuuden voi muodostaa parhaalla tavalla. Liiketoiminnan muotoilusta seuraava kerros on palvelumuotoilu. Suurin osa palvelumuotoilusta on palvelukonseptin muodostamista eli ei keskitytä sovellukseen vaan pohditaan käyttäjän kokemusta koko palvelusta. **”Palvelumuotoilussa keskeinen asia on sen ymmärtäminen, (että) se on yhteistä tekemistä. Se käyttäjä on koko ajan siinä mukana.”**

Palvelumuotoilusta suurin osa tehdään ennen toteutusprojektin aloittamista. Ideaalitapauksessa palvelumuotoilua kuitenkin jatketaan tuotteen kehityksen aikana ja palvelumuotoilu jatkuu edelleen tuotteen jatkokehityksen aikana. Kun palvelumuotoilua tehdään samaan aikaan kuin toteutusta, palvelumuotoilu nivotaan muuhun projektityöhön ja toteutettavien kokonaisuuksien palvelumuotoilu tehdään ennen kuin aletaan pohtia kuinka uudet ominaisuudet käytännössä toteutetaan sovellukseen. Palvelumuotoilu siis tehdään ennen UX- ja UI-suunnittelua.

Se, että kuinka pitkään palvelumuotoilua projektissa tehdään, riippuu hyvin paljon asiakkaasta. Palvelumuotoiluun ei välttämättä ymmärretä varata alun jälkeen rahaa. Lisäksi haastateltava toteaa, että **”jos halutaan säästää, niin asiakaskokemus tai käyttäjäkokemus on sellaista, mistä on helppo säästää, sen lisäarvoa on vaikea mitata, koska se tulee aina tavallaan viiveellä.”**

Palvelumuotoilun lopputulokset pitää saada viestittyä UX- ja UI- suunnitteluun. **”Palvelumuotoilun tuloksena täytyy tulla ne suunnitteluajurit, eli sieltä täytyy irro-**

**ta ne keskeisimmät asiat, mitkä on tärkeitä, sitten kun se UI-suunnittelija rupee miettimään sitä UI:ta."** Palvelumuotoilijan ja UX-suunnittelijan tärkein viestintäkeino on keskusteleminen, mutta dokumentaatiota käytetään myös. Dokumentaatiossa voidaan esimerkiksi jakaa asiakkaan kulkeminen palvelussa eri toimijoittain. Se, missä palvelumuotoilun raja loppuu ja käyttäjäkokemuksen suunnittelu alkaa, on tulkinnanvaraista. Rajaa hämärtää se, että osa yrityksen palvelumuotoilijoista tekee myös UX-suunnittelua. Käyttäjäkokemuksen suunnittelussa mietitään kuitenkin nimenomaan sitä, miten käyttäjä toimii toteutettavassa sovelluksessa, kun palvelumuotoilussa mietitään palvelua kokonaisuudessaan. Käyttöliittymän suunnittelu on viimeinen suunnittelun taso, jossa päätetään konkreettisesti se, miltä käyttöliittymä näyttää. Yrityksen projekteissa oli harvoin mukana graafikkoja. Jos projektissa oli mukana UX-suunnittelija, UX-suunnittelija vastasi yleensä myös UI-suunnittelusta. Toisaalta myös kehittäjät tekivät paljon etenkin pienempiä UI-ratkaisuja.

### 7.2.2 UX-suunnittelijoiden mukanaolo projektin elinkaaren aikana

Haastatteluissa kävi ilmi, että UX-suunnittelijan mukanaolo yrityksen projekteissa, joissa kehitetään uutta tuotetta, on lähes itsestään selvää. Erityisen tärkeä UX-suunnittelijan panos on projektin esiselvitysvaiheessa. Esiselvitysvaiheeseen pyritään saamaan mukaan joku, jolla on projektiteknistä osaamista, UX-suunnittelija ja joku, jolla on teknistä ymmärrystä. Tekninen asiantuntija voi olla esimerkiksi ohjelmistoarkkitehti. Projektipäällikkö toteaa, että hän ei suostu tekemään yhtään esiselvitystä ilman UX-suunnittelijoita: "… **koska niillä on se käyttäjän näkemys koko ajan niin voimakkaasti. Ihan ehdottomasti, ne on ihan äärettömän tärkeitä ihmisiä siinä esiselvitysvaiheessa.**"

Erityisesti käyttäjäkokemukseen vaikuttavien ratkaisujen tekemisen tärkeys korostui alkuvaiheessa olevissa projekteissa. Projektipäällikön mukaan "**esiselvityksen, määrittelyn ja parin ekan sprintin aikana muodostuu noin 75% UX:stä.**" Loput UX:stä on "**enemmän sellasta hienosäätöä ja sellasta, että aina siinä toteutuksen aikana nousee asioita, joita ei oo mietitty etukäteen tai nousee puskasta joku uus juttu".**

UX-suunnittelijat tekevät töitä yleensä useammassa kuin yhdessä projektissa. Yksikään haastatteluihin osallistuneista UX-suunnittelijoista tai palvelumuotoilijoista ei ollut vain yhdessä projektissa. Kuten haastateltava totesi: **"Ne (UX-suunnittelijat) tekee montaa (projektia) päällekkäin joka tapauksessa. Harvoin ne pääsee yhtä projektia tekeen."** Kaikki niistä haastatelluista projektitiimeistä, joissa oli mukana UX-suunnittelija, kertoivat tilanteista, joissa UX-suunnittelija oli ollut projektissa pullonkaulana. Toisaalta UX-suunnittelijoilla ei ollut toisia UX-suunnittelijoita työpareinaan. Eräs haastateltava totesi: **"Käytännössä kun on ollut yksin UX-suunnittelijana, niin tulee sokeaksi sille mitä tekee. Ei mun mielestä ikinä sitä UX-suunnittelua pitäis tehdä yksin."**

UX-suunnittelijoita on harvoin resursoitu ylläpitoprojekteihin. **“Harvoin niihin sitten enää ketään nimetään, koska se koetaan silleen, että kun se on kumminkin valmis tuote, niin ajatellaan, että ne muutokset on kuitenkin niin pieniä. Tai ne koetaan niin pieniks ettei tarvis”.** Kehityksen hidastuessa ja tehtävien pienentyessä mahdolliselle UX-suunnittelijalle jää myös vähemmän tehtävää. Kehittäjä ylläpitoprojektista toteaa: **“Mä sanoisin, että meillä on niin pieniä asioita, mitä tässä tehdään, tipoittain tätä kehitystä, että mä kaipaisin siihen mitään ylimääräistä (UX-kehittäjää)**.” Osittain tämä johtunee myös siitä, että projektin edetessä kehittäjien on helpompi tehdä UX-ratkaisuja itsenäisesti vanhan toteutuksen pohjalta. **S**aman projektin kehittäjät tosin toteavat myös, että on ollut tilanteita, joissa UX-suunnittelijan apua olisi tarvittu.

Jos ylläpitoprojektiin tulee uusia isompia asioita, ne voidaan toteuttaa ylläpitoprojektin osana. Usein kuitenkin halutaan luoda uusi projekti tehtäviä muutoksia varten. Uusien isompien kokonaisuuksien toteutukseen haluttaisiin usein saada mukaan sama UX-suunnittelija kuin projektissa on alkuvaiheessa ollut. **“Toisaalta, kun lähdetään niin kuin isompia kokonaisuuksia muuttamaan, niin se ongelma on ehkä lähinnä meidän näkökulmasta se, että se (UX-suunnittelija) on jo resursoitu muualle. Sillä ei ole enää aikaa käytettävissä ja siitä tulee se haaste. Ei niinkään siitä, ettei sitä haluttaisi käyttää, vaan siitä, ettei sillä ole enää aikaa käytettävissä. “**

### 7.2.3 UX-suunnittelijoiden ja kehittäjien välinen yhteistyö

UX-suunnittelijat kommunikoivat kehittäjien kanssa tyypillisesti käyttäen apunaan erilaisia näyttökuvia tai rautalankapiirroksia. Yleinen käytäntö on tehdä ominaisuuksista vain näyttökuva, eikä muuta erillistä dokumentaatiota: **“Me ei tehdä näistä mitään speksejä, vaan kuvat. Sinne jää kuitenkin aika paljon sellasia aukkokohtia”**. Lisäksi jo tehtyjä ratkaisuja käydään läpi ja niitä käytetään keskustelun pohjana pohdittaessa uusien ominaisuuksien lisäämistä.

Jos projektissa ei ollut graafikkoa, UX-suunnittelija vastasi myös graafisesta suunnittelusta. Tällöin UX-suunnittelijan tekemät näyttökuvat olivat valmiimpia kuin silloin, jos projektissa oli myös graafikko. UX-suunnittelija saattoi kokea graafisen suunnittelun ongelmallisena: “**Sehän se meillä on tässä ongelmana, kun meillä ei oo (projektissa) graafikkoa. En koe itseäni graafikoksi millään lailla. // Ne visuaaliset leiskat (layout) ja ratkaisut tuottaa välillä vähän haasteita. //Se ehkä hidastaa tätä työtä ja jättää myös koodarillekin silleen, että mikäs tässä nyt olis hyvä. “**

Toisaalta projektissa, jossa oli aiemmin ollut graafikko, graafikon ja UX-suunnittelijan yhteistyön koettiin myös poistavan projektin pullonkauloja: **“Se oli toisaalta hirveen kiva kuvio, että se antoi sellasta joustavuutta, että jos joku on poissa päivän tai kaks niin se ei viivästytä. Nythän (UX-suunnittelija) oikeestaan vastaa siitä visuaalisesta ilmeestä. Siinä mielessä asiat tulee valmiimmin pureskeltuna, mutta yhden ihmisen takaa. “** Graafikko ja UX-suunnittelija pystyivät siis jossain määrin korvaamaan toisiaan projektissa.

Myös kehittäjän UX-osaaminen vaikutti UX-suunnittelijan ja kehittäjän väliseen yhteistyöhön. UX-suunnittelijat ymmärtävät, että kehittäjien UX-osaamisessa on eroja. Lisäksi UX-suunnittelija saattoi tiedostaa, minkälaista tukea kehittäjä tarvitsi. Kokeneille kehittäjille annettiin vähemmän ohjeita ja heidän työtään valvottiin vähemmän. **“(Jos on hyvä fronttikoodari) niin pystyy enemmän luottamaan.// Ja tietää, että tolle mä en tarvii täydellistä kuvaa, kun se osaa kyllä. Mutta sitten, kun on vaikka näitä koulunpenkillä olevia, jotka tekee ensimmäisiä käyttöliittymiä, niin niitä joutuu paljon enemmän vahtaamaan.”** Toisaalta kehittäjä pystyy arvioimaan myös UX-suunnittelijan tekemien ratkaisujen järkevyyttä. Haastateltavan mukaan hyvä frontend-kehittäjä pystyy kyseenalaistamaan UX-suunnittelijan ratkaisut ja ehdottamaan parempia vaihtoehtoja.

Vaikutti siltä, että mitä tiiviimmin kehittäjät ja UX-suunnittelija ovat integroituneet tiimiksi, sitä enemmän kommunikaatiota kehittäjien ja UX-suunnittelijoiden välillä oli. Vastaavasti UX-suunittelijan ratkaisuja myös kyseenalaistettiin ja haastettiin enemmän. Eräs UX-suunnittelija ajatteli: **“On sekä hyvä asia, että huono asia, että aina kyseenalaistetaan. Onhan se nyt ihan jäätävän rasittavaa, että joka kerta kyseenalaistetaan... Hirmu usein niillä on hyviä pointteja, joita mä en ole ajatellut. Eihän se muuten olis yhteistyötä // Mä haluan mielipiteen, mä aina kysyn.”**

Toisaalta UX-suunnittelijan persoonallisuus ja tapa työskennellä vaikuttaa myös ryhmän dynamiikkaan. Tämä saattaa näkyä myös siinä, että UX-suunnittelijan vaihtuessa myös UX-suunnittelijan ja muun tiimin välinen vuorovaikutus voi muuttua. Eräässä projektissa todettiin: **“siinä vaiheessa mulle ei tullut yhtään mieleen, että (uusi UX-suunnittelija) tekis asioita eri tavalla kuin (vanha UX-suunnittelija). Myös se dynamiikka meni uusiks, et sit sieltä tuli vähän sellasia yllätyksiä.** “

## 7.3 Kehittäjien rooli UX-ratkaisuissa

Kehittäjiä haastatteluihin osallistui 12. Näistä 9 oli yrityksestä 1 ja loput muista yrityksistä. Haastatteluun osallistuneilta kysyttiin, millaisia käyttäjän kokemukseen vaikuttavia päätöksiä he tekevät. Kehittäjien vastaukset ovat nähtävillä taulukossa 7.3.1. Taulukossa on yhdistelty useammin kuin kerran esiintyneitä vastauksia. Useimmiten kehittäjät sanovat tekevänsä käyttöliittymäratkaisuja tai käyttöliittymän määrittelyä.

Haastateltavilta kysyttiin myös, mitä käyttäjäkokemus on. Näin haluttiin selvittää, kuinka hyvin haastateltavat ymmärtävät, mistä ovat puhumassa. Kehittäjät tietävät hyvin, mitä käyttäjäkokemus on. Yksikään kehittäjä ei ollut ymmärtänyt termiä väärin tai jättänyt vastaamatta kysymykseen. Oli havaittavissa, että kehittäjät, jotka olivat olleet tiiviissä yhteistyössä UX-suunnittelijan kanssa, kuvasivat käyttäjäkokemuksen monipuolisemmin kuin muut. Tarvittaisiin kuitenkin jatkotutkimuksia, jotta tämä pystyttäisiin varmistamaan ja ymmärtämään ilmiön syitä.

**Taulukko 7.3.1** Kehittäjien tekemät ratkaisut, jotka vaikuttavat heidän omasta mielestään käyttäjäkokemukseen.

| |
|---|
| Käyttöliittymäratkaisuja |
| Käyttöliittymän määrittelyä |
| Sopivien UI-elementtien valintaa valmiista joukosta |
| Tekstien valintaa annetun asettelun kanssa |
| Taustaprosessien palautteen suunnittelua |
| Tuon esiin visuaalisiin ratkaisuihin ja käytetttävyyteen vaikuttavia seikkoja |
| Kerron teknisten ratkaisujen rajoitteista ja mahdollisuuksista, varsinaiset ratkaisut tekee usein joku muu |
| Pyrin vaikuttamaan keskustelemalla lopputuloksiin |
| Miten tietoja täytetään, miten ne esitetään |
| Pienempiä suunnittelu- ja toteutustaskeja itsenäisesti |
| Kuinka hyvin ohjelmistosta saa tehtyä nopean, vikasietoisen ja varman |
| Torppaan UX-suunnittelijoiden visioita |

Millaisissa asioissa kehittäjät sitten kysyvät neuvoa UX-suunnittelijoilta? Pieniä käyttöliittymäratkaisuja ja toiminnallisuuksien määrittelyä kehittäjät tekevät paljon itsenäisesti. Tyypillisesti, kun UX-suunnittelijaa käytettiin, kyseessä oli joku isompi käyttöliittymäkokonaisuus. UX-suunnittelija ja kehittäjät kommunikoivat keskenään 1) näyttökuvien tai rautalankamallien avulla, 2) keskustelemalla ja 3) valmiita toteutuksia tutkimalla. Jos tapaus oli selkeä, kehittäjä ja UX-suunnittelija eivät välttämättä keskustelleet tapauksesta ollenkaan keskenään. Valmis käyttöliittymäkuva liitettiin mukaan tikettiin ja kehittäjä teki toteutuksen sen pohjalta. Tapaukset, joissa kehittäjä kertoi olleen epäselvyyksiä, liittyivät tyypillisesti toimintalogiikkaan. Tämä on ymmärrettävää. Kehittäjän on pakko ottaa kantaa toimintalogiikkaan ja toteuttaa se yksiselitteisesti.

Epäselvissä tapauksissa apua ei välttämättä kysytty UX-suunnittelijalta. Eräässä projektissa oli tapana kysyä mieluummin neuvoa toisilta koodareilta tai arkkitehdeiltä. Syyksi haastateltu kertoi, että hänellä oli hyvin korkea kynnys kysyä apua yrityksen yhteiseltä UX-suunnittelijalta. Toisaalta eräs haastateltava kertoi ehtineensä toteuttaa jo toiminnallisuuden, ennen kuin UX-suunnittelija oli ehtinyt muilta töiltään tarttua asiaan. Useat kehittäjät kokivat hankalaksi kysyä pieneksi koettuja ongelmia UX-suunnittelijalta. Toisaalta tästä oli myös poikkeuksia. Pääsääntöisesti vaikutti siltä, että mitä kiinteämmin UX-suunnittelija osallistui tiimin toimintaan, sitä luontevampaa yhteistyö oli. Eräs kehittäjä totesi, että on matala kynnys pyytää UX-suunnittelijalta apua, **”kun se istuu siinä kolmen tuolin päässä”**. Havaittiin, että pelkkä UX-suunnittelijan nimeäminen projektiin, ei saa kehittäjiä kysymään UX-suunnittelijalta neuvoa. Vaikka UX-suunnittelija ei osallistuisi kovin kiinteästi projektiin, kynnyksestä yhteistyöhön voidaan teh-

dä silti matala. Se, millainen kulttuuri projektissa ja laajemmin organisaatiossa onnistutaan luomaan, vaikuttanee tähänkin asiaan.

Erikoisena piirteenä havaittiin se, että kolmessa projektissa, UX-suunnittelija ei ollut paljoa tekemisissä asiakkaan tai käyttäjien kanssa. Kehittäjien yhteistyö heidän kanssaan oli tiiviimpää kuin UX-suunnittelijan. Samoin oli luonnollisesti myös projektissa, jossa ei ollut UX-suunnittelijaa ollenkaan. Näissä projekteissa siis kehittäjät päättivät UX-suunnittelun tarpeen. Ylipäätään kehittäjät kunnioittivat paljon asiakkaan toiveita ja pitivät asiakkaan tyytyväisyyttä yhtenä mittarina onnistumiselle.

Kun kysyttiin, millä perusteella kehittäjien tekemät ratkaisut tehdään, asiakkaan mielipide nousi usein ensimmäisenä esiin. Toinen usein esille noussut tapa oli yrittää tehdä olemassa olevan toteutuksen kanssa yhdenmukaisia ratkaisuja. Eräs kehittäjä sanoi kauniisti, että koodi on kehittäjän käyttöliittymä ja hän pyrkii tekemään käyttöliittymäratkaisuja samoin perustein kuin kirjoittaa luettavaa koodia. **”Ja sama tulee koodissa vastaan itselle: mä teen tän samaan tyyliin mitä tää on aina ennenkin tehty, siis samalla rakenteella. Ihan vaan sen takia, että tää on sille seuraavalle lukijalle tuttu ratkaisu, enkä mä lähde tekemään mikroskooppisesti parempaa ratkaisu. ”** Jotkut kehittäjät käyttivät apunaan myös käytettävyydestä opinnoissaan saamaansa tietoa, mutta kaikki kehittäjät eivät olleet saaneet mitään alan koulutusta. Mahdollisuutta arvioida tehtyjä ratkaisuja asiakkaan kanssa pidettiin yleisesti ottaen hyvin tärkeänä. Asiakkaan hyväksynnällä ja tyytyväisyydellä myös oikeutettiin tehdyt ratkaisut. Useampi kehittäjä uskoi, että käyttäjäkokemus oli hyvä, koska asiakas oli ollut tyytyväinen tai kukaan ei ainakaan ollut valittanut. Havainnot ratkaisujen teon perusteista voi tiivistää erään haastateltavan sanoin: **”Pyrin menemään mahdollisimman pitkälle sen mukaan, mitä asiakas on sanonut. Silloin kun ne ei vaivaudu sanomaan kovin tarkasti tai ei itsekään tiedä, niin tässä on hyvä katsoa niitä olemassa olevia standardeja ja komponentteja, että miten niitä on käytetty. Tavallaan vaikka mulla ei varsinaista kompetenssia olekaan, niin yritän katsoa siitä UX- näkökulmasta, että mikä olis se järkevä tapa, jos on sellaisia järkeviä komponentteja, mitä ne ei oo ite pohtinut. Ja sit tavallaan hyödynnän sitä mahdollisuutta iterointiin, että jos niillä on ajatuksia, niin ne saa tarkentaa sitten.”**

Se, kuinka paljon tai kuinka mielellään kehittäjät tekevät käyttöliittymään vaikuttavia ratkaisuja, vaihtelee kehittäjän mielenkiinnonkohteiden ja taitojen mukaan. Osa haastatelluista kehittäjistä oli selkeästi suuntautunut tekemään backend-ratkaisuja ja osa frontend-ratkaisuja. Joukossa oli myös melkoisia monitaitureita, jotka työskentelevät hyvin laajalla toimenkuvalla. Joku kehittäjä saattaa sanoa aika suoraan, ettei hän ole kiinnostunut käyttöliittymästä. Osa taas piti määrittelytyöstä laajemmin tai ilmaisi halunsa osallistua käyttöliittymäratkaisujen tekemiseen **“Mutta kyllä mä tykkään, että saa itekin sanoo oman mielipiteensä, en mä tykkäis olla sellasessa projektissa, jossa tulis vaan joku leiska.“**

Kuitenkaan kukaan kehittäjä ei tuonut esille, että UX-suunnittelua tehtäisiin projektissa liikaa. UX-suunnittelijoita pidetään tärkeinä ja kehittäjät kertoivat esimerkkitapauksia, joissa UX-suunnittelijaa olisi kaivattu projekteissa, joissa he olivat työskennelleet. Koettiin, että jos UX-suunnittelua ei ole projektissa tarpeeksi, tekniikka alkaa sanella liikaa tehtyjä ratkaisuja ja ratkaisuissa mennään liikaa teknisen toteutuksen helppouden perusteella. Yksi kehittäjä kuvaili, että kun **koodarit tekee käyttöliittymiä, eikä oo niin kauheesti mitään suunnitelmiakaan, siitä tulee sellanen insinööriltä insinöörille hässäkkä”**. Toinen kertoi sokeutuneensa sille, miltä käyttöliittymä näyttää, koska **”sitä alko aatteleen liikaa sitä niinku sen koodaamisen kautta ja kaiken sen työn kautta, että mitä nyt menis tohon vaikka (aikaa)”**.

Kehittäjät ovat yksi porras, joka validoi UX-suunnittelijan tekemiä ratkaisuja. Tätä tehdään kommentoimalla suunnitelmia ja kertomalla teknisistä rajoitteista, joita voi olla. Kehittäjien tehtäviin kuuluu myös eri vaihtoehtojen työmäärän arviointi. Työmäärä saattaa vaihdella yllättävästi riippuen käytetystä tekniikasta. Teknisiä rajoitteita voi olla erityisesti vanhemmissa sovelluksissa, joissa on käytetty vanhentuneita tekniikoita ja joissa jo tehdyt toteutukset rajoittavat sitä, mitä voi tehdä. UX-suunnittelija tai asiakas eivät pysty arvioimaan ohjelmiston teknisiä yksityiskohtia. Tämän vuoksi teknisten asiantuntijoiden pitää myös ymmärtää oman toimintansa taustoja. Kehittäjällä voi olla mielipide tietyn ratkaisun paremmuudesta. Pitää kuitenkin varoa perustelemasta ratkaisua teknisesti, silloin kun mielipide ei johdu teknisistä syistä. Kuten eräs haastateltava sanoi: **”Kun oot ite jotain mieltä ja alat jotain selittämään teknisillä rajoitteilla… pitää tunnistaa, että ite yrittää ohjata sitä johonkin suuntaan. Työmääräarvio voi vaihdella kehittäjän mielipiteen mukaan aika paljon... Aika tärkeää sellainen itsereflektio.”**

# 8 TUTKIMUSPROSESSIN ANALYSOINTI

Aliluvussa 7.3 tarkasteltiin yrityksen käytäntöjä, mutta muutoin näkökulmaksi valittiin ilmiön tarkastelu projekteittain. Pidän tätä hyvänä näkökulmana. Jos haastatteluja tehdään yksittäisille ihmisille koko yrityksen käytännöistä, saadaan helposti todellisuutta ruusuisempi kuva käytännöistä. Parhaiten toteutettuja projekteja käytetään herkästi esimerkkeinä ja unohdetaan ne projektit, joissa on puutteita. Toisaalta näkökulma aiheutti myös ongelmia. Tutkimus olisi vaatinut vielä laajemman haastattelupohjan. Lisäksi jo ensimmäisessä haastattelussa kävi ilmi, että projektikäytäntöjä on vaikea selvittää haastattelutilanteessa ja että tarkka selvitys veisi arvokasta aikaa muilta tutkimuskysymyksiltä. Tässä päädyttiin ratkaisuun, että projektin käytännöistä yritettiin saada nopeasti kuva haastattelun aikana. Jos projektin käytännöistä jäi kysyttävää, esitettiin tarkentavia kysymyksiä projektista haastattelun jälkeen viestisovellusten avulla. Haastatteluissa törmättiin ilmiöön, että haastateltava ei tuntenut projektin kaikkia käytäntöjä. Tämä oli sinänsä itsessään mielenkiintoinen havainto. Henkilöillä ei välttämättä ole kokonaiskuvaa projektin toiminnasta. Näissä tapauksissa toimittiin tilannekohtaisesti. Yhdessä tapauksissa tieto katsottiin merkityksettömäksi, yhdessä tapauksessa haastateltavaa pyydettiin selvittämään asiaa ja yhdessä tapauksessa haastateltava selvitti asian omatoimisesti.

Vaikka tavoitteena olikin muodostaa aiheesta yleiskuva, tutkimuksen kannalta projekteja olisi voitu rajata tarkemmin. Rajauskriteeri olisi voinut olla esimerkiksi käytetty projektimalli. Tosin on kyseenalaista, olisiko tällä ollut vaikutusta, koska projektimallin ja UX-suunnittelun toteutuksen välillä ei havaittu tuloksissa yhteyttä. Toinen hyvä rajauskriteeri olisi ollut projektin koko. On selvää, että 70 hengen projektiin sopivat erilaiset käytännöt kuin projektiin, jossa on vain kourallinen kehittäjiä. Toisaalta työ antaisi laajan näkökulmansa vuoksi hyvän pohjan tarkempien jatkotutkimuksien tekemiseen.

Haastatteluissa havaittiin, että niissä oli tärkeä luoda rauhallinen, kiireetön ilmapiiri, jossa haastateltava sai kertoa mielipiteensä. Havaittiin, että hyvä haastattelija on kärkäs kuuntelemaan ja hidas puhumaan itse. Kyseessä oli poikittaistutkimus eli yhtä henkilöä haastateltiin vain kerran. On siis mahdollista, että haastatteluissa kerrottiin kokemuksia sekä saman projektin nykyisistä käytännöistä, että vanhoista käytännöistä. Tarkasteltaessa tutkimustuloksia täytyy myös huomata, että haastateltavien annettiin kertoa kokemuksiaan myös edellisistä projekteistaan. Nämä kertomukset jäivät yksittäisiksi anekdooteiksi, eikä kertomusten projekteista kyselty sen tarkempia tietoja. Saatuja kokemuksia käytettiin kuitenkin työn tuloksissa. Toinen huomionarvoinen seikka on,

että aliluku 7.3 ei ole oma itsenäinen tutkimuksensa, vaan siinä tehtiin syvempiä havaintoja samoista projekteista, joita käsiteltiin jo aikaisemmissa aliluvuista. Haastatteluissa myös muiden kuin kehittäjien määrä on roolia kohti melko pieni.

Voidaan myös kysyä, vaikuttiko se, että osa haastattelutilanteista oli ryhmähaastatteluja ja osa yksilöhaastatteluja, työstä saataviin tuloksiin. Ryhmähaastatteluissa tiimit vaikuttivat vapautuneilta toistensa seurassa ja osallistujat esittivät myös kritiikkiä. On kuitenkin mahdollista, että jotain sellaista on jäänyt sanomatta, mikä olisi tullut yksilöhaastattelussa ilmi, ja osa ryhmähaastattelussa esiin nousseista kysymyksistä olisi jäänyt yksilöhaastattelussa käsittelemättä. Ero haastattelujen muodossa on huomioitu niin, että tulosten analysoinnissa on oltu tarkkana projektien vertaamisessa toisiinsa. Projektien vertailussa on muutenkin syytä olla varovainen, koska projektit eivät olleet samassa tilanteessa. Toisissa kehitystyö oli aktiivisempaa kuin toisissa. Projektien ja yritysten määrä tutkimuksessa on ollut kohtalaisen pieni ja on mahdollista, että suuremmalla otoksella oltaisiin saatu erilaisia tuloksia. On kuitenkin huomattava, että myös alan kirjallisuudessa osa tutkimuksista käyttää melko pieniä otoksia.

Luvussa 6 esiteltiin laadullisen tutkimuksen klassiset laatukriteerit luotettavuus, objektiivisuus ja oikeellisuus, joiden perusteella voidaan arvioida tämän tutkimuksen laatua. Tarkastellaan ensin luotettavuutta. Kaikki haastattelut nauhoitettiin ja litteroitiin. Litterointeihin lisättiin myös tietoja siitä, että missä kohdassa mikäkin sanottu asia oli nauhoituksessa. Tämä helpotti sitä, että oikea kohta nauhoituksesta oli helppo tarvittaessa kuunnella uudelleen. Nauhoituksella parannettiin tutkimuksen luotettavuutta. Myös avoin koodaus otettiin mukaan tulosten käsittelyyn, jotta tuloksista tulisi luotettavampia.

Työn objektiivisuutta parantaa se, että työtä ei ole tehty millekään organisaatiolle. Tekijä ei myöskään ole pyrkinyt ajamaan mitään tiettyä asiaa työtä tehdessään. Haastattelijoita työssä oli vain yksi. Koska haastateltavat työskentelevät samalla alalla haastattelijan kanssa, haastateltavien kanssa keskusteleminen oli helppoa. Ammattitermistö oli yhteistä ja haastateltavien arkeen oli helppo samaistua. Toisaalta haasteena on ollut pitää omat kokemuksensa erillään haastateltavien kokemuksista. On ollut tärkeää muistaa antaa tilaa haastateltavien kokemuksille, mutta kuitenkin tarvittaessa auttaa haastateltavaa kertomaan kokemuksistaan lisää. Tutkimuksen aikana havaitsin useaan kertaan, että omat kokemukseni, joita olin pitänyt yleisinä totuuksina, olivatkin melko triviaaleja erikoistapauksia.

Viimeinen laadullisen tutkimuksen klassinen laatukriteeri on oikeellisuus. Oikeellisuutta voi parantaa käyttäjämällä triangulaatiomenetelmiä tai antamalla haastateltavien tarkastaa tulokset. Työssä kerättiin dataa tekemällä yksittäisiä haastatteluja. Tietoa kerättiin käytännössä vain yhdellä menetelmällä. Työn oikeellisuutta arvioitiin siten, että kaksi haastateltavista luki tulokset ja kommentoi niitä. Toisaalta ulkoista oikeellisuutta voi arvioida kirjallisuudesta löydettävillä tutkimuksilla. Aiheesta löydetyt tutkimukset oli tehty useimmiten yritystasolta eikä projektitasolta tai vaihtoehtoisesti ne olivat tapaustutkimuksia, joissa oli hyvin pienet otokset. Ongelmana tutkimuksissa, jotka on teh-

ty yritystasolla, on se, että niiden tulokset eivät ole suoraan vertailtavissa tähän työhön. Tällöin myös oikeellisuutta on hankalampi arvioida aikaisemmin julkaistun tiedon perusteella.

# 9 TULOSTEN ARVIOINTI

Tässä luvussa pohditaan haastatteluissa saatuja tuloksia ja niiden suhdetta alan tutkimukseen. Aliluku 9.1 tarkastelee ketterien menetelmien ja UX-työn yhdistämistä käytännössä yleisesti. Aliluvussa 9.2 keskitytään kehittäjän rooliin ja yhteistyöhön UX-suunnittelijoiden kanssa.

## 9.1 UX-työn yhdistäminen ketterään kehitykseen

Haastattelututkimuksessa havaittiin, että UX-suunnittelu voidaan yhdistää ketterään kehitykseen monella eri tavalla. Työn teoriaosuudessa listattiin neljä eri mallia UX-suunnittelun ja ketterän kehityksen yhdistämiseen. Käytännössä todettiin kuitenkin, että yhtäkään näistä ei käytetty. Jokaisessa projektissa oli myös omat, toisista projekteista eroavat käytäntönsä. Satelliittiprojektin piirteitä oli yhdessä projektissa, jossa oltiin koulutettu UX-lähettiläitä. Tiimi ratkoi ongelmia kuitenkin suoraan UX-suunnittelijoiden kanssa, kun oli tarvetta. Samassa projektissa oli myös haettu ideoita LeanUX-mallista. Toisaalta on muistettava, että suurin osa työssä käsitellyistä projekteista oli pieniä, yhden UX-suunnittelijan projekteja, jolloin satelliittihenkilöiden valitseminen yhteydenpitoon on turhaa. UScrum-tyylistä käyttäjäkokemukseen keskittyvää ylimääräistä tuoteomistajaa ei ollut yhdessäkään projektissa.

Myöskään UX-sprinttejä ei ollut käytössä yhdessäkään projektissa. Kahdessa työhön haastatellussa projektissa, joissa oli kokeiltu UX-sprinttejä, niistä oli luovuttu, koska ne oli koettu hankalaksi. Tämä on sinänsä mielenkiintoinen havainto, joka vaatisi lisätutkimuksia. Käytännössä, vaikka varsinaista erillistä sprinttisysteemiä ei ollut, UX-suunnittelu pyrittiin kuitenkin tekemään, ennen kuin ominaisuus päätyi toteutettavaksi. Tämä saattoi olla esimerkiksi 2-3 sprinttiä etukäteen. Suunnittelutoimenpiteitä ei myöskään oltu missään projektissa toteutettu Lean UX -tyylisenä yhteistyönä. Lean UX on muutenkin raskaan oloinen kehittäjille ja uskon, että se ei tule Gothelfin (2013) esittämässä muodossa yleistymään. On vaikea kuvitella perus-backend-koodareita haastattelemassa käyttäjiä parityönä. On toisaalta muistettava, että ketterien menetelmien malleja ei myöskään toteuteta tyylipuhtaasti. Kaikki haastatellut projektit kuitenkin nimesivät jonkun ketterän mallin, jota käyttivät. UX-suunnittelun suhteen malleja nimettiin vain harvoin. Lisäksi ero todellisuuden ja teoreettisten UX-suunnitteluun tarkoitettujen mallien välillä oli suurempi.

Millaisia käytäntöjä projekteilla sitten oli? Yhdessä projektissa UX-suunnittelu oli otettu mukaan Kanban-tauluun omaksi sarakkeekseen. Yksi projekti oli liittänyt UX-

suunnittelun SAFen virstanpylväisiin. Yksi projekti käytti rinnakkaista UX-sprinttiä. Kolmessa projektissa UX-suunnittelua tehtiin silloin, kun kehittäjät tai projektipäällikkö katsoi, että sitä tarvitaan. Yhdessä projektissa ei ollut UX-suunnittelijaa, vaan kehittäjät hoitivat ratkaisujen tekemisen yhteistyössä asiakkaan kanssa.

Tulokset vastaavat huonosti Øvadin (2014) tekemiä havaintoja. Hän havaitsi tutkimuksessaan, että kolme yritystä kahdeksasta pyrki järjestämään UX-työn rinnakkaisiin sprintteihin. Kolme yritystä käytti puolestaan satelliittilähestymistapaa, jossa käytetään satelliittihenkilöä, joka toimii välimiehenä UX-suunnittelutiimin ja kehitystiimin välillä. Lisäksi kahdella yrityksellä oli käytössä joku muunlainen ratkaisu. Voidaan kysyä, kuinka hyvin Øvadin tutkimuksen tulokset ovat yleistettävissä tähän tutkimukseen. Tässä työssä tutkittiin projektitiimejä eikä yrityksiä niin kuin Øvadin tutkimuksessa. Lisäksi Øvadin ja Larsenin (2015) tutkimuksessa havaittiin, että vuonna 2015 hieman yli puolessa yrityksistä UX-työtä ei tehty ketterästi osana ketteriä tiimejä, vaikka kaikki seitsemän tutkimukseen osallistunutta yritystä käyttivät ketteriä menetelmiä tai ketteriä menetelmiä yhdistettynä vesiputousmalliin.

Øvad (2014) toteaa myös, että ”kuusi kahdeksasta yrityksestä kertoi prosessinsa olevan *ad hoc* -tyyppisiä, Hän pitää prosessien puuttumista suurimpana ongelmana UX-työssä, joka tehdään ketterässä ympäristössä. Øvad havaitsi myös, että muutamassa yrityksessä kukaan ei ollut vastuussa tehdystä UX-työstä ja osa siitä tuli tehtyä vain silloin, kun joku muisti. Øvad ei määrittele tarkkaan, mitä hän tarkoittaa *ad hoc* -tyyppisellä. A*d hoc* -tyyppisen prosessin voi tulkita tarkoittavan toimintatapaa, jossa ei ole *suunniteltu* erityisiä prosesseja UX-suunnittelun ja muun kehityksen yhdistämiseen. Käytännössä tiettyjä toimintotapoja epäilemättä syntyy. Tässä työssä tehtiin samansuuntaisia havaintoja. Suunnilleen puolessa projekteista prosessit vaikuttivat syntyneen suuremmin suunnittelematta. Lisäksi muissa projekteissa kerrottiin tehtävän välillä *ad hoc* -työtä etukäteissuunnittelun epäonnistua tai yllättävien tilanteiden sattuessa.

Työssä havaittiin, että myös yhden tiimin projekteissa oltiin hyödynnetty SAFen ominaisuuksia. Voi olla, että UX-suunnittelu ja palvelumuotoilu hyötyvät SAFe:n tai sen hankeinkrementtien käyttöönotosta. SAFe tarjoaa sprinttejä pidemmän säännöllisen suunnittelusyklin, mikä helpottaa suurempien kokonaisuuksien hallintaa ja helpottaa näin kokonaiskuvan säilymistä. Käyttäjäkokemuksen kokonaiskuvan hallinta on haaste, joka on havaittu sekä alan julkaisuissa (mm. Salah ym. 2014) että työhön tehdyissä haastatteluissa. Tutkimustietoa SAFesta on kuitenkin vielä hyvin niukasti saatavissa. Tukea tutkimustiedosta ei ole myöskään saatavilla, jos halutaan tehdä palvelumuotoilua samaan aikaan ketterästi ohjelmistokehityksen kanssa. Tutkimusta siitä, kuinka palvelumuotoilu yhdistetään ketterään kehitykseen ei käytännössä ole. Haettaessa tietokannoista aiheesta ei löytynyt edes tapaustutkimuksia, saati sitten valmiita malleja. Koska palvelumuotoilussa käsiteltävät kokonaisuudet ovat suurempia kuin mitä sprintin aikana tehdään, yhdistämistä täytyisi miettiä suurempina kokonaisuuksina. Toisaalta täytyy muistaa myös, että termit UX-suunnittelu ja palvelumuotoilu ovat osin päällekkäisiä.

Lisäksi osa sellaisista asioista, joita aikaisemmin on luokiteltu kuuluvan UX-suunnittelun alle, saatetaan pitää nykyisin osana palvelumuotoilua.

Ne UX-suunnittelijat, joiden kanssa olen keskustellut haastattelujen ulkopuolella, ovat kertoneet, että periaatteessa UX-työ yhdistetään ketteriin menetelmiin käyttämällä rinnakkaisia UX-sprinttejä. Käytännössä tämä ei kuitenkaan totetunut työhön haastatelluissa projekteissa. Uskon osin tästä syystä, että jos työssä olisi tarkasteltu asiaa yritystasolla, oltaisiin saatu vastauksia, jotka olisivat olleet lähempänä kirjallisuudessa esitettyjä malleja. Toisaalta tuloksiin vaikuttaa myös se, että puolet tämän tutkimuksen projekteista oli yli viisivuotiaita. Projektit kantavat tiettyä historian taakkaa ja ainakin yrityksessä 1 vanhaan projektiin otetaan harvoin uutta UX-suunnittelijaa mukaan. Väitänkin, että UX-työn integroinnin etenemistä yrityksessä voi seurata jossain määrin yrityksen eri-ikäisten projektien tilanteella. Toisaalta, jos tutkimukseen otetaan mukaan myös vanhempia projekteja, on hyväksyttävä se, että projektien käytännöt eivät ole samanlaisia, eivätkä tule muodostumaan samanlaisiksi kuin saman yrityksen vasta aloitetuissa projekteissa. Vanhempien projektien mukaan ottaminen oli silti perusteltua, koska työssä oli tarkoitus luoda mahdollisimman hyvä kokonaiskuva aiheesta ja vanhemmat projektit ovat osa alan todellisuutta ihan yhtä lailla kuin uudemmatkin.

Vaikka työ keskittyikin tarkastelemaan asiaa projektitasolla, niin vaikutti siltä, että eri yrityksissä käyttäjäkokemuksen integroiminen ohjelmistokehitystyöhön oli eri kypsyysvaiheissa. Øvad (2015) toteaa, että tietyn pisteen jälkeen käytettävyyden parantaminen vaatii johdon tukea ja resurssien varaamista UX-työlle. Øvad jatkaa, että, jos UX-suunnittelua ei nähdä tuotekehityksen ydinelementtinä, se on väistämättä vaarassa tulla karsituksi pois. Se, että miten paljon yrityksessä oli kehittäjiä suhteessa UX-suunnittelijoihin välillä, vaihteli paljon työssä haastateltujen yritysten välillä.

## 9.2 Kehittäjien rooli ja yhteistyö UX-suunnittelijoiden kanssa

Kehittäjän roolista käyttäjäkokemuksen suunnittelussa on tehty vain vähän tutkimuksia. Aikaisempia tutkimuksia kehittäjän roolista etsittiin haulla (”developer” tai ”software engineer”) ja (”UX” tai ”user experience” tai ”user-experience”). Löytyneistä artikkeleista valikoitiin ne, jotka sopivat aihepiiriltään. Tämän jälkeen yritettiin etsiä lumipallomenetelmällä lisää lähteitä niistä artikkeleista, joihin viitattiin lähteissä tai joihin lähteistä viitattiin. Tiedetään, että julkaisuissa, jotka käsittelevät ketterien menetelmien ja käyttäjäkokemuksen suunnittelun yhdistämistä, on välillä huomioita myös kehittäjien roolista. Tälläisiä julkaisuja ei kuitenkaan lähdetty tutkimaan järjestelmällisesti julkaisujen suuren määrän vuoksi. Hakutulokset ovat niukkoja. Kuusinen (2015b) on tutkinut UX-suunnittelijoiden ja kehittäjien työnjakoa, Øvad (2015) puolestaan on tutkinut kehittäjien kouluttamista tekemään yksinkertaisia UX-tehtäviä. Samaa asiaa on pohdittu jo joissakin aikaisemmissa tutkimuksissa. Lisäksi haettiin hakutermeillä ”agile ja ux ja

cooperation”. Tällä haulla löydettiin Ferreiran ym. (2011) yhteistyöhön keskittyvä tapaustutkimus.

Kirjallisuudesta ei löydetty selvitystä kehittäjien käyttäjäkokemukseen vaikuttavista ratkaisuista. UX-kehittäjien tehtävistä on tehty kyllä tutkimusta (esim. da Silva ym., 2013), mutta lähin vastaava kehittäjistä löydetty tieto oli Kuusisen (2015b) erittely kehittäjien ja tuoteomistajien tekemistä UX-tehtävistä. Kuusisen taulukosta selviää, että kehittäjät ja tuoteomistajat tekivät itsenäisesti ilman UX-asiantuntijoita muutoksia erityisesti suunnitelmaan (*design*), katselmoivat toteutusta, suunnittelivat ominaisuuksia ja päättivät, kuinka käyttöliittymän yksityiskohdat toteutetaan. Kuusisen tutkimuksessa nousee esille myös käyttäjätutkimusten ja käyttäjätestien suunnittelu, mutta tämä koskee luultavasti enemmän tuoteomistajia kuin kehittäjiä. Lisäksi Kuusinen toteaa, että kehittäjät selvensivät käyttäjävaatimuksia, suunnittelivat ominaisuuksia ja katselmoivat toimintoja riippumatta siitä, osallistuiko UX-suunnittelija työhön vai ei. Hän tulkitsi sen johtuvan siitä, että nämä ovat ydintoiminnallisuuksia, jotka on pakko tehdä, jotta saadaan aikaan toimiva ohjelmisto. Kuusinen väittää, että UX-suunnittelijoiden on tärkeä tehdä yhteistyötä kehittäjien kanssa näissä ydintoiminnallisuuksissa tai muuten kehittäjät tekevät työn itsenäisesti ja saattavat muodostaa asiasta merkittävällä tavalla eroavan käsityksen kuin UX-suunnittelijat.

Kuusinen havaitsi myös, että UX-asiantuntijat tekivät yhteistyötä kehittäjien kanssa erityisesti demosessioissa, suunnittelivat ominaisuuksia ja jakoivat ymmärrystä käyttöliittymästä. Tässä työssä tehtyjen haastattelujen perusteella kehittäjät tekevät itsenäisesti erityisesti pieniä käyttöliittymäratkaisuja. Ominaisuuksia kehittäjät suunnittelivat myös, yleisesti yhteistyössä muun kehitystiimin ja asiakkaan kanssa. Työhön liittyvissä haastatteluissa havaittiin myös, että kehittäjät osallistuvat useassa projektissa kyllä katselmointiin ja katselmoivat koodia, mutta ominaisuuksien katselmointi koettiin asiakkaan tehtäväksi. Ferreira havaitsi tapaustutkimuksessaan, että kehittäjät olivat vastuussa UX-suunnitelmien läpikäymisestä, jotta ne voitiin toteuttaa toimiviksi ohjelmistoiksi. Hänen kokemuksensa mukaan kehittäjät kyseenalaistivat, kokeilivat ja testasivat jo olemassa olevia ratkaisuja. Tämä auttoi kehittäjiä löytämään suunnitelman eroavaisuudet vanhan toteutuksen kanssa ja huomaamaan ne kohdat suunnitelmasta, jotka olivat toteutuskelvottomia. Tehdessään tätä kehittäjät törmäsivät haasteisiin, joihin he tarvitsivat UX-suunnittelijan apua. Kaikki tässä työssä haastatellut kehittäjät kuvasivat tätä prosessia jollain tavalla. Tosin työhön liittyvissä haastatteluissa havaittiin, että ongelmatilanteet saatettiin ratkaista myös ilman UX-suunnittelijan apua.

Kuusinen (2015b) havaitsi, että yhteistyömalleissa oli merkittäviä eroja eri projektien välillä. Hän identifioi kolme erilaista mallia yhdistää UX-suunnittelu projektiin. Kuusisen havaintojen mukaan yhdessä projektissa saattoi olla piirteitä useammasta mallista. Ensinnäkin oli **mahdollisimman vähäisen yhteistyön malli,** jossa yksittäinen UX-suunnittelija tai erillinen UX-tiimi tekivät UX-työn ilman muun tiimin apua. Kuusisen mukaan tätä mallia tulisi välttää, koska siitä seuraa enemmän haittaa kuin hyötyä.

Sitten oli **UX-suunnittelijoiden ja tuoteomistajan läheisen yhteistyön malli**, jossa UX-suunnittelija ja tuoteomistaja tekevät UX-työn yhteistyössä ja tulokset esitetään kehittäjille. Kolmantena oli **UX-suunnittelijoiden ja kehittäjien läheisen yhteistyön malli**, tuoteomistajan rooli on tässä mallissa pienempi. Kaikkia näitä piirteitä löydettiin tutkituista projekteista. Lisäksi löydettiin kuitenkin vielä yksi tapa toimia. Oli myös projekteja, joissa kehittäjät olivat viime kädessä vastuussa UX-ratkaisujen tekemisestä. He joko tekivät ne yhteistyössä asiakkaan kanssa tai päättivät, koska UX-suunnittelija tarvitaan suunnittelemaan ratkaisut. Se, että tässä työssä löydettiin ylimääräinen malli, johtuu luultavasti siitä, että työhön valittiin kehittäjän näkökulma. Jos tutkimus oltaisiin tehty UX-suunnittelijalähtöisesti, mukaan ei olisi luultavasti valikoitunut projekteja, joissa UX-suunnittelijan rooli on niin vähäinen.

Kaikissa projekteissa ei edelleenkään ole UX-suunnittelijaa käytössä tai UX-suunnittelija saattaa olla mukana vain paperilla. Väänänen-Vainio-Mattila ym. (2008) esittävät, että tutkimusyhteisön keskittyessä erityisesti käyttäjäkokemuksen hedonistisiin piirteisiin, ohjelmistoyritykset keskittyvät erityisesti käytännölliseen puoleen. Herää kysymys, johtuuko tämä UX-suunnittelijoiden rajoitetusta mukana olosta projekteissa. Kuusisen mukaan kehittäjät näyttävät pystyvän ymmärtämään käyttäjäkokemuksen käytännöllistä puolta, mutta hedonistiseen puoleen tarvitaan UX-asiantuntijaa. Toisaalta ohjelmistokehityksessä on historiallista painolastia keskittyä toiminnallisiin vaatimuksiin. Jos tehdään tarkat vaatimusmäärittelyt, toiminnalliset vaatimukset on helpompi ymmärtää ja mitata kuin laadulliset. Jos ajatellaan esimerkiksi toimintopistelaskentaan perustuvia menetelmiä, laadullisten vaatimusten tarpeellisuus kyllä tunnistetaan, mutta määrittely painottuu nimenomaan toiminnallisiin vaatimuksiin, eikä käyttäjän kokemukseen liittyviin vaatimuksiin anneta työkaluja.

Toisaalta Da Silva (2012) toteaa, että ohjelmistokehittäjillä on taipumus kuunnella sitä, mitä asiakkaat haluavat eikä pohtia sitä, mitä he oikeasti tarvitsevat. Tämä näkyi haastatteluissa esimerkiksi siten, että kehittäjillä oli korkea kynnys kyseenalaistaa asiakkaan toiveita käyttäjäkokemukseen vaikuttavissa ratkaisuissa. On kuitenkin muistettava, että kehittäjillä ei ole useinkaan sellaista näkyvyyttä loppukäyttäjien toimintaan tai kokonaisnäkemystä asiakkaan toimialasta, että kehittäjät pystyisivät kyseenalaistamaan asiakkaan vaatimukset.

Øvadin ym. (2015) tutkimuksessa pohdittiin kehittäjien kouluttamista UX-tehtäviin. Hän havaitsi, että kuudessa seitsemästä yrityksistä oltiin kiinnostuneita mahdollisuudesta, että kehittäjät tekisivät pieniä UX-töitä ja kahdessa näistä yrityksessä tehtiin jo niin. Ovad toteaa, että kehittäjien tekemät pienimuotoiset käytettävyys ja UX-tehtävät voivat parantaa UX-suunnittelijoiden ja kehittäjien välistä ymmärtämystä. Ovadin kokemusten mukaan kehittäjät pystyivät kommunikoimaan paremmin UX-suunnittelijoiden kanssa ja antamaan heille hyödyllistä palautetta. Kun kehittäjiltä kysyttiin tämän työn haastatteluissa, millä perusteella he tekevät käyttöliittymäratkaisut, yksikään ei nostanut esille työpaikalla saamaansa koulutusta. Sen sijaan kolme kehittäjää mainitsi opinnois-

saan saamansa tietämyksen. Yhden projektin käytännöissä oli järjestelmällisesti pyritty lisäämään joidenkin kehittäjien UX-taitoja. Toisaalta yksi haastatelluista kertoi ollensa niin paljon tekemisissä UX-suunnittelijoiden kanssa, että oli alkanut itsekin ymmärtää paremmin käyttäjien tarpeita. Voi siis ajatella, että jo jatkuva vuorovaikutus projektitiimissä kehittäjien ja UX-suunnittelijoiden välillä, auttaa kehittäjiä ymmärtämään käyttäjien tarpeita ja toisaalta UX-suunnittelijoita ymmärtämään teknisiä rajoitteita.

Tässä työssä on puhuttu yllättävän paljon siitä, miten ihmiset viestivät toisilleen. Conwayn laiksi kutsutaan Melvin Conwayn ajatusta: "Organisaatiot, jotka suunnittelevat järjestelmiä... rajoittuvat tuottamaan tuotteita, jotka ovat kopioita organisaation viestintärakenteista." (Wikipedia, 2018a) Olisi mielenkiintoista nähdä, heijastuuko lopputulokseen se malli, millä ketterä kehitys on yhdistetty käyttäjäkeskeisiin menetelmiin. Tavoitteena ketterässä kehityksessä on pyrkiä yhtenäisiin monialaisiin tiimeihin, joissa eri alojen ammattilaiset tekevät saumatonta yhteistyötä. Käytännössä tämä ei kuitenkaan aina toteudu ja UX-suunnittelija jää erilleen toteutustiimistä. Kuusisen (2015a) mukaan erilliset kehitys- ja UX-tiimit tekivät vähemmän yhteistyötä. Lisäksi, kun kehittäjät saivat UX-suunnittelijan tekemät suunnitelmat valmiina, kehittäjät tunsivat tulevansa määräillyksi ja suunnitelmaa ei pidetty niin toteutuskelpoisena. Kuusinen huomasi tutkimuksissaan myös, että jos kommunikaatio oli vähäistä, UX-asiantuntija ja kehittäjät tekivät omia erillisiä suunnitelmiaan. Kehittäjät myös usein tekivät muutoksia käyttöliittymään ja päättivät kuinka toteuttaa käyttöliittymäratkaisuja konsultoimatta UX-suunnittelijaa. Tämä havaittiin myös tämän työn haastatteluissa.

Ja jos käyttäjäkeskeisen suunnittelun yhdistäminen ketteriin menetelmiin pyrkii lisäämään viestintää kehittäjien ja UX-suunnittelijoiden välillä ja poistamaan siiloja organisaatiossa, niin sitä yritetään tehdä myös muualla ohjelmistokehityksessä. Devops-malli pyrkii purkamaan raja-aitoja toisessa suunnassa. Devops tulee sanoista development ja operations ja se pyrkii parantamaan kehittäjien ja palveluntuottajien yhteistyötä. Palveluntuottajilla tarkoitetaan tässä yhteydessä niitä henkilöitä, jotka huolehtivat ympäristöistä, asennuksesta ja ylläpidosta. Devopsin piirissä puhutaan hämmennyksen muurista, jonka yli kehittäjä paiskaa valmiiksi koodatun ohjelmiston ja jättää lopun palvelutuotannon huoleksi ja tätä muuria pyritään purkamaan tätä muuria tiivistämällä palveluntuottajien yhteistyötä kehittäjien kanssa. Samanlaisia muureja syntyy helposti myös muiden asiantuntijaryhmien, kuten kehittäjien ja UX-suunnittelijoiden välille.

Toisaalta devops laajentaa myös kehittäjien roolia. Devopsissa kehittäjien tehtäviin kuuluu sovelluksen tekemisen lisäksi myös operatiivisen tietämyksen hankkiminen niin, että kehittäjä voi tehdä osan niistä asioista, jotka ovat ennen kuuluneet palveluntuottajille. Toimintamalli on osa suurempaa muutosta, johon kuuluvat pilvipalvelujen yleistyminen ja ketteryyden mukanaan tuoma julkaisusyklin nopeutuminen. Kehittäjiin kohdistuu siis painetta oppia operatiivisia tehtäviä. Vaikka kehittäjien tekemistä UX-ratkaisuista ei käydä samanlaista keskustelua, kehittäjillä on toisaalta tarve ymmärtää myös tekemiensä ratkaisujen vaikutuksia käyttäjiin. Lisäksi Lean UX vaatii paljonkin

UX-työhön osallistumista kehittäjiltä. Kehittäjät roolina ovat siis kahden tulen välissä, kun työnkuvaa yritetään laajentaa niin käyttäjien ymmärtämisen kuin operatiivisen toiminnan suuntaan. Lisäksi myös tietoturvallisen suunnittelun (*secure design*) puolelta toivotaan, että kehittäjät ottaisivat laajempaa vastuuta tietoturvasta.

On kuitenkin muistettava, että kehittäjät eivät ole yhtenäinen ryhmä. Luultavasti kehittäjät, jotka ovat jo valmiiksi suuntautuneet käyttäjälähtöisesti, ottavat helpommin vastuullensa pieniä UX-tehtäviä. Toiset kehittäjät ovat puolestaan valmiiksi operatiivisesti suuntautuneempia, jolloin operatiiviset tehtävät tuntuvat luontevilta. Vaikka puhutaan *fullstack*-kehittäjistä, ehkä olisi kuitenkin jatkossa tarve jakaa kehittäjien ammattikuntaa eri tavoin suuntautuneiksi rooleiksi. Toisaalta kuitenkin ketterän ajattelutavan mukaisesti osaamissiilojen syntymistä pitäisi pyrkiä välttämään. Jotkut kehittäjät varmasti pystyvätkin hoitamaan koko paletin. Tyypillisesti kuitenkin osaamisen laajentuessa sen syvyys vähenee. Ammatit syntyivät sen takia, että erikoistumalla saadaan lisättyä tehokkuutta. Lisäksi näen vaatimukset *koko* tiimin osallistumisesta johonkin tehtävään melko idealistisena ajattelutapana, mikä ei toimi vähänkään suuremmissa projekteissa. Kultaisen keskitien löytäminen lienee siis tässäkin paikallaan.

Ylipäätään kirjallisuudessa esiintyvät roolikuvaukset ovat yksinkertaistuksia, jotka ei vastaa todellisuutta. Todellisuudessa havaittiin, että haastateltujen henkilöiden työnkuva meni usein useamman roolin alueelle. Jos projektiin ei ole resursoitu graafista suunnittelijaa, jonkun muun on tehtävä graafinen suunnittelu. Jos muita ei ole tarjolla, tämä henkilö saattaa olla vastahakoinen kehittäjä. Havaittiin myös, että eri kehittäjillä on omat mieltymyksensä ja he hakeutuvat aktiivisesti tekemään niitä tehtäviä, jotka heitä kiinnostavat. Kehittäjä saattoi sanoa aika suoraan, että häntä eivät näyttökuvat kiinnosta. Toinen kehittäjä puolestaan kertoi nauttivansa graafisesta suunnittelusta. McInerney (2017) toteaakin, että yhteistyöllä kehittäjän kanssa voi olla suunnittelutyöhön suuri positiivinen tai negatiivinen vaikutus riippuen siitä, millainen kehittäjä on. Kehittäjän rooli ohjelmistoprojektissa ei siis riipu pelkästään siitä, että mihin tehtävään hänet on resursoitu. Rooliin vaikuttavat myös kehittäjän omat taidot ja mielenkiinnonkohteet, se mitä taitoja ja mielenkiinnonkohteita muilla projektin henkilöillä on ja se, että millaista roolia projektin henkilöt itse hakevat. Vaikka työn organisointiin käytettäisiin mitä mallia tahansa, projektit koostuvat siis lopulta ihmisistä, joilla on erilaisia taitoja ja kiinnostuksen kohteita.

# 10 YHTEENVETO

Tätä työtä tehdessä havaittiin, että oikeassa elämässä projektit eivät ole aina järjestetty oppikirjan mukaan ja käyttäjäkokemukseen vaikuttavia ratkaisuja tehdään projekteissa monella eri tavalla. Työssä havaittiin myös, että kehittäjät tekevät käyttäjäkokemukseen vaikuttavia ratkaisuja. Ratkaisujen laajuus riippui paljon siitä, mikä UX-suunnittelijan rooli projektissa oli. Kirjallisuudesta löytyy paljon hyviä neuvoja siihen, kuinka ketterä kehitys tulisi yhdistää käyttäjäkeskeiseen suunnitteluun. Lisätään joukkoon muutama tätä työtä tehdessä kypsynyt ajatus.

1. Jos halutaan, että kehittäjillä on matala kynnys kysyä ongelmatilanteissa UX-suunnittelijoilta neuvoa, UX-suunnittelija pitää integroida tiimiin jollain tavalla, esimerkiksi niin, että UX-suunnittelija osallistuu tiimitapaamisiin. Keskustelua syntyy helpoiten silloin, kun UX-suunnittelija on saatavilla.
2. Ei ole olemassa valmista mallia, jonka mukaan UX-ratkaisut tulisi tehdä ketterissä menetelmissä. Projekteille varatut resurssit vaihtelevat niin toimittajan kuin asiakkaan puolella. Organisaation totutut tavat toimia ovat erilaisia. Yksittäiset ihmiset ovat erilaisia. Olemassa olevat mallit ovat puutteellisia. Parhaimmat ratkaisut projektissa saadaan todennäköisesti iteroimalla valmiita malleja sopimaan omaan projektiin ja arvioimalla säännöllisin väliajoin, mitä parannettavaa mallissa on. Oleellista on, että käyttäjien olemassaolo muistetaan ja koko tiimi pyrkii tekemään käyttäjäystävällisen tuotteen. Kuitenkin malli riippuu paljon siitä, mitkä resurssit projektissa on. Suuressa projektissa, jossa on useita UX-suunnittelijoita ja mahdollisesti palvelumuotoilija, on täysin eri mahdollisuudet toteuttaa erilaisia UX-suunnittelun ketterään kehitykseen yhdistäviä malleja kuin projekteissa joissa on osa-aikainen UX-suunnittelija.
3. UX-suunnittelijan työtaakan tulisi olla sellainen, että UX-suunnittelija pystyy olemaan mukana projektissa täyspainoisesti ja olemaan se henkilö, joka viime kädessä edustaa käyttäjän näkökulmaa.

Henkilökohtaisesti olen myös tajunnut, miten tärkeää on, että projektissa on mukana joku, jonka tehtävä on katsoa projektia käyttäjän näkökulmasta. Kehittäjällä on taakkanaan (joskin myös etunaan) se, että kehittäjä katsoo asioita aina jonkin verran myös teknisestä näkökulmasta. Kehittäjät pystyvät tekemään käyttäjäystävällisiä ratkaisuja ja kehittäjille voi antaa vastuuta myös käyttäjäkokemuksen suunnittelusta. Mukana tulisi aina olla myös joku, jolle käyttäjä on kaiken keskiössä.

# LÄHTEET

Abrahamsson, P. *ym.* (2002) ”Agile software development methods: Review and analysis”, *VTT Publications*, (478), ss. 3–107. doi: 10.1076/csed.12.3.167.8613.

Agile Alliance (2018a) *Scrum of Scrums*. Saatavissa: https://www.agilealliance.org/glossary/scrum-of-scrums (Viitattu: 15. marraskuuta 2018).

Agile Alliance (2018b) *What is Extreme Programming*. Saatavissa: https://www.agilealliance.org/glossary/xp/ (Viitattu: 24. marraskuuta 2018).

Ahmad, M. O. *ym.* (2018) ”Kanban in software engineering: A systematic mapping study”, *Journal of Systems and Software*. Elsevier, 137, ss. 96–113. doi: 10.1016/J.JSS.2017.11.045.

Ahmad, M. O., Markkula, J. ja Oivo, M. (2013) ”Kanban in software development: A systematic literature review”, teoksessa *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, ss. 9–16. doi: 10.1109/SEAA.2013.28.

Al-Baik, O. ja Miller, J. (2015) ”The kanban approach, between agility and leanness: a systematic review”, *Empirical Software Engineering*. Springer US, 20(6), ss. 1861–1897. doi: 10.1007/s10664-014-9340-x.

Anderson, D. J. (2010) *Kanban: successful evolutionary change for your technology business*. Seattle: Blue Hole Press.

Ardito, C. *ym.* (2017) ”Integrating a SCRUM-Based Process with Human Centred Design: An Experience from an Action Research Study”, teoksessa *2017 IEEE/ACM 5th International Workshop on Conducting Empirical Studies in Industry (CESI)*. IEEE, ss. 2–8. doi: 10.1109/CESI.2017.7.

Beck, K. *ym.* (2001) *Agile Manifesto. Manifesto for Agile Software Development*. Saatavissa: http://agilemanifesto.org/ (Viitattu: 20. toukokuuta 2017).

Brhel, M. *ym.* (2015) ”Exploring principles of user-centered agile software development: A literature review”, *Information and Software Technology*, 61, ss. 163–181. doi: 10.1016/j.infsof.2015.01.004.

Budwig, M., Jeong, S. ja Kelkar, K. (2009) ”When user experience met agile”, teoksessa *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems - CHI EA ’09*. New York, New York, USA: ACM

Press, s. 3075. doi: 10.1145/1520340.1520434.

Charette, R. (2018) ”Challenging the Fundamental Notions of Software Development”.

Costabile, M. F. (2000) ”Usability in the Software Lifecycle”, *Handbook of Software Engineering and Knowledge Engineering*. Saatavissa: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.9076&rep=rep1&type=pdf (Viitattu: 5. elokuuta 2017).

Düchting, M., Zimmermann, D. ja Nebe, K. (2007) ”Incorporating User Centered Requirement Engineering into Agile Software Development”, teoksessa *Human-Computer Interaction. Interaction Design and Usability*. Berlin, Heidelberg: Springer Berlin Heidelberg, ss. 58–67. doi: 978-3-540-73105-4_7.

Ebert, C. ja Paasivaara, M. (2017) ”Scaling Agile”, *IEEE Software*, 34(6), ss. 98–103. doi: 10.1109/MS.2017.4121226.

Endsley, M. R. (2016) *Designing for Situation Awareness: An Approach to User-Centered Design, Second Edition*. 2nd p, *Design*. 2nd p. CRC Press. doi: 10.1201/9780203485088.

Ferreira, J., Sharp, H. ja Robinson, H. (2011) ”User experience design and agile development: managing cooperation through articulation work”, *Software: Practice and Experience*. Wiley-Blackwell, 41(9), ss. 963–974. doi: 10.1002/spe.1012.

Flewelling, P. (2018) *The Agile Developer's Handbook*. Packt Publishing.

Flick, U. (2006) *An introduction to qualitative research.* 3. London: Sage.

Fowler, M. (2008) *AgileVersusLean*, *MartinFowler.com*. Saatavissa: https://martinfowler.com/bliki/AgileVersusLean.html.

Fox, D., Sillito, J. ja Maurer, F. (2008) ”Agile Methods and User-Centered Design: How These Two Methodologies are Being Successfully Integrated in Industry”, teoksessa *Agile 2008 Conference*. IEEE, ss. 63–72. doi: 10.1109/Agile.2008.78.

Gothelf, J. (2013) *Lean Ux*. Toimittanut J. Seiden. Sebastopol, CA: O'Reilly.

Grudin, J. (2011) ”A moving target: The evolution of HCI”, teoksessa *Human-computer interaction handbook, 3rd edition, J. Jacko (Ed.)*. Taylor & Francis, ss. 1–40. Saatavissa: https://www.microsoft.com/en-us/research/publication/a-moving-target-the-evolution-of-hci/.

Haikala, I. ja Märijärvi, J. (2004) *Ohjelmistotuotanto*. 10. uud. p. Helsinki: Talentum.

Haikala, I. ja Mikkonen, T. (2011) *Ohjelmistotuotannon käytännöt*. 12. uud p. Helsinki: Talentum.

Hassenzahl, M. (2001) ”The effect of perceived hedonic quality on product

appealingness", *International Journal of Human-Computer Interaction*, 13(4), ss. 481–499. doi: 10.1207/S15327590IJHC1304_07.

Hassenzahl, M. (2003) "The Thing and I: Understanding the Relationship Between User and Product", teoksessa. Springer, Dordrecht, ss. 31–42. doi: 10.1007/1-4020-2967-5_4.

Hassenzahl, M. (2004) "The Interplay of Beauty, Goodness, and Usability in Interactive Products", *Human-Computer Interaction*, 19(4), ss. 319–349. doi: 10.1207/s15327051hci1904_2.

Hassenzahl, M. (2008) "User experience (UX)", teoksessa *Proceedings of the 20th International Conference of the Association Francophone d'Interaction Homme-Machine on - IHM '08*. New York: ACM Press, s. 11. doi: 10.1145/1512714.1512717.

Hietaniemi, J. (2016) *SAFe tuo ketteryyttä XL-kokoisille*. Saatavissa: https://gofore.com/safe-ketteryytta-xl-kokoisille/ (Viitattu: 15. marraskuuta 2018).

Hiren, D. (2016) *The Three Pillars of Empiricism (Scrum) | Scrum.org*. Saatavissa: https://www.scrum.org/resources/blog/three-pillars-empiricism-scrum (Viitattu: 28. heinäkuuta 2017).

Hyytiälä, H. (2011) *Systems thinking and the Vanguard method*, *Reaktor*. Saatavissa: https://www.reaktor.com/blog/systems-thinking-and-the-vanguard-method/ (Viitattu: 20. elokuuta 2018).

Illmensee, T. ja Muff, A. (2009) "5 Users Every Friday: A Case Study in Applied Research", teoksessa *2009 Agile Conference*. IEEE, ss. 404–409. doi: 10.1109/AGILE.2009.45.

Koppa (2014a) *Aineistonhankintamenetelmät*, *Jyväskylän yliopisto*. Saatavissa: https://koppa.jyu.fi/avoimet/hum/menetelmapolkuja/menetelmapolku/aineistonhankintamenetelmat (Viitattu: 20. marraskuuta 2018).

Koppa (2014b) *Tutkimusstrategiat*, *Jyväskylän yliopisto*. Saatavissa: https://koppa.jyu.fi/avoimet/hum/menetelmapolkuja/menetelmapolku/tutkimusstrategiat.

Koppa (2015) *Määrällinen tutkimus*, *Jyväskylän yliopisto*. Saatavissa: https://koppa.jyu.fi/avoimet/hum/menetelmapolkuja/menetelmapolku/tutkimusstrategiat/maarallinen-tutkimus (Viitattu: 11. marraskuuta 2018).

Kotler, P. ja Keller, K. L. (2012) *Marketing management*. 14th globa. Harlow: Pearson.

Kujala, S. *ym.* (2011) "Identifying hedonic factors in long-term user experience", teoksessa *Proceedings of the 2011 Conference on Designing Pleasurable Products and Interfaces (DPPI '11)*. New York: ACM Press, s. 17:1-17:8. doi: 10.1145/2347504.2347523.

Kuusinen, K. (2015a) *Integrating UX Work in Agile Enterprise Software Development*. Tampere: Tampere University of Technology.

Kuusinen, K. (2015b) ”Task Allocation Between UX Specialists and Developers in Agile Software Development Projects”, ss. 27–44. doi: 10.1007/978-3-319-22698-9.

Kuusinen, K. *ym.* (2016) ”Towards Understanding How Agile Teams Predict User Experience”, teoksessa. Springer, Cham, ss. 163–189. doi: 10.1007/978-3-319-32165-3_7.

Larman, C. (2004) *Agile and iterative development : a manager's guide*. Addison-Wesley.

Larman, C. (2004) ”Iterative, Evolutionary, and Agile”, teoksessa *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition*. Prentice Hall.

Larman, C. ja Basili, V. R. (2003) ”Iterative and Incremental Development: A Brief history”. Saatavissa: https://pdfs.semanticscholar.org/f9b3/ca89c69bacfade039c8be40762c6857bda11.pdf (Viitattu: 16. toukokuuta 2018).

Larusdottir, M., Gulliksen, J. ja Cajander, Å. (2017) ”A license to kill – Improving UCSD in Agile development”, *Journal of Systems and Software*. Elsevier, 123, ss. 214–222. doi: 10.1016/J.JSS.2016.01.024.

Lee, J. C., McCrickard, D. S. ja Stevens, K. T. (2009) ”Examining the Foundations of Agile Usability with eXtreme Scenario-Based Design”, teoksessa *2009 Agile Conference*. IEEE, ss. 3–10. doi: 10.1109/AGILE.2009.30.

Liikkanen, L. A. *ym.* (2014) ”Lean UX - The Next Generation of User-Centered Agile Development?”, teoksessa *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational*. Helsinki: ACM, ss. 1095–1100. doi: 10.1145/2639189.2670285.

Logistiikan maailma (2018) *JIT (Just-in-time) ja imuohjaus*. Saatavissa: http://www.logistiikanmaailma.fi/logistiikka/tuotanto/jit-just-in-time-ja-imuohjaus/ (Viitattu: 24. marraskuuta 2018).

McCauley, R. ja Renée (2001) ”Agile development methods poised to upset status quo”, *ACM SIGCSE Bulletin*. ACM, 33(4), s. 14. doi: 10.1145/572139.572150.

McInerney, P. (2017) ”UX in Agile Projects: Taking Stock After 12 Years”, *Interactions*. New York, NY, USA: ACM, 24(2), ss. 58–61. doi: 10.1145/3029605.

Memmel, T., Gundelsweiler, F. ja Reiterer, H. (2007) ”Agile Human-Centered Software Engineering”, teoksessa *Proceedings of the 21st British HCI Group Annual Conference*

*on People and Computers: HCI...but not as we know it - Volume 1*. Swinton: British Computer Society., ss. 167–175.

Miller, L. (2005) ”Case study of customer input for a successful product”, teoksessa *Proceedings - AGILE Confernce 2005*. IEEE Comput. Soc, ss. 225–234. doi: 10.1109/ADC.2005.16.

Modig, N. ja Åhlström, P. (2015) *Tätä on lean: ratkaisu tehokkuusparadoksiin*. Tukholma: Rheologica Publishing.

Narasimhan, R., Swink, M. ja Kim, S. W. (2006) ”Disentangling leanness and agility: An empirical investigation”, *Journal of Operations Management*. Elsevier, 24(5), ss. 440–457. doi: 10.1016/J.JOM.2005.11.011.

Naylor, B. J., Naim, M. M. ja Berry, D. (1999) ”Leagility: Integrating the lean and agile manufacturing paradigms in the total supply chain”, *International Journal of Production Economics*. Elsevier, 62(1–2), ss. 107–118. doi: 10.1016/S0925-5273(98)00223-0.

Nielsen, J. (1993) *Usability engineering*. San Francisco: Academic Press.

Nielsen, J. (1995) *Multimedia and hypertext: the Internet and beyond*. Boston: AP Professional.

Nitor (2018) *SAFe 4.5 sanasto*. Saatavissa: https://www.nitor.com/application/files/8415/2524/8051/Nitor-SAFe-4.5-FIN.pdf (Viitattu: 15. marraskuuta 2018).

Oates, B. J. (2006) *Researching Information Systems and Computing*. Sage Publications Ltd.

Office, C. (2011) *ITIL Service Design 2011 Edition*. Norwich: The Stationery Office.

Øvad, T. (2014) *The Current State of Agile UX in the Danish Industry: the analysis.*, *Aalborg Universitet*. Saatavissa: http://vbn.aau.dk/files/209798499/The_Current_State_of_Agile_UX_in_the_Danish_Industry.pdf (Viitattu: 20. marraskuuta 2018).

Øvad, T. *ym.* (2015) ”Teaching Software Developers to Perform UX Tasks”, teoksessa *Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction on - OzCHI '15*. New York, New York, USA: ACM Press, ss. 397–406. doi: 10.1145/2838739.2838764.

Ovad, T. ja Larsen, L. B. (2015) ”The Prevalence of UX Design in Agile Development Processes in Industry”, teoksessa *2015 Agile Conference*. IEEE, ss. 40–49. doi: 10.1109/Agile.2015.13.

Partogi, J. (2018) *Scrum And eXtreme Programming (XP)*, *Scrum.org*. Saatavissa:

https://www.scrum.org/resources/blog/scrum-and-extreme-programming-xp (Viitattu: 26. marraskuuta 2018).

Poppendieck.LLC (2018) *The Lean Mindset*. Saatavissa: http://poppendieck.com/ (Viitattu: 17. elokuuta 2018).

Poppendieck, M. (Mary B. . ja Poppendieck, T. D. (2003) *Lean software development : an agile toolkit*. Addison-Wesley. Saatavissa: https://dl.acm.org/citation.cfm?id=829556 (Viitattu: 17. elokuuta 2018).

Poppendieck, M. ja Cusumano, M. A. (2012) ”Lean Software Development: A Tutorial”, *IEEE Software*, 29(5), ss. 26–32. doi: 10.1109/MS.2012.107.

Power, K. ja Conboy, K. (2015) ”A metric-based approach to managing architecture-related impediments in product development flow: an industry case study from Cisco”, *Proceedings of the Second International Workshop on Software Architecture and Metrics*. IEEE Press, ss. 15–21. Saatavissa: https://dl.acm.org/citation.cfm?id=2821331 (Viitattu: 12. heinäkuuta 2018).

Preece, J., Rogers, Y. ja Sharp, H. (2002) *Interaction design: beyond human-computer interaction*. New York: Wiley.

Pressman, R. S. (2010) *Software engineering : a practitioner's approach*. New York: McGraw-Hill Education.

Purvis, L., Gosling, J. ja Naim, M. M. (2014) ”The development of a lean, agile and leagile supply network taxonomy based on differing types of flexibility”, *International Journal of Production Economics*. Elsevier, 151, ss. 100–111. doi: 10.1016/J.IJPE.2014.02.002.

Royce, D. W. W. (1970) ”Managing the Development of large Software Systems”, *Ieee Wescon*, (August), ss. 1–9.

Saaranen-Puusniekka, A. ja Kauppinen, A. (2006) *KvaliMOTV - Menetelmäopetuksen tietovaranto*, *Tampere: Yhteiskuntatieteellinen tietoarkisto [ylläpitäjä ja tuottaja]*. Saatavissa: http://www.fsd.uta.fi/menetelmaopetus.

Saffer, D. (2009) *Designing for Interaction: Creating Innovative Applications and Devices*. 2. p. Berkeley: New Riders.

Salah, D., Paige, R. F. ja Cairns, P. (2014) ”A systematic literature review for agile development processes and user centred design integration”, teoksessa *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14*. New York, New York, USA: ACM Press, ss. 1–10. doi: 10.1145/2601248.2601276.

Sauro, J. (2015) *5 Types of Qualitative Methods*, *MeasuringU*. Saatavissa:

https://measuringu.com/qual-methods/.

Scaled Agile (2017) *What's New in SAFe 4.5*. Saatavissa: https://www.scaledagile.com/whats-new-in-safe-4-5/ (Viitattu: 25. marraskuuta 2018).

Scaled Agile (2018) *SAFe provided by Scaled Agile*. Saatavissa: https://www.scaledagileframework.com/# (Viitattu: 16. marraskuuta 2018).

Scrum.org (2017) *What is a Scrum Master?* Saatavissa: https://www.scrum.org/resources/what-is-a-scrum-master (Viitattu: 28. heinäkuuta 2017).

Scrum Guides.org (2016) *Scrum Guide | Scrum Guides*. Saatavissa: http://www.scrumguides.org/scrum-guide.html (Viitattu: 28. heinäkuuta 2017).

*SFS-EN ISO 9241-11, Näyttöpäätteillä tehtävän työn käyttäjäkeskeinen suunnitteluprosessi* (1998). Helsinki: Suomen Standardoimisliitto.

*SFS-EN ISO 9241-210, Ihmisen ja järjestelmän vuorovaikutuksen ergonomia: Vuorovaikutteisten järjestelmien käyttäjäkeskeinen suunnittelu* (2010). Helsinki: Suomen standardoimisliitto.

da Silva, T. S. *ym.* (2013) ”Understanding the UX Designer's Role within Agile Teams”, teoksessa Marcus, A. (toim.) *Design, User Experience, and Usability. Design Philosophy, Methods, and Tools*. Berlin, Heidelberg: Springer Berlin Heidelberg, ss. 599–609. doi: 10.1007/978-3-642-39229-0_64.

Da Silva, T. S. *ym.* (2011) ”User-centered design and agile methods: A systematic review”, *Proceedings - 2011 Agile Conference, Agile 2011*, ss. 77–86. doi: 10.1109/AGILE.2011.24.

Da Silva, T. S. (2012) ”A FRAMEWORK FOR INTEGRATING INTERACTION DESIGN AND AGILE DEVELOPMENT”. Saatavissa: http://ase.cpsc.ucalgary.ca/uploads/Publications/Tiago_Dissertation.pdf (Viitattu: 6. elokuuta 2017).

Sohaib, O. ja Khan, K. (2010) ”Integrating Usability Engineering and Agile Software Development”:, 2(Iccda), ss. 32–38.

Sonderegger, A. ja Sauer, J. (2010) ”The influence of design aesthetics in usability testing: Effects on user performance and perceived usability”, *Applied Ergonomics*. Elsevier, 41(3), ss. 403–410. doi: 10.1016/J.APERGO.2009.09.002.

Subhajit, D. (2007) ”Iterative and Incremental Development: A Brief Background”, teoksessa *Metrics-Driven Enterprise Software Development : Effectively Meeting Evolving Business Needs*. Lauderdale: J. Ross Publishing, Inc.

Sy, D. (2007) ”Adapting Usability Investigations for Agile User-Centered Design”,

*Journal of Usability Studies*, 2(3), ss. 112–132.

Tuulaniemi, J. (2011) *Palvelumuotoilu*. Helsinki: Talentum.

Väänänen-Vainio-Mattila, K., Roto, V. ja Hassenzahl, M. (2008) ”Towards Practical User Experience Evaluation Methods”, *Proceedings of the International Workshop on Meaningful Measure: Valid Useful User Experience Measurement (VUUM 2008)*, (April), ss. 19–22. doi: citeulike-article-id:8362765.

Valli, R. (2018) *Ikkunoita tutkimusmetodeihin : 1, Metodin valinta ja aineistonkeruu : virikkeitä aloittelevalle tutkijalle*. 5. uud. p. Jyväskylä: PS-kustannus.

Wang, X., Conboy, K. ja Cawley, O. (2012) ”’Leagile’ software development: An experience report analysis of the application of lean approaches in agile software development”, *The Journal of Systems & Software*, 85, ss. 1287–1299. doi: 10.1016/j.jss.2012.01.061.

Ward, A. C. (2007) *Lean Product and Process Development*. Cambridge, MA: Lean Enterprise Institute.

Waterman, M., Noble, J. ja Allan, G. (2015) ”How much up-front? A grounded theory of agile architecture”, *Proceedings - International Conference on Software Engineering*, 1, ss. 347–357. doi: 10.1109/ICSE.2015.54.

Wikipedia (2018a) *Conway’s law*. Saatavissa: https://en.wikipedia.org/wiki/Conway%27s_law (Viitattu: 1. marraskuuta 2018).

Wikipedia (2018b) *Theory of Constraints*. Saatavissa: https://en.wikipedia.org/wiki/Theory_of_constraints (Viitattu: 20. elokuuta 2018).

Wikipedia (2018c) *User interface design*. Saatavissa: https://en.wikipedia.org/wiki/User_interface_design (Viitattu: 2. elokuuta 2018).

Wikipedia (2018d) *Waterfall model*. Saatavissa: https://en.wikipedia.org/wiki/Waterfall_model (Viitattu: 24. marraskuuta 2018).

Williams, K. *ym.* (2010) ”Design of Emerging Digital Services: A Taxonomy Credentialization in the Modern World View project ENACT -ERP development networks in action View project”. doi: 10.1007/978-1-4419-5653-8_16.

Zimmermann, D. ja Grötzbach, L. (2007) ”A Requirement Engineering Approach to User Centered Design”, teoksessa *Human-Computer Interaction. Interaction Design and Usability*. Berlin, Heidelberg: Springer Berlin Heidelberg, ss. 360–369. doi: 10.1007/978-3-540-73105-4_40.

# LIITTE A: HAASTATTELURUNKO

### Haastattelun tarkoitus

1. Saada vastauksia tutkimuskysymyksiin.
2. Herätellä ihmisiä ajattelemaan, mikä vaikuttaa käyttäjäkokemukseen ja tehdä tiedostamattomia päätöksiä näkyviksi.

### Haastattelun sisältö

Teemat:

1. Keitä projektiin osallistuu? (Kaikki projektin henkilöt eivät ole välttämättä paikalla.)
    - myös projektipäälliköt?, asiakas?, service center? Devops-ihmiset?

2. Millainen julkaisusykli on?
    - Käytetäänkö jotain kehystä, kuten scrum? Miten kehys käytännössä näkyy työssä?
    - Kuinka usein uudet versiot julkaistaan?
    - Kuinka suuri osa projektissa tehtävästä kehityksestä on 1) suurempia uusia kokonaisuuksia, 2) pienkehitystä, 3) bugikorjauksia.
    - Millaisia tapaamisia julkaisusykliin kuuluu? Keitä näihin tapaamisiin osallistuu?
    - Miten UX-suunnittelu yhdistetään sprintteihin?
    - Miten UX-suunnitellaan?

3. Esimerkkitehtävät:
    a) sovellukseen tulee uusi toiminto, käyttäjä valitsee toiminnon ja sovellus kysyy käyttäjältä varmistuksen, että tehdäänkö toiminto
    b) Sovellukseen lisätään isompi toiminnallisuus, joka vaatii uuden näkymän

    - Ketkä osallistuvat toiminnallisuuden suunnitteluun ja toteutukseen?
    - Mikä vaikuttaa siihen, että millainen *logiikka* toiminnallisuudessa on?
    - Minne uudet käyttöliittymäkomponentit sijoitetaan sovelluksessa?
    - Miltä ne näyttävät? Miksi?

4. Miten projektissanne tehdään käytettävyyteen vaikuttavia päätöksiä?
    - Ketkä kaikki projektissanne tekevät käytettävyyteen ja käyttäjäkokemukseen vaikuttavia päätöksiä?
    - Millaisia ne päätökset ovat?
    - Mitkä vaikuttavat päätöksiin? (ohjeistus, kysy kaverilta, yhdenmukaisuus)

- Ovatko päätökset mielestänne tietoisia vai tiedostamattomia?
- Onko projektissa käytössä jotain UX:n liittyvää ohjeistusta?

5. Miten kehittäjät kokevat yhteistyön UX-henkilöiden kanssa?
    - Pelaako teidän kommunikatio?
    - Miten viestitte?
    - Osallistuvatko UX-suunnittelijat tapaamisiin?
    - onko apua helppo saada?
    - Minkälaisissa asioissa olet kysynyt apua?
    - Miten UX-suunnittelijan mukanaolo/mukanaolemattomuus vaikuttaa käyttäjäkokemukseen

# LIITE B: HAASTATTELULUPA

____ Suostun haastateltavaksi tutkimukseen, joka koskee käytettävyyteen vaikuttavien ratkaisujen tekemistä ohjelmistoprojekteissa. Haastattelu nauhoitetaan ja haastateltavalla on oikeus keskeyttää haastattelu missä tahansa vaiheessa haastattelua. Nauhoitteet poistetaan, kun tutkinto on suoritettu hyväksytysti tai viimeistään vuoden kuluttua. Tutkimukseen osallistuja ei ole yksilöitävissä tutkimuksesta.

Aineistoja käsitellään luottamuksellisesti ja niitä hyödynnetään tutkimustarkoituksessa diplomityössä.

Allekirjoitus ja nimenselvennys

---

# LIITE C: ESITIETOLOMAKE

sukupuoli _____

ikä
☐ <24
☐ 25-34
☐ 35-44
☐ 45-54
☐ 55-64
☐ > 65

Ammatti _____

Kauanko olet ollut nykyisessä toimenkuvassasi? _____ vuotta

Olen tehnyt nykyisessä projektissani/projekteissani:

☐ Palvelumuotoilua
☐ UX-suunnittelua
☐ Graafista suunnittelua
☐ Front-end koodausta
☐ Back-end koodausta
☐ Ohjelmistoarkkitehtuurin suunnittelua
☐ Ympäristöjen tai pipelinejen pystyttämistä
☐ Scrum-tiimin (tai vastaavan) vetämistä
☐ Myyntityötä
☐ Muuta mitä:

_____
_____
_____

Ohjelmiston käyttäjäkokemus (UX). Selitä mitä käsite sinulle tarkoittaa ja mistä käyttäjäkokemus muodostuu.

_____
_____
_____

Teetkö käytettäjäkokemukseen vaikuttavia päätöksiä työssäsi? Millaisia?

_____
_____
_____
_____
_____