Tarmo Terimaa

KÄYTTÄJÄKOKEMUKSEN SUUNNITTELU KETTERÄSSÄ OHJELMISTOKEHITYKSESSÄ

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2019

TIIVISTELMÄ

Terimaa, Tarmo
Käyttäjäkokemuksen suunnittelu ketterässä ohjelmistokehityksessä
Jyväskylä: Jyväskylän yliopisto, 2019, 29 s.
Tietojärjestelmätiede, kandidaatintutkielma
Ohjaaja(t): Palonen, Teija

Nykyaikaisilta ohjelmistoilta vaaditaan niiden teknisten ominaisuuksien lisäksi saumatonta käyttäjäkokemusta. Käyttäjäkokemuksella tarkoitetaan subjektiivista, dynaamista ja kontekstista riippuvaista kokemusta, joka tuotteen tai palvelun käytöstä aiheutuu sen käyttäjälle. Käyttäjälähtöisellä suunnittelulla viitataan prosessiin, jolla pyritään saavuttamaan haluttu käyttäjäkokemus. Laadukkaan käyttäjäkokemuksen lisäksi ohjelmistokehityksessä täytyy huomioida myös tekniset ja taloudelliset haasteet, joita nopeasti muuttuva toimintaympäristö vahvistaa. Ketterät ohjelmistokehitysmenetelmät kehitettiin vastaamaan näihin haasteisiin. Ketterät menetelmät korostavat ihmisten välistä vuorovaikutusta, asiakasyhteistyötä sekä joustavuutta ja usein ajatellaan, että ketteriä menetelmiä käytettäessä saadaan aikaan myös käyttäjän kannalta laadukkaita ohjelmistoja. Käytännössä on kuitenkin todettu, että näin ei aina ole. Tämän tutkielman tarkoituksena on tarkastella sitä, kuinka hyvin ketterä kehitys ottaa huomioon käyttäjäkokemuksen suunnittelun, mitä haasteita käyttäjälähtöisen suunnittelun yhdistämisessä ketteriin menetelmiin liittyy ja miten näitä haasteita on pyritty ratkaisemaan. Ketteristä menetelmistä tarkastelun kohteena on erityisesti Scrum, mutta ketteriä menetelmiä käsitellään enimmäkseen yleisellä tasolla niiden taustalla vaikuttavien arvojen kautta. Tutkielmassa tarkastellusta kirjallisuudesta löydetyt haasteet voidaan pitkälti liittää ketteriin arvoihin, jotka eivät tue riittävästi käyttäjälähtöisyyttä. Käyttäjälähtöisten menetelmien hitaus ja muodollisuus aiheuttavat myös haasteita niiden sovittamiselle nopeisiin ja epämuodollisiin ketteriin prosesseihin. Käytännössä myös ympäristö vaikuttaa siihen, miten käyttäjäkokemus otetaan huomioon ketterissä kehitysprojekteissa: organisaatio voi tukea tai haitata käyttäjäkokemuksen suunnittelua ketterissä kehitysprojekteissa. Ehdotettuja ratkaisuja olivat käyttäjälähtöisyyttä tukevat lisäykset ketteriin arvoihin, nopeat ja epämuodolliset käyttäjälähtöisen suunnittelun menetelmät sekä suunnittelijoiden ja kehittäjien välisen yhteistyön helpottaminen organisaatioissa.

Asiasanat: käyttäjäkokemus, ketterä ohjelmistokehitys, käyttäjälähtöinen suunnittelu

ABSTRACT

Terimaa, Tarmo
User experience design in agile development
Jyväskylä: University of Jyväskylä, 2019, 29 pp.
Information systems, bachelor's thesis
Supervisor(s): Palonen, Teija

In addition to technical features, modern information systems are also required to have a seamless user experience. User experience is a subjective, dynamic and context dependent concept that a user of a system experiences during the use. User-centered design is a process where a specific user experience is being designed and developed. To build high quality information systems, software development process also has to consider the financial and technical challenges. Agile software development was developed to address these challenges. Agile methods emphasize communication, customer collaboration and flexibility, so it is often thought that agile methods also promote good user experience. In practice it has been found that this is not always the case, as many software projects end up with great deal of problems in terms of user experience. In this bachelor's thesis I've done a literature review on integrating agile software development and user-centered design. Agile methods and user-centered design are being observed in general level but also more specifically by taking a look into user centeredness of Scrum, one of the most widely used agile framework. Integrating agile development and user-centered design has been found out to be challenging because of the lack of user focus in agile values and the differences between the two approaches. In this study I found out that most of the challenges are related to the fact that agile values do not support user-centricity well enough. The second issue is that heavy and formal user-experience design methods are difficult to integrate in to light and fast agile processes. Also, the environment where the development is being executed affects how well user experience is considered in the development process in practice. A development organization can either promote or hinder the user experience design activities. Solutions for the better integration of agile development and user-centric design found in this study were: adding more user focus to agile values, making user-centered design methods more informal and supporting the collaboration of agile developers and user experience designers.

Keywords: User experience, agile development, user-centered design

KUVIOT

SISÄLLYS

1 Johdanto

Ketterät menetelmät ovat vakiinnuttaneet asemansa ohjelmistokehityksessä monien niihin liittyvien etujen vuoksi. Ketterien ohjelmistokehitysmenetelmien ajatellaan monesti tukevan myös käyttäjälähtöisyyttä, koska ketterissä arvoissa korostetaan kommunikaatiota, asiakasyhteistyötä ja palautteen merkitystä. Ketterissä ohjelmistokehitysprojekteissa on kuitenkin huomattu, että niiden käyttö ei automaattisesti johda kehitettävän ohjelmiston hyvään käyttäjäkokemukseen (Jurca, Hellman & Maurer, 2014).

Tämä kandidaatintutkielma tarkastelee ketterän ohjelmistokehityksen ja käyttäjälähtöisen suunnittelun välisen integraation haasteita ja ratkaisuja kirjallisuuskatsauksen keinoin. Käyttäjälähtöisellä suunnittelulla tarkoitetaan prosessia, jolla tietynlainen käyttäjäkokemus pyritään saavuttamaan tuotetta tai palvelua kehitettäessä (Garret, 2010). Käyttäjäkokemus määritellään tässä tutkielmassa laajana kokonaisuutena, joka sisältää perinteisesti käytettävyyteen liitettävien toiminnallisten aspektien lisäksi myös käyttöön liittyvät hedonistiset tekijät. Täten käyttäjälähtöinen suunnittelu käsittää joukon erilaisia menetelmiä ja periaatteita, jotka ohjaavat kehitysprosessia kokonaisvaltaisesti miellyttävän järjestelmän kehittämiseksi. Ihmisen ja teknologian välisen vuorovaikutuksen tutkimuksessa käyttäjäkokemuksesta tuotetun tiedon vieminen käytännön ohjelmistokehitysprosesseihin on todettu haastavaksi (Memmel, Gundelsweiler & Reiterer, 2007). Raskaat ja tutkimusorientoituneet käyttäjälähtöisen suunnittelun menetelmät eivät istu nykyaikaisiin joustaviin ketterän ohjelmistokehityksen prosesseihin. Kuilu käyttäjälähtöisen suunnittelun akateemisen tutkimuksen ja sen käytännön toteutuksen välillä on todettu olevan edelleen suuri (Øvad & Larsen, 2015). Tässä tutkielmassa pyritään kuromaan umpeen tätä kuilua tutkimalla syitä sille, miksi käyttäjälähtöisyyttä on toisinaan vaikeaa edistää ketterissä ohjelmistokehitysprojekteissa.

Tutkielmassa on yksi päätutkimuskysymys ja kaksi apukysymystä. Päätutkimuskysymys on: ”Mitä haasteita käyttäjäkokemuksen suunnitteluun ketterissä ohjelmistokehitysprojekteissa liittyy ja mitä ratkaisuja näihin haasteisiin on esitetty?”. Päätutkimuskysymystä lähestytään kahden apukysymyksen avulla, jotka ovat: ”Mitä tarkoittaa käyttäjäkokemus ja miten sen suunnittelua toteutetaan käytännössä?” ja ”Mitä on ketterä ohjelmistokehitys?”.

Tutkielma toteutettiin kirjallisuuskatsauksena, jossa lähdeaineistona käytettiin pääasiassa vertaisarvoituja tieteellisiä julkaisuja. Tutkielmassa käytettyjä ei-tieteellisiä lähteitä käytettiin lähinnä viittaamaan kansainvälisiin standardeihin ja käytettävyyden käsitteen määrittelyyn. Valtaosa lähdeaineistosta haettiin Elsevierin Scopus sekä Googlen Scholar palveluista. Lähdeaineistojen laatua arvioitiin sekä niihin liittyvien viittausten lukumäärällä, että julkaisukanavan laadun perusteella. Laadun lisäksi arvioitiin lähdemateriaalin soveltuvuutta julkaisuvuoden ja sisällön perusteella.

Seuraava kappale käsittelee käyttäjäkokemusta, käytettävyyttä ja käyttäjälähtöistä suunnittelua sekä vastaa ensimmäiseen apukysymykseen. Käyttäjäkokemuksen käsitteestä ei ole täysin yksiselitteistä käsitystä, mikä aiheuttaa haasteita siihen liittyvään tutkimukseen sekä käytännön työhön. Kolmannessa kappaleessa käsitellään ketterää ohjelmistokehitystä ja vastataan toiseen apukysymykseen. Ketteriä menetelmiä käsitellään yleisellä tasolla niihin liittyvien periaatteiden ja arvojen kautta ja lisäksi tarkastellaan tarkemmin Scrum-menetelmää, joka on vakiinnuttanut asemansa yhtenä suosituimmista nykyaikaisista ohjelmistokehitysmenetelmistä. Viimeisessä sisältökappaleessa siirrytään varsinaiseen tutkimuskysymykseen, eli siihen mitä haasteita käyttäjälähtöisen suunnittelun ja ketterän ohjelmistokehityksen integrointiin liittyy ja mitä ratkaisuja näihin haasteisiin on esitetty. Integraation haasteita käsitellään yleisesti ketterien ja käyttäjälähtöisten arvojen kautta, mutta myös yksityiskohtaisemmin, tutkimalla kolmannessa kappaleessa esitellyn Scrum-menetelmän käyttäjälähtöisyyttä. Lopuksi vedetään yhteen tutkimuksen tulokset ja esitetään muutamia johtopäätöksiä sekä jatkotutkimusaiheita.

2 Käyttäjäkokemus

Käyttäjäkokemuksen määrittelyn vaikeus on yksi syy sille, miksi käyttäjäkokemuksen suunnittelua on ollut haastavaa viedä käytäntöön. Käsitteistä käyttäjäkokemus, käytettävyys ja käyttäjälähtöinen suunnittelu vallitsee erilaisia käsityksiä, joita tässä alaluvussa pyritään selvittämään ja määrittelemään. Aloitamme tarkastelemalla käyttäjäkokemuksen käsitettä ja siihen liittyviä haasteita. Sen jälkeen pohdimme, mitä on käytettävyys ja mikä on sen suhde käyttäjäkokemukseen. Tutkielman seuraavien lukujen kannalta on tärkeää, että määrittelemme perustellusti oman näkökulmamme edellä mainittuihin käsitteisiin ja niiden välisiin suhteisiin. Lopuksi määritellään, mitä käyttäjälähtöisellä suunnittelulla tarkoitetaan, sekä esitellään lyhyesti muutamia menetelmiä, joilla sitä toteutetaan käytännön ohjelmistokehityksessä. Käyttäjälähtöisen suunnittelun ominaispiirteiden ymmärtäminen on tärkeää, jotta myöhemmin on mahdollista tutkia käyttäjälähtöisen suunnittelun ja ketterän ohjelmistokehityksen integraatiota ja sen haasteita.

2.1 Käyttäjäkokemuksen käsite

Käyttäjäkokemuksen eksplisiittinen määrittely on hankalaa sen laaja-alaisten merkitysten vuoksi (Forlizzi & Battarbee, 2004). Siihen liittyy paljon epämääräisiä ja dynaamisia tekijöitä, kuten tunteet sekä hedonistiset ja esteettiset tekijät (Law, Roto, Hassenzahl, Vermeeren & Kort, 2009). Näitä tekijöitä yhdistää usein se, että niitä on vaikeaa mitata ja arvioida täsmällisesti. Käyttäjäkokemuksen käsitteen on katsottu laajentavan perinteistä näkemystä käytettävyydestä, joka korostaa käytön toiminnallisuuteen liittyviä tekijöitä, kuten esimerkiksi käyttäjän tavoitteiden saavuttamista mahdollisimman tehokkaasti (Law ym., 2009). Monimuotoisimmatkin käyttäjäkokemuksen määritelmät ovat samaa mieltä siitä, että sillä tarkoitetaan muutakin kuin pelkkää tuotteen hyödyllisyyttä ja käytettävyyttä (Väänänen-Vainio-Mattila, Roto & Hassenzahl, 2008). Suurin osa käyttäjäkokemuksen tutkijoista ja ammatinharjoittajista ovat samaa mieltä myös siitä, että käyttäjäkokemus on luonteeltaan dynaamista, kontekstista riippuvaista ja

subjektiivista (Law ym., 2009). ISO 9241-11 standardin mukaan käyttäjäkokemus tarkoittaa systeemin, tuotteen tai palvelun käytöstä tai sen odotetusta käytöstä tehtyjä havaintoja ja niistä aiheutuvia vasteita (International Organization for Standardization, 2018). Hassenzahlin ja Tractinskyn (2006) mukaan käyttäjäkokemus on seurausta käyttäjän sisäisestä tilasta, suunnitellun järjestelmän ominaisuuksista ja käytön kontekstista. Käyttäjän sisäisellä tilalla tarkoitetaan käyttäjän taipumuksia, odotuksia, tarpeita, motivaatioita ja mielialoja. Suunnitellun järjestelmän ominaisuuksilla tarkoitetaan järjestelmän monimutkaisuutta, tarkoitusta ja toiminnallisuutta. Käytön kontekstilla tarkoitetaan ympäristöä, kuten sosiaalista tai organisaationaalista ympäristöä, toiminnan merkityksellisyyttä ja käytön vapaaehtoisuutta (Hazzenhahl & Tractinsky 2006). Käyttäjäkokemukselle voidaan siis antaa erilaisia määritelmiä, joista vallitsee konsensus, mutta nämä määritelmät jättävät paljon tulkinnanvaraa käytännön toteutusta ajatellen.

2.2 Käytettävyys

ISO-9241-11 standardin mukaan käytettävyys tarkoittaa sitä, miten hyvin systeemiä, palvelua tai tuotetta voidaan käyttää tiettyjen käyttäjien toimesta heidän tavoitteiden saavuttamiseen vaikuttavasti, tehokkaasti ja miellyttävästi tietyssä kontekstissa. Vaikuttavuudella viitataan siihen, kuinka hyvin käyttäjät voivat saavuttaa tavoitteensa. Tehokkuudella tarkoitetaan sitä, kuinka paljon resursseja, esimerkiksi aikaa näiden tavoitteiden saavuttamiseen kuluu. Tyytyväisyys puolestaan viittaa siihen, kuinka hyvin käytöstä aiheutuvat tuntemukset vastaavat odotuksia (International Organization for Standardization, 2018). Nielsenin (2012) mukaan käytettävyys voidaan määritellä viidellä laatuattribuutilla, jotka ovat:

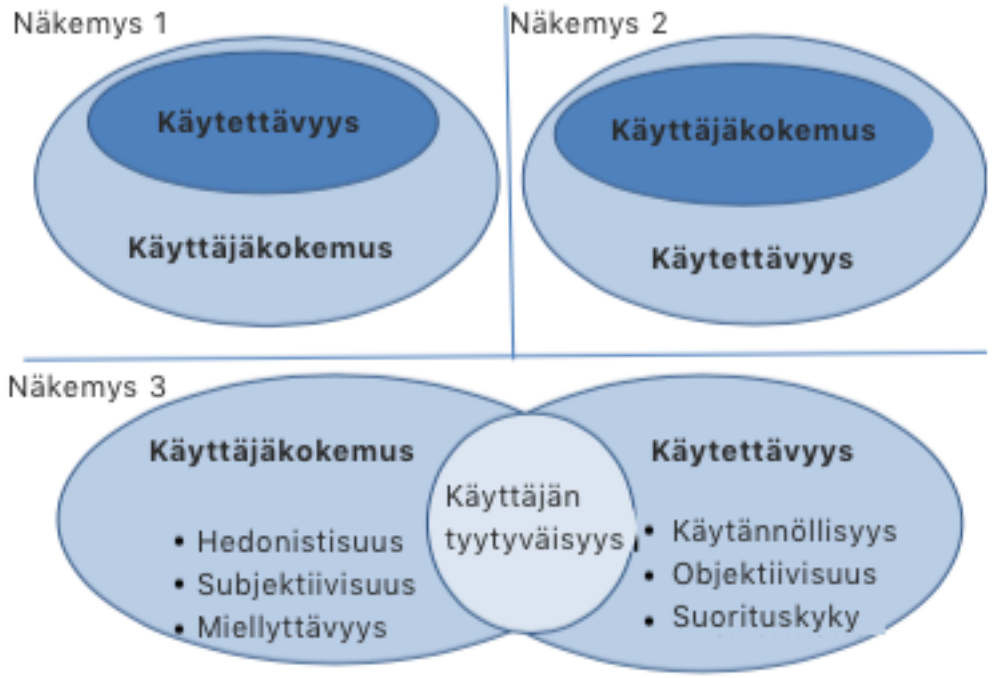
- **Opittavuus.** Kuinka helppoa uuden käyttäjän on oppia suorittamaan tehtäviä järjestelmässä?
- **Tehokkuus.** Kun käyttäjät ovat oppineet käyttämään järjestelmää, kuinka nopeasti he pystyvät suorittamaan tehtäviä?
- **Muistettavuus.** Kuinka hyvin käyttäjät säilyttävät osaamisensa järjestelmän käytön suhteen, kun he ovat jonkin aikaa käyttämättä järjestelmää?
- **Virheet.** Kuinka usein käyttäjät tekevät virheitä, kuinka vakavia ne ovat ja kuinka hyvin niistä voidaan palautua?
- **Tyytyväisyys.** Kuinka miellyttävää järjestelmän käyttö on?

Nielsenin (2012) mukaan yllä mainittujen laatuattribuuttien lisäksi tärkeä laatuattribuutti on hyöty. Hyödyllä tarkoitetaan sitä, tekeekö järjestelmä niitä asioita mitä käyttäjä haluaa. Yhdessä käytettävyyden kanssa hyöty muodostaa tuotteen lopullisen hyödyllisyyden (Nielsen, 2012). Helppokäyttöinen ja miellyttävä käyttöliittymä on turha, jos järjestelmä ei toteuta käyttäjän toiminnallisia tarpeita.

2.3 Käyttäjäkokemuksen ja käytettävyyden välinen suhde

Käyttäjäkokemuksen ja käytettävyyden välisestä suhteesta vallitsee useita erilaisia näkemyksiä. Joskus näitä termejä käytetään myös toistensa synonyymeinä. Käytettävyydellä voidaan katsoa tarkoitettavan enemmän tehtävien suorittamiseen liittyviä pragmaattisia tekijöitä, kun taas käyttäjäkokemus viittaa enemmän käytön hedonistisiin aspekteihin (Moczarny, Villiers & Biljon, 2012). Vaikka käytettävyyden määritelmissäkin huomioidaan kontekstin merkitys, voidaan katsoa, että käyttäjäkokemus korostaa vielä enemmän käytöstä aiheutuvan yksilöllisen kokemuksen merkitystä. Jurcan ym. (2014) mukaan näitä kahta käsitettä erottaa se, että käyttäjäkokemuksen prioriteettina on käyttäjän tyytyväisyys, kun taas käytettävyys keskittyy enemmän tuotteen toiminnallisten vaatimusten täyttymiseen. Moczarnyn ym. (2012) mukaan käyttäjäkokemus on ilmiö, missä kokonaisvaltainen kokemus on merkittävämpi kuin yksittäisistä käytettävyyden osatekijöistä muodostuva summa.

Alla on esitetty havainnollistava kuva kolmesta erilaisesta näkemyksestä käyttäjäkokemuksen ja käytettävyyden välisestä suhteesta (kuvio 1).



KUVIO 1 Käyttäjäkokemuksen ja käytettävyyden välinen suhde (Moczarny ym., 2012, s. 217)

Ensimmäisen näkemyksen mukaan käytettävyys on osa laajempaa käyttäjäkokemuksen käsitettä. Toisen näkemyksen mukaan käyttäjäkokemus on osa käytettävyyttä, ja vielä tarkemmin sen tyytyväisyys komponenttia. Kolmas näkemys ehdottaa, että käytettävyys ja käyttäjäkokemus ovat toisistaan erillisiä, mutta osittain päällekkäisiä ja joitain samoja ominaisuuksia ilmentäviä konsepteja (Moczarny ym., 2012). Yrityksissä tapahtuvan tuotekehityksen kontekstissa käyttäjäkokemus ymmärretään useimmiten ensimmäisen näkemyksen mukaisesti

laajempana kokonaisuutena, joka sisältää käytettävyyden (Väänänen-Vainio-Mattila ym., 2008).

Tässä kandidaatintutkielmassa tarkastelemme käyttäjäkokemusta kokonaisuutena, joka sisältää käytettävyyden, koska tämä vaikuttaa olevan vallitseva näkemys ammatinharjoittajien keskuudessa (Väänänen-Vainio-Mattila ym., 2008). Pelkkien hedonistisien tai pragmaattisten aspektien käsitteleminen erillisinä kokonaisuuksina ei ole käytännössä mielekästä, koska ne molemmat vaikuttavat toisiinsa. Heikot toiminnalliset ominaisuudet järjestelmässä aiheuttavat negatiivisia tuntemuksia ja toisaalta käytöstä aiheutuvat negatiiviset tuntemukset voivat vaikuttaa toiminnallisten tavoitteiden saavuttamiseen. Tämän kandidaatintutkielman keskiössä olevat käyttäjälähtöisen suunnittelun menetelmät pyrkivät kokonaisvaltaiseen käyttäjän ja tuotteen välisen vuorovaikutuksen parantamiseen. Käyttäjälähtöistä suunnittelua käsitellään tarkemmin seuraavassa alaluvussa.

2.4 Käyttäjälähtöinen suunnittelu

Gulliksen ym. (2014) määrittelevät käyttäjälähtöisen suunnittelun tarkoittavan prosessia, jossa keskitytään tuotteen käytettävyyteen sen kehityksen ja koko elinkaaren aikana. Tarkemman määritelmän saavuttamiseksi samat kirjoittajat esittävät seuraavia perusperiaatteita, joita käyttäjälähtöisessä suunnittelussa tulisi noudattaa:

- **Käyttäjälähtöisyys.** Käyttäjän tehtävät, tavoitteet, tarpeet tulisivat ohjata kehitystä jo varhaisessa vaiheessa.
- **Aktiivinen käyttäjien osallistaminen.** Käyttäjiä edustavat henkilöt tulisivat olla aktiivisesti mukana kehityksessä.
- **Iteratiivinen ja inkrementaalinen kehitys.** Kehitystä tulee tehdä sykleissä, joissa käyttäjien vaatimuksia ja tarpeita voidaan arvioida ja kehittää loppukäyttäjiltä hankitun palautteen perusteella.
- **Yksinkertainen suunnitelmien esitystapa.** Suunnitelmat tulee olla esitettävissä kaikille sidosryhmille ymmärrettävällä tavalla.
- **Aikainen ja jatkuva prototyyppien käyttö.** Prototyyppejä tulisi käyttää ideoiden validoimiseen jo kehityksen alkumetreiltä ja niitä tulee testata oikeilla käyttäjillä.
- **Käytön arvioiminen kontekstissa.** Käytettävyyttä tulee pyrkiä arvioimaan oikeissa käyttökonteksteissa.
- **Eksplisiittiset ja tietoiset suunnittelutoiminnot.** Käyttöliittymän suunnittelua tulee tehdä tietoisesti ja systemaattisesti.
- **Ammattimainen lähestymistapa.** Kehitysprosessia tulee suorittaa tehokas ja monialainen tiimi.
- **Käytettävyysekspertit.** Käytettävyyteen erikoistuneita asiantuntijoita tulee osallistaa kehitysprosessiin jo varhaisessa vaiheessa ja jatkuvasti koko kehitysprosessin ajan.

- **Holistinen lähestymistapa.** Ohjelmisto ei toimi tyhjiössä: myös muita ohjelmiston käyttöön liittyviä järjestelmiä, kuten organisaatiota, täytyy kehittää samanaikaisesti parhaan mahdollisen lopputuloksen saavuttamiseksi.
- **Prosessien muokattavuus.** Käyttäjälähtöisen suunnittelu tulee sopeuttaa jokaiseen ympäristöön erikseen.
- **Käyttäjälähtöinen asenne.** Kehitykseen osallistuvien sidosryhmien täytyy ymmärtää käytettävyyden ja käyttäjien osallistamisen tärkeys.

Vaikka Gulliksen ym. (2014) puhuvatkin käyttävyyteen keskittymisestä, voidaan käyttäjälähtöisen suunnittelun katsoa käsittävän laajemmin myös kokonaisvaltaisen käyttäjäkokemuksen suunnittelun ja toteutuksen (Garret, 2010). Käyttäjälähtöinen suunnittelu on siis prosessi, jonka tavoitteena on saavuttaa tietynlainen käyttäjäkokemus. Yrityksissä tapahtuvassa tuotekehityksessä käyttäjäkokemuksen suunnitteluun ovat erikoistuneet käyttäjäkokemuksen suunnittelijat. Ohjelmistokehitysprosessissa käyttäjäkokemukseen erikoistuneet suunnittelijat toimivat yhdessä mm. visuaalisten suunnittelijoiden ja ohjelmistokehittäjien kanssa.

Jurca ym. (2014) esittelevät lyhyesti yleisesti käytettyjä käyttäjälähtöisen suunnittelun menetelmiä, joilla käyttäjälähtöistä suunnittelua usein toteutetaan käytännössä. Nämä menetelmät voidaan karkeasti jakaa käyttäjien vaatimusten keräämiseen ja niiden analysoimiseen käytettyihin menetelmiin. Käyttäjien vaatimuksien keräämiseen käytettyjä menetelmiä ovat:

- **Fokusryhmähaastattelut.** Kuudesta yhdeksään käyttäjää kerätään samaan tilaan keskustelemaan tunteista ja ajatuksista käyttöliittymään liittyen. Valvoja on läsnä ohjaamassa keskustelua.
- **Korttien lajittelu.** Käyttäjät ryhmittelevät korteille kirjoitettuja ideoita järjestelmästä. Tuloksena syntyvien ryhmien pohjalta voidaan ymmärtää, mitkä asiat mielletään yhteenkuuluviksi.
- **Heuristinen arviointi.** Pieni ryhmä eksperttejä arvioi käyttöliittymää heuristiikoiksi kutsutuiden käytettävyysperiaatteiden mukaan.

Käyttäjien vaatimuksien analysointiin käytettyjä menetelmiä ovat:

- **Käyttäjäpersoonat.** Fiktiivinen persoona, joka kuvastaa tyypillistä käyttäjää. Käyttäjäpersoonien avulla käyttäjäkokemuksen suunnittelijat voivat samaistua käyttäjään paremmin ja perustella suunnittelutyössään tekemiään ratkaisuja.
- **Skenaariot.** Fiktiivinen kertomus, joka kuvastaa käyttäjän mahdollisesti kohtaamia ongelmia.

Käyttäjälähtöisessä suunnittelussa keskeistä on käyttäjän tiivis osallistaminen tuotekehitykseen. Gray (2016) selvitti haastattelututkimuksen avulla, mitä menetelmiä yritysympäristöissä toimivat käyttäjälähtöiseen suunnitteluun erikoistuneet ammattilaiset käyttävät. Tutkimuksessa löydettyjä menetelmiä olivat mm. käyttäjäpersoonat, käytettävyystestaus, ryhmähaastattelut ja erilaisten prototyyppien rakentaminen ja testaaminen. Tärkeä havainto tutkimuksessa oli, että

tiettyjen menetelmien hallitsemisen sijaan osallistujat pitivät tärkeämpänä, että suunnittelijalla on holistinen näkemys kokonaisuudesta, johon sisältyy suunnittelun ja käyttäjätutkimuksen lisäksi myös kommunikaatio eri sidosryhmien välillä. Tutkimuksen tuloksia tarkastellessa tulee ottaa huomioon, että kyseisessä tutkimuksessa kartoitettiin ammattilaisten mielipiteitä, ei parhaiksi todistettuja käytäntöjä (Gray, 2016). Tutkimuksen tulokset havainnollistavat hyvin sitä, miten käyttäjäkokemuksen tapaan myös käyttäjälähtöinen suunnittelu on vahvasti kontekstista riippuvaista. Tämä asettaa haasteita parhaiden käytänteiden kehittämiselle.

3 Ketterä ohjelmistokehitys

Tässä kappaleessa määritellään, mitä ketterä ohjelmistokehitys on, miten se eroaa perinteisestä ohjelmistokehityksestä sekä esitellään yksi yleisimmistä ketterän ohjelmistokehityksen menetelmistä nimeltään Scrum. Perinteistä ja ketterää ohjelmistokehitystä vertaillessa on tärkeää ottaa huomioon teknologinen konteksti, jossa kyseiset menetelmät ovat saaneet alkunsa. Ennen ketterien menetelmien yksityiskohtaisempaa tarkastelua tutustumme lyhyesti ketterän ohjelmistokehityksen taustoihin.

3.1 Taustaa

Tarve kevyemmille ohjelmistokehitysmenetelmille syntyi hiljalleen tekniikan kehittyessä ja ohjelmistoprojektien monimutkaisuuden kasvaessa. Epäonnistuneiden, viivästyneiden ja peruuntuneiden ohjelmistoprojektien suuri määrä pakotti ammatinharjoittajat miettimään vaihtoehtoa perinteisille suunnitelmalähtöisille ohjelmistokehitysmenetelmille (Anwer, Aftab, Shah & Waheed, 2017). Perinteiset ohjelmistokehitysmenetelmät koettiin ohjelmistokehittäjien keskuudessa haasteellisiksi toteuttaa dynaamisissa käytännön ympäristöissä (Abrahamsson, Salo, Ronkainen & Warsta, 2002). Abrahamssonin ym. (2002) mukaan Nandhakumar ja Avision (1999) tulkitsevat perinteisiä ohjelmistokehitysmenetelmiä käytettävän lähinnä antamaan valheellista kuvaa siitä, että kaikki on hallinnassa, vaikka todellisuudessa täydellistä hallintaa on hyvin vaikea saavuttaa jatkuvasti muuttuvassa ympäristössä. Myöhemmin Jiang ja Eberlein (2009) ovat kuitenkin todenneet, että ohjelmistoprojektin epäonnistuminen ei johdu menetelmien huonoudesta vaan siitä, että tiettyä menetelmää on sovellettu väärään kontekstiin.

1990-luvun puolivälissä joukko ammatinharjoittajia alkoi kehittää kevyiksi menetelmiksi kutsuttuja menetelmiä, joiden tarkoituksena oli kyetä mukautumaan paremmin muutoksiin, joita projekteissa väistämättä kohdattiin (Cohen, Lindvall & Costa, 2004). Näitä menetelmiä kutsuttiin aluksi kevyiksi menetelmiksi, kunnes vuonna 2001 joukko näiden menetelmien kehittäjiä kokoontui yhteen ja laati yhteisen julistuksen, jossa määriteltiin ketterän kehityksen 12

periaatetta ja neljä arvoa (Agile Alliance, 2001). Nämä arvot jakavaa menetelmien joukkoa ruvettiin kutsumaan ketteriksi menetelmiksi. Abrahamssonin ym. (2002) mukaan ketterien menetelmien kehityksen on laajalti tunnistettu alkaneeksi kunnolla sen jälkeen, kun Kent Bock esitteli ”extreme programming”-menetelmän vuonna 1999.

Ketterien menetelmien arvot ja periaatteet ovat merkittävin ketterää ja perinteistä lähestymistapaa erottava tekijä, sillä monet ketterien menetelmien käytänteistä eivät sinänsä ole uusia (Cohen ym., 2004). Myös Anwar ym. (2017) mukaan ketterän kehityksen periaatteet eivät olleet uusia, mutta niitä käytettiin uusilla tavoilla, jotka tekivät ohjelmistokehityksestä joustavampaa ja mukautuvampaa. Jiangin ja Eberleinin (2009) mukaan monien ketterien menetelmien taustalta löytyvä ajatus iteratiivisesta ja inkrementaalisesta kehittämisestä voidaan nähdä saaneen alkunsa jo 1930-luvulla.

3.2 Perinteinen ja ketterä ohjelmistokehitys

1960-luvun lopulta alkaen monia erilaisia ohjelmistokehitysmenetelmiä on kehitetty ja kokeiltu käytännössä (Huo, Verner, Zhu & Babar, 2004). Myös Abrahamsson ym. (2002) toteaa, että ohjelmistoalalla kehitettyjen erilaisten menetelmien lukumäärä on suuri. Kehitettyjen menetelmien monimuotoisuus ja suuri määrä ovat aiheuttaneet kiivasta keskustelua eri menetelmien paremmuudesta (Jiang & Eberlein, 2009). Nykyisin nämä menetelmät jaetaan useimmiten kahteen eri kategoriaan: perinteiset ja ketterät menetelmät.

Perinteisiä menetelmiä kutsutaan joskus myös suunnitelmalähtöisiksi menetelmiksi, joka kuvaa niille ominaista toimintatapaa pyrkiä suunnittelemaan mahdollisimman paljon etukäteen huolellisesti kerättyjen vaatimusten perusteella. Perinteiset menetelmät syntyivät aikana, jolloin tietokoneet olivat hitaita ja levytila kallista. Elektronisen tallennustilan kalleudesta ja dokumentaatiotyökalujen kehittymättömyydestä johtuen projekteissa jouduttiin palkkaamaan henkilöitä kirjoittamaan ja järjestelemään paperille kirjoitettua dokumentaatiota. Perinteiset menetelmät korostavat huolellista suunnittelua ja vaatimusmäärittelyä etukäteen, koska paperisten asiakirjojen muokkaaminen ja uudelleenjärjesteleminen oli työlästä (Jiang & Eberlein, 2009). Vaatimusten muuttuessa kesken projektin perinteiset ohjelmistokehitysmenetelmät ajautuvat usein vaikeuksiin, koska niitä ei ole kehitetty vastaamaan muuttuvaan ympäristöön.

Perinteisistä menetelmistä tunnetuin on vesiputousmalli, joka kehittyi 1960-luvulla, kun sekä tietokoneet että ohjelmointikielet olivat nykyiseen verrattuna tehottomia (Jiang & Eberlein, 2009). Vesiputousmallia on käytetty paljon niin pienissä kuin suurissakin ohjelmistoprojekteissa ja se on todettu toimivaksi erityisesti laajoissa ja monimutkaisissa projekteissa, joissa kohdealue tunnetaan hyvin (Huo ym., 2004). Kuten perinteisissä ohjelmistokehitysmenetelmissä yleensä, myös vesiputousmallissa prosessi aloitetaan määrittelemällä ja dokumentoimalla joukko vaatimuksia, jonka jälkeen siirrytään korkean tason suunnittelun kautta toteutukseen (Cohen ym., 2004). Vesiputousmalli jakaa ohjelmistokehitysprosessin tarkemmin viiteen eri vaiheeseen:

1) Vaatimusten määrittely ja analyysi
2) Järjestelmän suunnittelu
3) Toteutus ja yksikkötestaus
4) Integraatio ja järjestelmätestaus
5) Toiminta ja huolto

Vaiheet etenevät mallin nimen mukaisesti vesiputousmaisesti: kun yksi vaihe on ohitettu, siihen ei enää palata. Vesiputousmallin menestyksestä huolimatta sillä on myös useita heikkouksia, kuten heikko sopeutuminen muuttuviin vaatimuksiin ja hyvin formaalit toimintatavat projektin luonteesta riippumatta (Huo ym., 2004). Schwaberin (1997) mukaan suurin osa vesiputousmalliin liittyvistä ongelmista selittyy sen lineaarisella luonteella.

Ketterät ohjelmistokehitysmenetelmät syntyivät vastaamaan haasteisiin, joita ohjelmistokehitysprojektien jatkuva monimutkaistuminen ja liiketoimintaympäristön kiihtyvä muutos aiheutti. Perinteiset menetelmät eivät lineaarisen luonteensa vuoksi pystyneet reagoimaan muuttuviin vaatimuksiin riittävän tehokkaasti. Ketterien menetelmien syntyhetkellä teknologinen ympäristö oli muuttunut merkittävästi perinteisten menetelmien aikakaudelta. Tehokkaita tietokoneita oli saatavilla halvalla, tallennustila oli jatkuvasti halvempaa, ohjelmointikielet tehokkaampia ja dokumentaatiotyökalut huomattavasti aiempaa edistyneempiä. Tämä teknisesti mahdollisti ketterien menetelmien syntymisen, koska vaatimuksia ja dokumentaatiota pystyttiin muokkaamaan entistä kustannustehokkaammin kesken projektin (Jiang & Eberlein, 2009).

Ketterät menetelmät ovat menetelmiä, jotka jakavat ketterässä julistuksessa (engl. agile manifesto) määritellyt seuraavat arvot:

- **Yksilöt ja vuorovaikutus** ennen prosesseja ja työkaluja. Ensimmäinen arvo korostaa ihmisten välisen vuorovaikutuksen merkitystä institutionalisoitujen prosessien sijaan.

- **Toimiva ohjelmisto** ennen kokonaisvaltaista dokumentaatiota. Kehitettävästä ohjelmistosta julkaistaan säännöllisin väliajoin uusia testattuja ja toimivia versioita. Tämä pakottaa kehittäjät pitämään kirjoitettavan ohjelmakoodin yksinkertaisena.

- **Asiakasyhteistyö** ennen sopimusneuvotteluja. Kommunikaatiota kehittäjien ja asiakkaiden välillä arvostetaan tiukkoja sopimusneuvotteluja enemmän. Tiiviin asiakasyhteistyön ansiosta mahdollisesti muuttuvat vaatimukset saadaan selville ajoissa.

- **Muutokseen vastaaminen** ennen suunnitelman noudattamista. Kaikki ohjelmistokehitysprojektin sidosryhmät tulisivat tarpeen vaatiessa olla kykeneviä muuttamaan kehityksen suuntaa ja olemassa olevat sopimukset tulisi laatia siten, että tähän on mahdollisuus.

Ketterillä menetelmillä ei siis viitata yksittäiseen tarkasti määriteltyyn prosessiin, vaan joukkoon menetelmiä, jotka jakavat yllä mainitut arvot. Arvoja tulisi tulkita niin, että myös lihavoidulla tekstillä kirjoitettujen asioiden jälkeen mainituilla asioilla on arvoa, mutta ne ovat toissijaisia (Agile Alliance, 2001). Ketterien arvojen lisäksi niihin liittyy joukko periaatteita, joita ketterässä kehityksessä tulisi noudattaa. Nämä periaatteet on esitelty alla:

- Tärkein tavoite on asiakkaan tarpeiden tyydyttäminen varhaisen ja jatkuvan toimituksen avulla.
- Toivota tervetulleeksi muuttuvat vaatimukset myöhäisessäkin vaiheessa kehitystä. Ketterät prosessit valjastavat muutoksen asiakkaan kilpailueduksi.
- Toimita toimivaa ohjelmakoodia säännöllisesti ja riittävän usein asiakkaalle.
- Liiketoiminnan kanssa työskentelevien henkilöiden ja ohjelmistokehittäjien tulee työskennellä yhdessä päivittäin projektin aikana.
- Rakenna projektit osaavien henkilöiden varaan. Luota siihen, että he saavat työn tehdyksi, kunhan heille tarjotaan siihen tarvittava tuki.
- Kasvotusten käytävät keskustelut ovat tehokkain tapa tiedon siirtämiseen kehitystiimin sisällä.
- Toimiva ohjelmisto on pääasiallinen mittari edistymiselle.
- Ketterät prosessit pyrkivät ohjelmistokehityksen kestävyyteen. Kehitystiimin tulisi voida säilyttää sama tahti kehityksessä jatkuvasti.

Tutkielman aiheen kannalta näistä arvoista ja periaatteista on tärkeää huomata, että ne eivät suoraan ota kantaa käyttäjäkokemukseen tai käyttäjälähtöiseen suunnitteluun erikoistuneiden asiantuntijoiden työskentelyyn kehitystiimissä, vaikka esimerkiksi liiketoimintahenkilöiden ja kehittäjien välistä päivittäistä yhteistyötä pidetäänkin tärkeänä periaatteena. Ketterän julistuksen laatimiseen ei tiettävästi osallistunut käytettävyysasiantuntijoita (Constantine, 2002).

3.3 Scrum

Scrum on yksi suosituimmista ketterän ohjelmistokehityksen menetelmistä (Anwar ym., 2017). Ensimmäisiä kirjallisuusviittauksia Scrumista voidaan katsoa löytyneen Takeuchin ja Nonakan (1986) artikkelista, jossa esitellään Japanista lähtöisin oleva nopea, mukautuva ja itseohjautuva tuotekehitysprosessi (Abrahamsson ym., 2002). Nimensä Scrum on saanut rugby pelin strategiasta, jossa pelistä poistunut pallo saadaan takaisin peliin tiimityön avulla (Schwaber & Beedle 2002).

Perinteiset menetelmät, kuten vesiputousmalli olettavat järjestelmäkehitysprosessien olevan ennalta määriteltyjä, vaikka todellisuudessa ne muistuttavat usein enemmän mustaa laatikkoa: syötteen perusteella ei voida päätellä, mitä tulee ulos. Perinteiset menetelmät eivät tunnusta järjestelmäkehitysprosessien

vaikeaa ennustettavuutta, mistä seuraa, ettei kokonaisuutta pystytä hallita, koska odottamattomiin tuloksiin ei olla varauduttu. Scrum tunnustaa järjestelmäkehitysprosessien ennustamattomuuden ja pyrkii näin säilyttämään hallinnan koko kehitysprosessissa (Schwaber, 1997). Se ei määrittele tiukasti, miten kehitysprosessit tulisi käytännössä toteuttaa vaan keskittyy tiimin jäsenten välisen toiminnan järjestämiseen siten, että jatkuvasti muuttuvassa ympäristössä tapahtuva järjestelmäkehitys olisi mahdollisimman sujuvaa (Abrahamsson ym., 2002).

Ketterien menetelmien tapaan myös Scrumin kehittymisen taustalta voidaan löytää teknologisen ja liiketoiminnallisen ympäristön muutoksen aiheuttamat haasteet. Scrumin perusajatus on, että järjestelmäkehitysprosessit tapahtuvat monimutkaisissa ympäristöissä, jotka sisältävät useita ympäristöön ja teknologiaan liittyviä muuttujia, kuten esimerkiksi vaatimukset, teknologia ja resurssit. Näiden tekijöiden muuttumista voidaan pitää todennäköisenä, joten muutokseen vastaamiseksi ohjelmistokehitysprosessin on oltava riittävän joustava (Abrahamsson ym., 2002)

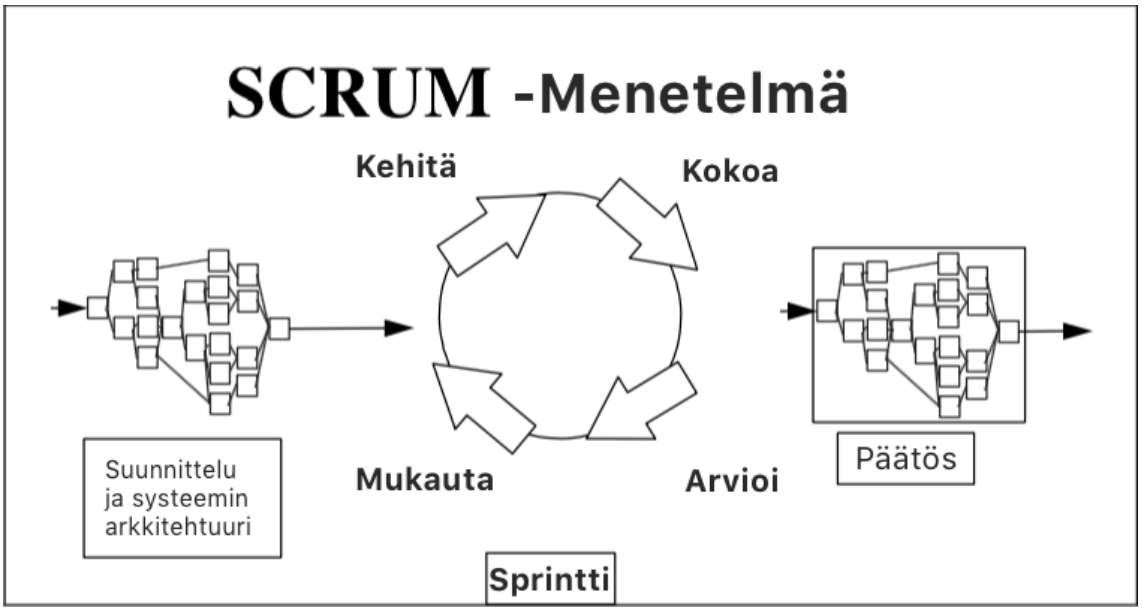
3.3.1 Roolit

Ennen Scrum-prosessin yksityiskohtaisempaa tarkastelua on hyvä katsoa lyhyesti erilaisia Scrum-prosessiin liittyviä rooleja. Scrum-prosessista voidaan tunnistaa kuusi erilaista roolia, jotka Schwaber ja Beedle (2002) määrittelevät seuraavasti:

- **Scrum-mestari** tuntee Scrum -prosessin käytänteet ja varmistaa, että niitä noudatetaan prosessissa. Scrum-mestari kommunikoi sekä asiakkaan, että kehitystiimin kanssa ja varmistaa prosessin tuottavuuden.
- **Tuotteen omistaja** (engl. product owner) on vastuussa projektin johtamiseen ja valvontaan liittyvistä tehtävistä. Tuotteen omistaja osallistuu kehitysjonossa olevien tehtävien työmäärien arviointiin ja päättää viime kädessä kehitysjonossa olevien tehtävien priorisoinnista.
- **Scrum tiimi** on projektitiimi, jolla on valtuudet toimia järjestäytyä ja haluamillaan tavoilla tavoitteiden saavuttamiseksi. Kehitystyön lisäksi Scrum tiimi osallistuu tehtävien työmäärän arviointiin, sprintin kehitysjonon luomiseen sekä tuotteen kehitysjonon arviointiin ja muokkaamiseen.
- **Asiakas** osallistuu tuotteen kehitysjonon luomiseen, kun uutta tuotetta aletaan kehittämään tai vanhaa parantelemaan.
- **Johto** määrittelee projektissa käytettävät standardit ja käytänteet. Johdolla on viimeinen päätäntävalta projektissa tehtävistä päätöksistä. Johto osallistuu projektin tavoitteiden ja vaatimusten määrittelyyn.
- **Käyttäjä** osallistuu sprintin suunnitteluun ja arviointiin.

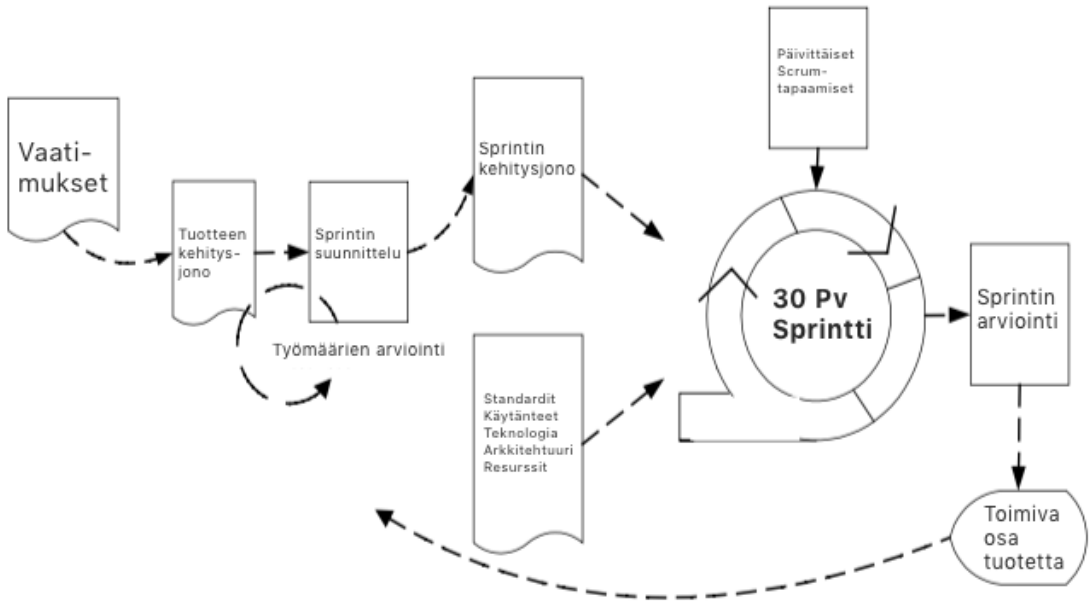
3.3.2 Prosessi

Scrum koostuu kolmesta alla kuvatusta vaiheesta (kuvio 2).



KUVIO 2 Scrum-prosessi (Schwaber, 1997, s. 10)

Näistä vaiheista ainoastaan suunnitteluvaihe ja projektin päätösvaihe ovat tarkasti määriteltyjä prosesseja. Keskimmäinen, yleensä kehitys- tai sprinttivaiheeksi kutsuttu vaihe on löyhästi määritelty empiirinen prosessi (Schwaber, 1997; Abrahamsson ym., 2002). Ensimmäinen vaihe voidaan katsoa koostuvan kahdesta alavaiheesta: suunnittelu- ja arkkitehtuurivaihe. Suunnitteluvaiheessa kerätään toistaiseksi tiedossa olevat vaatimukset ja luodaan niiden perusteella tuotteen kehitysjono (engl. product backlog). Tuotteen kehitysjono sisältää tuotteelta vaadittuja ominaisuuksia ja arvioita siitä, kuinka kauan kunkin ominaisuuden toteuttaminen vie aikaa (Abrahamsson ym., 2002). Arkkitehtuurivaiheessa tehdään korkean tason suunnitelmat siitä, kuinka kehitysjonossa olevat asiat toteutetaan käytännössä (Schwaber, 1997). Keskimmäinen, kehitysvaiheeksikin kutsuttu vaihe, on iteratiivinen prosessi, jossa toteutetaan sprinteiksi kutsuttuja kehityssyklejä. Yksi sprintti kestää tavallisesti 1-4 viikkoa (Schwaber, 1997). Yksittäisen sprintin vaiheet on esitetty alla (Kuvio 3).



KUVIO 3 Scrum-sprintti (Abrahamsson ym., 2002, s. 28)

Sprintti aloitetaan kahdesta vaiheesta koostuvalla Scrum-mestarin järjestämällä sprintin suunnittelutapaamisella, jonka ensimmäisessä vaiheessa määritellään sprintissä toteutettavan tuotteen toiminnallisuudet ja seuraavassa vaiheessa suunnitellaan niiden toteutus. Toteutus aloitetaan sprintin kehitysjonon pohjalta, joka on tuotteen kehitysjonosta poimittu joukko ominaisuuksia, jotka kyseisessä sprintissä toteutetaan. Sprintin aikana tiimi kokoontuu päivittäin lyhyisiin Scrum-palavereihin keskustelemaan siitä, mitä on saatu aikaan ja mitä seuraavaksi ollaan tekemässä (Abrahamsson ym., 2002). Jokaista sprinttiä seuraa arviointi, jossa eri sidosryhmät arvioivat toteutettuja ominaisuuksia ja tarvittaessa lisäävät uusia tehtäviä kehitysjonoon. Lyhyet sprintit mahdollistavat palautteen keräämisen riittävän usein, jotta muuttuviin vaatimuksiin ja mahdollisiin vikoihin tuotteessa voidaan reagoida ajoissa (Schwaber, 1997).

Päätösvaiheeseen siirrytään, kun projektin johto toteaa, että tuote täyttää sille asetetut vaatimukset. Viimeinen vaihe valmistelee tuotteen yleistä julkaisua varten. Päätösvaiheen tehtäviin kuuluvat esimerkiksi integraatio, järjestelmätestaus ja markkinointimateriaalien valmistelu (Schwaber, 1997).

Yllä kuvatuista vaiheista on helppo huomata, että Scrum korostaa yksilöiden välistä vuorovaikutusta ja yhteistyötä eikä ota kovinkaan yksityiskohtaista kantaa siihen, miten kehitystyötä tulee toteuttaa käytännössä. Mallin ketteryys konkretisoituu kehitysvaiheessa, joka on suunniteltu mukautumaan muuttuviin vaatimuksiin lyhyillä kehityssykleillä, tiiviillä kommunikaatiolla ja mahdollisuudella muuttaa olemassa olevia vaatimuksia tarvittaessa.

4 Käyttäjäkokemus ketterissä menetelmissä

Larusdottirin, Gulliksenin ja Cajanderin (2017) mukaan Baxter ja Somerville (2011) väittävät, että ketteriä menetelmiä käytettäessä tuotetaan automaattisesti myös käyttäjäystävällisiä järjestelmiä. Jian, Larusdottirin ja Cajanderin (2012) tekemän kattavan kirjallisuuskatsauksen perusteella ketterät menetelmät eivät kuitenkaan huomioi loppukäyttäjiä riittävästi. Myös Singh (2008) toteaa, että perinteinen Scrum ei ole paras vaihtoehto kehitysprosessiin, jossa käytettävyys vaatii paljon huomiota. Tässä kappaleessa vastataan päätutkimuskysymykseen: ”Mitä haasteita käyttäjäkokemuksen suunnitteluun liittyy ketterissä ohjelmistokehitysprojekteissa ja mitä ratkaisuja näihin haasteisiin on olemassa olevassa kirjallisuudessa esitetty?”. Ketterät menetelmät vastaavat pääasiassa ohjelmistokehityksen teknisiin ja taloudellisiin haasteisiin, kun taas käyttäjälähtöinen suunnittelu pyrkii mahdollisimman hyvään käyttäjäkokemukseen. Molemmat menetelmät pyrkivät laadukkaan ohjelmiston tuottamiseen, mutta lähestymistapojen erilaisuus aiheuttaa haasteita niiden yhdistämiselle. Haasteet liittyvät lähestymistapojen erilaisuuden lisäksi ketterässä julistuksessa määriteltyihin perusarvoihin sekä toimintaympäristöön. Ensimmäisessä alaluvussa tarkastellaan ketterässä julistuksessa määriteltyihin arvoihin liittyviä ongelmia käyttäjälähtöisyyden suhteen. Toisessa alaluvussa pohditaan ketterän ja käyttäjälähtöisen lähestymistavan eroavaisuuksia ja niistä aiheutuvia haasteita. Viimeinen alaluku käsittelee toimintaympäristön vaikutusta ketterien kehitysprojektien käyttäjälähtöisyydelle.

4.1 Ketterät arvot ja käyttäjäkokemus

Ketterät menetelmät jakavat joukon aiemmin esiteltyjä arvoja ja periaatteita, jotka määrittelevät niiden tapaa lähestyä ohjelmistokehitysprosessia. Suuri osa ketteriin menetelmiin liittyvistä ongelmista käyttäjälähtöisyyden suhteen voidaan yhdistää näihin arvoihin, joissa ei oteta selkeästi kantaa käyttäjäkokemuksen suunnitteluun liittyviin asioihin. On huomionarvoista, että ketterän

julistuksen laatimiseen ei tiettävästi osallistunut käyttäjälähtöisyyteen erikoistuneita ammattilaisia (Constantine, 2002).

Ketterien menetelmien arvot korostavat yhteistyötä asiakkaan kanssa, mutta asiakkaan tunnistaminen on osoittautunut käytännössä haastavaksi. Larusdottirin ym. (2017) mukaan jotkut ketterän kehityksen ammattilaisista määrittelevät asiakkaaksi tuotteen omistajan, jotkut ohjelmistosta maksavan asiakkaan ja jotkut varsinaisen loppukäyttäjän. Samat kirjoittajat ovat huomanneet, että tuotteen omistajan rooliin valitaan usein henkilö ohjelmistoa kehittävän organisaation sisältä sen sijaan, että valittaisiin edustaja asiakasorganisaatiosta tai todellisista loppukäyttäjistä (Larusdottir ym., 2017). Lisäksi Singhin (2008) mukaan tuotteen omistajan roolissa toimiva henkilö on usein epäpätevä tai liian kiireinen ottamaan huomioon käyttäjäkokemukseen liittyviä tekijöitä kehitysprosessissa. Tuotteen omistaja on tärkeä rooli Scrum-prosessissa, sillä hänellä on valta priorisoida kehitysjonossa olevia tehtäviä (Abrahamsson ym., 2002). Jos tuotteen omistaja ei edusta todellista loppukäyttäjää, on kyseenalaista osaako hän priorisoida tehtäviä loppukäyttäjän kannalta oikealla tavalla.

Toimivan ohjelmiston korostaminen yli kokonaisvaltaisen dokumentaation tarkoittaa käytännössä sitä, että kehitetään pieniä toiminnallisia kokonaisuuksia ohjelmistosta nopeissa sprinteissä ja arvioidaan vaatimuksia jatkuvasti uudelleen. Käyttäjäkokemuksen kanssa työskentelevät asiantuntijat ovat raportoineet, että tämä vaikeuttaa kokonaisvaltaisen vision hahmottamista tuotteen käyttäjäkokemuksesta (Larusdottir, Cajander & Gulliksen, 2012). Toisaalta toimivan ohjelmiston tuottaminen käyttäjille mahdollisimman usein nopeuttaa palautteen saamista käyttäjiltä, joten näin ollen se voidaan nähdä myös käyttäjälähtöisyyttä tukevana tekijänä (Jurca ym., 2014).

Muutokseen vastaamisen korostaminen yli suunnitelmien noudattamisen näkyy ketterissä menetelmissä vähäisenä etukäteen tehtävänä suunnitteluna. Muutokseen vastaamisella on tarkoitus korostaa kehityksen aikana kerättävän palautteen tärkeyttä kehitystä ohjaavana tekijänä sen sijaan, että tehtäisiin kattavia suunnitelmia etukäteen. Larusdottir ym. (2017) toteaa, että toisinaan ketteriä menetelmiä käyttävät ammattilaiset vähättelevät suunnittelun merkitystä liikaa. Vähäistä suunnittelua perustellaan sillä, että käyttäjien vaatimukset muuttuvat jatkuvasti, joten on turhaa suunnitella mitään etukäteen. On kuitenkin argumentoitu, että käyttäjien fundamentaaliset vaatimukset käyttäjäkokemukselle eivät muutu niin paljoa, ettei sitä voitaisi suunnitella etukäteen jossain määrin (Larusdottir ym., 2012). Muuttuviin vaatimuksiin vastataan keräämällä käyttäjiltä palautetta ohjelmistosta ja sen jälkeen lisäämällä vaadittuja ominaisuuksia kehitysjonoon. Käytännössä on kuitenkin huomattu, että toiminnalliset ongelmat tavataan yleensä priorisoida käytettävyyteen liittyviä ongelmia korkeammalle (Larusdottir ym., 2017, Kuusinen, 2014). Kärjistetysti voidaan siis sanoa, että muuttuviin vaatimuksiin vastataan, jos ne liittyvät toiminnallisiin ominaisuuksiin.

Ketterissä arvoissa painotetaan yksilöitä ja vuorovaikutusta ennen prosesseja ja työkaluja. Kehitystiimin tulisi olla joustava ja itseorganisoituva. Itseorganisoituvuus on käyttäjäkokemuksen suunnittelijoita ajatellen hankalaa, koska he eivät oletusarvoisesti ole osana kehitystiimiä. Ketterissä kehitystiimeissä on havaittu vallitsevan huono käsitys siitä, kuka on vastuussa käyttäjäkokemukseen

liittyvistä asioista (Larusdottir ym., 2017). Tämä voi johtua siitä, että ketterässä manifestissa yksilöt ja vuorovaikutus on määritelty hyvin väljästi.

Ketterässä julistuksessa määritellyt arvot eivät määrittele riittävän selkeästi sitä, kuka tulisi olla vastuussa käyttäjäkokemuksesta, millä tavoin käyttäjiä tulisi osallistaa, tai missä vaiheessa käyttäjäkokemuksen suunnittelua tulisi toteuttaa ja miten (Agile Alliance, 2001). Blomkvist (2006) kuitenkin sanoo, että ketterät arvot eivät toimi täysin käyttäjälähtöistä suunnittelua vastaan, vaan niistä puuttuu riittävä painotus käytettävyyttä kohtaan. Ratkaisuksi ketterien arvojen puutteisiin Larusdottir ym. (2017) esittää seuraavia lisäyksiä ketteriin arvoihin:

1. Yksilöt ja vuorovaikutus.
    - Määrittele kaikille rooleille vastuu, joka niille kuuluu käyttäjäkokemukseen ja käytettävyyteen liittyvistä asioista.
    - Käyttäjäkokemuksesta vastuussa olevien henkilöiden täytyy olla säännöllisesti yhteydessä loppukäyttäjiin.
    - Osallista käyttäjiä kehitysprosessiin kasvotusten käydyn keskustelun lisäksi esimerkiksi sosiaalisen median kanavia hyödyntäen. Tämä on kevyt vaihtoehto käyttäjätutkimukselle ja tukee siten ketterää ajattelutapaa.
2. Toimiva ohjelmisto.
    - Aseta selkeä visio tuotteen käyttäjäkokemukselle, jotta kokonaiskuva kehitettävän tuotteen käyttäjäkokemuksesta pysyy kirkkaampana.
    - Aseta tuotteen käyttäjäkokemukselle selkeät tavoitteet ja mittaa niitä säännöllisesti.
3. Asiakasyhteistyö.
    - Mittaa käyttäjien vaatimuksien toteutumista oikeiden käyttäjien avulla.
    - Arvioi tuotteen arvoa käyttäjälle säännöllisesti.
    - Anna käyttäjäkokemuksesta vastuussa olevalle henkilölle riittävät valtuudet projektin suunnitteluun liittyen.
    - Laadi kehitystiimin sisäinen viestintäsuunnitelma, jotta kaikki ymmärtävät käyttäjäkokemuksen arviointien tulokset ja merkityksen.
    - Arviointien tuloksien täytyy johtaa toimenpiteisiin.
4. Muutokseen vastaaminen.
    - Määrittele teemoja sprinttien jälkeisille tapaamisille ja ota yhdeksi teemaksi käyttäjäkokemus.
    - Priorisoi käyttäjiltä tulevat muutosehdotukset korkealle.

Nämä lisäykset kattavat teoriassa hyvin pitkälti tutkielmassa tarkastellusta kirjallisuudesta löytyneet ketteriin arvoihin liittyvät puutteet käyttäjäkokemuksen suunnittelua koskien. Ketterien arvojen täydentäminen voidaan katsoa olevan ketteryyttä tukeva ratkaisu, koska siinä ei pyritä määrittelemään täsmällisesti kehitysprosessia vaan ainoastaan uudistamaan sitä ohjaavia ketteryyden arvoja. Yksityiskohtaisempiakin ratkaisuja ketterien menetelmien käyttäjälähtöisyyden

parantamiseksi on esitetty mm. Singhin (2008) toimesta, joka esittää vaihtoehtoisen Scrum-mallin, jossa tuotteen omistajan rooli jaetaan kahdelle henkilölle, joista toinen toimii kuten perinteisessä Scrum-prosessissa ja toinen keskittyy käyttäjäkokemukseen liittyviin asioihin.

4.2 Erilaiset lähestymistavat haasteena integraatiolle

Ketterä ja käyttäjälähtöinen järjestelmäkehitys tähtäävät kummatkin laadukkaan ohjelmiston tuottamiseen, mutta lähestyvät työtä hieman eri näkökulmista (Jurca ym., 2014). Käyttäjälähtöisen järjestelmäkehityksen tarkoitus on saada aikaan mahdollisimman käyttäjäystävällinen järjestelmä. Tähän tavoitteeseen pyritään iteratiivisen käyttäjiä osallistavan prosessin kautta (Larusdottir ym., 2017). Tämä prosessi sisältää usein paljon etukäteen tehtävää käyttäjätutkimusta, suunnittelua ja dokumentointia, jonka voidaan nähdä olevan ristiriidassa ketterien menetelmien keveyttä ja nopeutta kuvastavan arvomaailman kanssa. Siinä missä käyttäjälähtöiset menetelmät keskittyvät suunnittelemaan käyttäjän ja tuotteen välistä vuorovaikutusta, ketterät menetelmät pyrkivät vastaamaan lähinnä ohjelmistokoodin tuotantoon ja projektinhallintaan liittyviin kysymyksiin (Silva, Silveira, Maurer & Hellman, 2012; Ferreira, Sharp & Robinson, 2011). Yhteistä menetelmille on se, että ne ovat molemmat ihmiskeskeisiä ja sykleissä toteutettavia (Jurca ym., 2014).

Yksi merkittävimmistä eroista näiden kahden lähestymistavan välillä on niiden tapa allokoida resursseja projektin aikana. Ketterät menetelmät pyrkivät toimittamaan pieniä osia toimivaa ohjelmakoodia tasaisen nopealla tahdilla, kun taas käyttäjälähtöisessä suunnittelussa käytetään paljon resursseja tutkimukseen ennen kehityksen aloittamista (Silva, Martin, Maurer & Silveira, 2011). Käyttäjälähtöisen suunnittelun menetelmien onkin usein sanottu olevan liian täsmällisiä ja aikaa vieviä mukautuakseen ketterien menetelmien keveyttä ja nopeutta korostavaan arvomaailmaan (Larusdottir ym., 2017). Käyttäjälähtöisen suunnittelun menetelmiä muokataan käytännössä usein vähemmän aikaa vieviksi, jotta niitä voitaisiin toteuttaa ketterän kehityksen lyhyissä iteraatioissa (Ferreira, 2012).

Käytännössä kaikkein käytetyimmiksi käyttäjälähtöisen suunnittelun menetelmiksi Scrum projekteissa on todettu Jian ym. (2012) tekemän tutkimuksen mukaan työpajat ja yksinkertaiset prototyypit. Useimmin käytettyjä tekniikoita yhdisti epämuodollisuus ja nopeus (Jia ym., 2012). Larusdottir ym. (2017) ehdottaa, että yksi ratkaisu Scrumin ja käyttäjälähtöisen suunnittelun parempaan integroimiseen voisi olla käyttäjälähtöisen suunnittelun standardien madaltaminen ja muodollisista toimintatavoista luopuminen. Tästä esimerkkinä mainitaan sosiaalisen median kanavien hyödyntäminen käyttäjäpalautetta kerätessä (Larusdottir ym., 2017).

4.3 Ympäristön vaikutus käyttäjäkokemuksen suunnitteluun

Ferreira ym. (2011) mukaan olemassa oleva kirjallisuus tarkastelee käyttäjälähtöisen suunnittelun ja ketterän kehittämisen integroimista lähinnä kahden prosessin yhdistämisenä. Samojen kirjoittajien mukaan prosesseihin ja menetelmiin keskittymisessä on ongelmana se, että niitä harvoin toteutetaan täsmällisesti käytännössä (Ferreira ym., 2011). Ketterän kehittämisen ja käyttäjälähtöisen suunnittelun integroimisen haasteita tulisi tarkastella kokonaisuutena, johon liittyy prosessien lisäksi myös ympäristöt, jossa niitä toteutetaan. Ketterässä manifestissa määriteltyjen perusarvojen lisäksi myös organisaation sisällä vallitsevat arvot ja käytänteet vaikuttavat siihen, miten käyttäjäkokemus otetaan huomioon ketterässä ohjelmistokehityksessä.

Arvot ja olettamukset ohjaavat näkemyksiä siitä, kuinka suunnittelijoiden ja kehittäjien työ tulisi yhdistää laadukkaan ohjelmiston tuottamiseksi (Ferreira, 2012). Ferreira ym. (2011) tekemässä tutkimuksessa yrityksen johto oli päättänyt erottaa kehittäjät ja suunnittelijat toisistaan fyysisesti. Päätös perustui käyttäjäkokemuksen suunnittelijoiden johtajan näkemykseen siitä, että suunnittelijat työskentelevät paremmin, kun he eivät joudu häiriintymään ohjelmiston toteutukseen liittyvistä asioista. Tämä johti käytännössä siihen, että kehittäjien ja suunnittelijoiden välillä ei useinkaan vallinnut yhteisymmärrystä kehityksen tilasta (Ferreira ym., 2011).

Toiseksi merkittäväksi ongelmaksi organisaatioissa on todettu se, että käyttäjälähtöisyyteen panostetaan liian vähän. Usein ketterissä kehitysprojekteissa on osallisena liian vähän käyttäjäkokemuksen suunnittelijoita. Tästä johtuen yksi suunnittelija joutuu osallistumaan useaan projektiin, mikä vaikeuttaa tehtävien suorittamista ja vision säilyttämistä jokaisessa projektissa (Jurca ym., 2014). Silva ym. (2012) uskovat, että käyttäjäkokemuksen suunnittelijoiden kiireisyys on merkittävin ongelma ketterän kehityksen ja käyttäjälähtöisen suunnittelun yhdistämiselle käytännön tasolla. Jurca ym. (2014) mukaan Raisonin ym. (2013) tekemässä tutkimuksessa todettiin, että käyttäjälähtöistä suunnittelua pidettiin usein toissijaisena aktiviteettina kehitystyöhön nähden, eikä käyttäjäkokemuksen suunnittelijoita pidetty ketterän kehitystiimin vakituisina jäseninä.

Organisaation sisällä vallitseva kulttuuri vaikuttaa siihen, miten käyttäjäkokemukseen liittyvään työhön suhtaudutaan. Organisaation sisällä vallitseva kulttuuri kannustaa tai rajoittaa yksilöiden toimintaa (Ferreira, 2012). Ferreira ym. (2011) ehdottaa, että ketterän kehityksen ja käyttäjäkokemuksen suunnittelun yhdistämisessä organisaation tulisi toimia työntekijöiden välisen yhteistyön mahdollistajana, eikä pyrkiä liikaa ohjaamaan sitä esimerkiksi johtohenkilöiden subjektiivisten näkemysten perusteella. Käyttäjäkokemuksen suunnittelijat tulisi lukea paremmin osaksi itseorganisoituvaa kehitystiimiä. Yhteistyön parantamiseksi on ehdotettu esimerkiksi sitä, että käyttäjäkokemuksen suunnittelijoilla ja kehittäjillä olisi yhteinen työskentelytila (Jurca ym., 2014). Puutteelliset panostukset käyttäjäkokemukseen voivat johtua siitä, että sen arvoa ei ymmärretä riittävän hyvin tai sitä ei voida määrittää riittävän tarkasti, jotta tehtävät panostukset voitaisiin perustella yrityksen johdolle. Käyttäjäkokemus on luonteeltaan vaikeasti mitattavissa, kuten aikaisemmin on esitetty. Organisaatiokulttuurin lisäksi

myös aiemmin tässä tutkielmassa esitellyt ongelmat ketterässä julistuksessa vaikuttavat siihen, ettei käyttäjäkokemuksen suunnittelijoita aina koeta osana ketterää kehitystiimiä.

5 Yhteenveto ja johtopäätökset

Tutkielmassa pyrittiin selvittämään käyttäjälähtöisen suunnittelun ja ketterän ohjelmistokehityksen integraatiota ja sen haasteita. Varsinaista tutkimuskysymystä lähestyttiin kahden apukysymyksen kautta, joista ensimmäinen käsitteli käyttäjäkokemusta ja toinen ketterää ohjelmistokehitystä. Tutkielmassa havaittiin, että käyttäjäkokemus on laaja käsite, joka tarkoittaa käyttäjän kokemaa subjektiivista, dynaamista ja kontekstista riippuvaista kokemusta. Käyttäjäkokemuksen suunnittelusta puhuttaessa käytetään usein termiä käyttäjälähtöinen suunnittelu, jonka tavoitteena on toteuttaa mahdollisimman käyttäjäystävällinen järjestelmä. Päätutkimuskysymyksenä tutkielmassa oli: ”Mitä haasteita käyttäjäkokemuksen suunnitteluun ketterissä ohjelmistokehitysprojekteissa liittyy ja mitä ratkaisuja näihin haasteisiin on esitetty?”. Tähän liittyen tutkielmassa havaittiin, että käyttäjälähtöisen suunnittelun ja ketterän kehittämisen integraatiota tulee lähestyä ennemmin kahden erilaisen arvomaailman yhteensovittamisena, kuin kahden erilaisen prosessin liittämisenä yhteen, koska teoreettisia prosesseja harvoin noudatetaan mekaanisesti käytännössä. Monet haasteet ketterän ohjelmistokehityksen ja käyttäjälähtöisen suunnittelun yhdistämisessä selittyvät ketterässä manifestissa määritellyillä arvoilla, joiden ohjaamana ketterää ohjelmistokehitystä tehdään. Näistä arvoista puuttuu riittävä painotus käyttäjälähtöisyyttä kohtaan. Hitaiden ja tutkimusorientoituneiden käyttäjälähtöisten suunnittelumenetelmien yhdistäminen ketteriin prosesseihin on toinen merkittävä haaste. Ketterän kehityksen käyttäjälähtöisyyteen vaikuttaa lopulta eniten se, miten käyttäjälähtöisyyteen liittyvää työtä tuetaan tai vaikeutetaan organisaatiossa, jossa kehitystyö tapahtuu. Tutkielmassa havaittiin, että organisaatioissa ei aina panosteta riittävästi käyttäjälähtöisiin aktiviteetteihin. Lisäksi kehittäjien ja suunnittelijoiden välinen yhteistyö saatetaan organisoida epäedullisella tavalla.

Ketterä ohjelmistokehitys on todettu monissa tapauksissa toimivaksi ja se on vakiinnuttanut asemansa ohjelmistotuotannossa, joten voi olla mielekkäämpää pyrkiä muokkaamaan ketteriä menetelmiä käyttäjälähtöisempään suuntaan sen sijaan, että kehitettäisiin jotain kokonaan uutta. Ketteriä menetelmiä voidaan tehdä käyttäjälähtöisemmiksi muokkaamalla niiden taustalla vaikuttavia arvoja tukemaan paremmin käyttäjälähtöisyyttä. Käyttäjälähtöisistä suunnittelumenetelmistä parhaiten ketteriin menetelmiin istuvia ovat epämuodolliset, nopeat

ja ”ketterät” menetelmät. Yhtenä ratkaisuna tutkielmassa esitettiin käyttäjälähtöisten prosessien standardien madaltaminen, jotta ne istuisivat paremmin nopeisiin ja ketteriin prosesseihin. Organisaatiot voivat tukea käyttäjäkokemuksen suunnittelijoiden ja kehittäjien välistä yhteistyötä liittämällä käyttäjäkokemuksen suunnittelijat vahvemmin osaksi ketterää kehitystiimiä ja mahdollistamalla tämän tiimin itseorganisoitumisen ketterien arvojen mukaisesti. Käyttäjäkokemuksen arvon parempi ymmärrys voi auttaa perustelemaan paremmin siihen tehtäviä panostuksia. Tuloksia tarkasteltaessa tulee kuitenkin huomioida, että ehdotukset ovat hyvin yleisluontoisia, eikä niistä voida vetää tarkkoja johtopäätöksiä siitä, mikä on paras tapa parantaa ketterien prosessien käyttäjälähtöisyyttä tietyssä kontekstissa. Käsitellyssä aineistossa oli aiheen subjektiivisuuteen nähden verrattain vähän empiiristä tutkimusta.

Käyttäjälähtöisen suunnittelun ja ketterän ohjelmistokehityksen on todettu myös jakavan monia piirteitä, kuten ihmiskeskeisyys, palautteen korostaminen kehitystä ohjaavana tekijänä ja iteratiivisuus. Näin voidaan olettaa, että näiden kahden lähestymistavan integrointi on haasteistaan huolimatta mahdollista. Yhteistoimintapotentiaali on havaittu myös tarkastellussa kirjallisuudessa. Tehdyn tutkimuksen perusteella kummankin lähestymistavan arvomaailman ja käytänteiden täytyy muuttua, jotta integraatio toimisi paremmin. Käyttäjälähtöisistä toimintatavoista ei ole hyötyä, jos ohjelmistokehitysprojekti epäonnistuu teknisten tai taloudellisten rajoitteiden vuoksi. Toisaalta käytettävyydeltään huonoa ohjelmistoa ei voida myöskään sanoa onnistuneeksi, vaikka tekniset vaatimukset oltaisiinkin täytetty annetussa budjetissa. Ohjelmistokehitysprojekteissa joudutaan rajallisten resurssien vuoksi aina tekemään kompromisseja, jotka vaikuttavat lopputulokseen. Jatkotutkimuksena olisi mielenkiintoista tarkastella, minkälaisia kompromisseja käyttäjäkokemuksen ja teknisten ominaisuuksien suhteen tehdään ja missä määrin nämä kompromissit ovat onnistuneita lopputuloksen kannalta. Tuotteen arvon kannalta sen käyttäjäkokemus on usein hyvin olennainen tekijä, joten toinen mielenkiintoinen kysymys on, miksi käyttäjäkokemukseen liittyviä tehtäviä priorisoidaan heikosti, kuten tässä tutkimuksessa havaittiin. Yleisesti ottaen näin subjektiivisen aiheen ymmärtämiseksi tarvitaan erityisesti empiiristä jatkotutkimusta aiheesta. Tämä tarve oli havaittu myös tutkielmassa käsitellyssä kirjallisuudessa.

LÄHTEET

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods-Review and analysis. Finland: VTT Publications.

Agile Alliance (2001). Manifesto for Agile Software Development. Haettu 18.4.2019 osoitteesta http://www.agilemanifesto.org

Anwer, F., Aftab, S., Shah, S. M., & Waheed, U. (2017). Comparative Analysis of Two Popular Agile Process Models: Extreme Programming and Scrum. International Journal of Computer Science and Telecommunications, 8(2), 1-7.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., & Kern, J. (2001). Manifesto for agile software development.

Blomkvist, S. (2006). User-centred design and agile development of IT systems. Uppsala: Deptartment of Information Technology Uppsala University Uppsala.

Cohen, D., Lindvall, M., & Costa, P. (2004). An introduction to agile methods. Advances in computers, 62(03), 1-66.

Constantine, L. & Lockwood, L. (May 3, 2003). Usage-centered software engineering. pp.746-747. IEEE Computer Society.

Ferreira, J. (2012, June). Agile development and UX design: towards understanding work cultures to support integration. In International Conference on Advanced Information Systems Engineering (pp. 608-615).

Ferreira, J., Sharp, H. & Robinson, H. (2011). User experience design and agile development: Managing cooperation through articulation work. Software: Practice and Experience, 41(9), 963-974.

Forlizzi, J. & Battarbee, K. (Aug 1, 2004). Understanding experience in interactive systems. pp.261-268. ACM.

Garrett, J. J. (2010). Elements of user experience, the: user-centered design for the web and beyond. Pearson Education.

Gray, C. M. (2016, May). It's more of a mindset than a method: UX practitioners' conception of design methods. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (pp. 4044-4055). ACM.

Gulliksen, J., Göransson, B., Boivie, I., Blomkvist, S., Persson, J. & Cajander, Å. (2003). Key principles for user-centred systems design. Behaviour & Information Technology, 22(6), 397-409.

Hassenzahl, M. & Tractinsky, N. (2006) User experience - a research agenda, Behaviour & Information Technology, 25:2, 91-97

International Organization for Standardization (2018). Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts (ISO Standard No. 9241). Haettu 18.4.2019 osoitteesta https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en

Jia, Y., Larusdottir, M. & Cajander, Å. (2012). The usage of usability techniques in scrum projects. Human-centered software engineering (s. 331-341). Berlin, Heidelberg: Springer Berlin Heidelberg.

Jurca, G., Hellmann, T. D. & Maurer, F. (Jul 2014). Integrating agile and user-centered design: A systematic mapping and review of evaluation and validation studies of agile-UX. pp.24-32. IEEE.

Kuusinen, K. (2014). Improving UX work in scrum development: A three-year follow-up study in a company. pp.259-266. Springer.

Larusdottir, M., Gulliksen, J. & Cajander, Å. (2017). A license to kill – improving UCSD in agile development. The Journal of Systems & Software, 123, 214-222.

Lárusdóttir, M. K., Cajander, Å., & Gulliksen, J. (2012). The big picture of UX is missing in Scrum projects. In Proceedings of the International Workshop on the Interplay between User Experience (UX) Evaluation and System Development (I-UxSED) (pp. 49-54).

Law, E., Roto, V., Hassenzahl, M., Vermeeren, A. & Kort, J. (Apr 4, 2009). Understanding, scoping and defining user experience. pp.719-728. ACM.

Li Jiang & Eberlein, A. (Oct 2009). An analysis of the history of classical software development and agile development. pp.3733-3738. IEEE.

Memmel, T., Gundelsweiler, F., & Reiterer, H. (2007, September). Agile human-centered software engineering. In Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI... but not as we know it-Volume 1 (pp. 167-175).

Ming Huo, Verner, J., Liming Zhu & Babar, M. A. (2004). Software quality and agile methods. pp.525 vol.1. IEEE.

Moczarny, I. M., De Villiers, M. R., & Van Biljon, J. A. (2012, October). How can usability contribute to user experience?: a study in the domain of e-commerce. In Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference (pp. 216-225). ACM.

Nielsen, J. (2012, January 4). Usability 101: Introduction to Usability. Haettu 18.4.2019 osoitteesta https://www.nngroup.com/articles/usability-101-introduction-to-usability/

Schwaber, K. (1997). Scrum development process. In Business object design and implementation (pp. 117-134). Springer, London.

Schwaber, K., & Beedle, M. (2002). Agile software development with Scrum (Vol. 1). Upper Saddle River: Prentice Hall.

Silva da Silva, T., Martin, A., Maurer, F. & Silveira, M. (Aug 2011). User-centered design and agile methods: A systematic review. pp.77-86. IEEE.

Silva da Silva, T., Selbach Silveira, M., Maurer, F. & Hellmann, T. (2012). User experience design and agile development: From theory to practice. Journal of Software Engineering and Applications, 5(10), 743-751.

Singh, M. (Aug 2008). U-SCRUM: An agile methodology for promoting usability. pp.555-560. IEEE.

Väänänen-Vainio-Mattila, K., Roto, V., & Hassenzahl, M. (2008). Towards practical user experience evaluation methods. Meaningful measures: Valid useful user experience measurement (VUUM), 19-22.

Øvad, T. & Larsen, L. B. (Aug 2015). The prevalence of UX design in agile development processes in industry. pp.40-49. IEEE.