Petri Mattila

KÄYTTÄJÄKESKEISEN SUUNNITTELUN INTEGROINTI KETTERÄN KEHITTÄMISEN PROSESSIIN JA ROOLEIHIN

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
2014

TIIVISTELMÄ

Mattila, Petri
Käyttäjäkeskeisen suunnittelun integrointi ketterän kehittämisen prosessiin ja rooleihin
Jyväskylä: Jyväskylän yliopisto, 2014, 28 s.
Tietojärjestelmätiede, kandidaatin tutkielma
Ohjaaja: Leppänen, Mauri

Viime vuosikymmenen aikana ohjelmistokehityksessä on tapahtunut suuria edistysaskeleita. Erityisesti ketterät kehittämismenetelmät ovat kasvattaneet suosiotaan. Yksi tärkeimmistä ketterien menetelmien periaatteista on asiakkaan tarpeiden tyydyttäminen. Lisäksi tuotteen käyttöliittymän tulee olla helposti käytettävä. On kuitenkin todettu, että ketterät menetelmät eivät välttämättä itsessään kykene varmistamaan ohjelmistojen käytettävyyttä. Alan kirjallisuudessa on tutkittu tapoja, joilla voitaisiin yhdistää ketterän kehittämisen ja käyttäjäkeskeisen suunnittelun parhaimpia puolia. Tämän tutkielman tarkoituksena on muodostaa lukijalle käsitys ketterästä kehittämisestä, käyttäjäkeskeisestä suunnittelusta sekä niiden integroimisesta. Työssä keskitytään yleisimpiin ketteriin menetelmiin, Scrumiin ja XP:hen, sekä niiden prosesseihin ja rooleihin. Tutkielman tuloksia voidaan hyödyntää sellaisissa ohjelmistoprojekteissa, joissa ketterien menetelmien lisäksi halutaan korostaa käytettävyyden näkökulmaa.

Asiasanat: ketterä ohjelmistokehitys, XP, Scrum, käyttäjäkeskeinen suunnittelu

ABSTRACT

Mattila, Petri
Integrating User-Centered Design with the Processes and Roles of Agile Software Development
Jyväskylä: University of Jyväskylä, 2014, 28 p.
Information Systems, Bachelor's Thesis
Supervisor: Leppänen, Mauri

Software development has made great strides in the past decade. Agile software development, in particular, has become more popular. One of the most important principles of agile methodology is to satisfy the customer's needs. In addition, the product should have a usable user interface. However, there has been evidence that agile methods alone cannot always ensure the usability of a user interface. In the literature, ways of combining best practices from agile development and user-centered design have been studied. The purpose of this thesis is to introduce the reader to agile development, user-centered design as well as the integration of these methods. It focuses on the processes and roles of the most widely used agile methods – Scrum and XP. The results of this study can be used in agile software projects in which promoting the usability aspect is desired.

Keywords: agile software development, XP, Scrum, user-centered design

KUVIOT

SISÄLLYS

1 JOHDANTO

Ohjelmistokehitys on viimeisen vuosikymmenen aikana kokenut merkittävän paradigmamuutoksen. Vuosituhannen vaihteen jälkeen ketterät menetelmät ovat kasvattaneet suosiotaan ja siten vaikuttaneet ohjelmistoalan ajattelutapoihin (Beck ym., 2001a; Ambler, 2005). Viimeaikoina yhä useampi yritys soveltaa ketteriä menetelmiä (Silva da Silva, 2012; VersionOne, 2013; Rodriguez ym. 2012). Ketterälle ohjelmistokehitykselle ovat ominaisia lyhytkestoiset kehityssyklit sekä kyky reagoida muutoksiin nopeasti. Lisäksi ketterät menetelmät pyrkivät tuottamaan valmiita ohjelmaosia mahdollisimman tehokkaasti (Huo, Verner, Zhu & Babar, 2004). Ne korostavat myös toimivan ohjelmiston ensisijaisuutta ja aktiivista yhteydenpitoa asiakkaan kanssa (Beck ym., 2001a). Ketteristä menetelmistä yleisimmät ovat Scrum ja Extreme Programming (XP) (Silva da Silva, 2012; VersionOne, 2013).

Käytettävyys on keskeinen ohjelmiston laatuominaisuus. Nielsenin (1993) mukaan se koostuu viidestä ominaisuudesta, joita ovat opittavuus, tehokkuus, muistettavuus, virheet ja tyytyväisyys. Hyvän käytettävyyden takaamiseksi on kehitetty käyttäjäkeskeisen suunnittelun lähestymistapa (ISO 13407:1999). Se perustuu kahteentoista perusperiaatteeseen, joiden tarkoituksena on tukea kehittämisprosessia sekä helpottaa kommunikaatiota ja käyttäjäkeskeisen suunnitteluprosessin arviointia (Gulliksen ym., 2003).

Ketterän kehittämisen manifestissa ei oteta kantaa tuotteen käytettävyyteen (Beck ym., 2001a). Sen periaatteiden keskiössä ovat ohjelmiston toiminnalliset ominaisuudet (Beck ym., 2001b). Pattonin (2002) mukaan ketterät menetelmät eivät aina yksin kykene varmistamaan ohjelmistojen hyvää käytettävyyttä. Tämän vuoksi onkin ollut tarpeellista kehittää uusia lähestymistapoja käytettävyyden takaamiseksi. Alan kirjallisuudesta (esim. Sy, 2007; Ungar & White, 2008; Singh, 2008) löytyykin tutkimuksia, joissa ketterien kehittämismenetelmien yhteyteen on liitetty käytettävyyssuunnittelun piirteitä. Ehdotukset poikkeavat jonkin verran toisistaan sen mukaan, miten integrointi on tehty ketterään kehittämisprosessiin ja rooleihin.

Tämän tutkielman tarkoituksena on muodostaa lukijalle käsitys ketterästä kehittämisestä, käyttäjäkeskeisestä suunnittelusta sekä näiden kahden yhdistä-

misestä ohjelmistokehitysprojektissa. Työssä esitellään tutkimuksia, joissa käyttäjäkeskeisen suunnittelun piirteitä on sovellettu ketterässä ympäristössä. Lisäksi tutkielmassa esitellään ehdotettuja ratkaisuja menetelmien integroimiseksi. Tutkielma pohjautuu alan tieteelliseen kirjallisuuteen.

Tutkielman tutkimusongelmana on selvittää, *kuinka käyttäjäkeskeistä suunnittelua voidaan integroida ketteriin kehittämismenetelmiin*. Ketterien menetelmien joukosta tutkittavaksi on valittu Scrum ja XP. Tarkastelun kohteeksi on rajattu edellä mainittujen menetelmien prosessit ja roolit. Tutkimuskysymyksiä muodostui kaksi:

1. Miten käyttäjäkeskeisen suunnittelun piirteitä voidaan liittää ketterän kehittämisen prosessiin ja rooleihin?
2. Millaisia haasteita esiintyy, kun käyttäjäkeskeinen suunnittelu integroidaan ketterään kehittämiseen?

Tutkielma on jäsennetty viiteen lukuun. Toisessa luvussa kerrotaan taustaa ketterästä ohjelmistokehityksestä ja sen periaatteista sekä esitellään ketteristä menetelmistä Scrum ja XP. Luvun pääfokus on ketterän ohjelmistokehityksen prosessissa ja rooleissa. Kolmannessa luvussa määritellään käytettävyyden käsite sekä esitellään käyttäjäkeskeisen suunnittelun periaatteet ja prosessi. Neljännessä luvussa esitellään tutkimuksia, joissa ketterien menetelmien yhteyteen on integroitu käyttäjäkeskeisen suunnittelun piirteitä. Lisäksi luvussa tehdään yhteenvetoa tutkimustuloksista. Tutkielman viimeinen luku on yhteenveto, jossa kerrataan tutkielman keskeisimmät havainnot ja pohditaan mahdollisia jatkotutkimusaiheita.

2 KETTERÄ OHJELMISTOKEHITYS

Tässä luvussa kerrotaan ensin yleisellä tasolla ketterästä ohjelmistokehityksestä ja sen periaatteista. Ketteristä menetelmistä esitellään Scrum ja XP. Luvussa keskitytään enimmäkseen ketterän ohjelmistokehityksen prosessiin ja rooleihin.

2.1 Taustaa ketteristä menetelmistä

Ketterällä ohjelmistokehittämisellä tarkoitetaan ohjelmistotuotannon menetelmiä, joille on yhteistä lyhytkestoiset kehityssyklit, nopea muutoksiin reagoiminen ja pyrkimys tuottaa valmiita toiminnallisuuksia mahdollisimman nopeasti (Huo ym., 2004). Ketterät menetelmät korostavat myös toimivan ohjelmiston ensisijaisuutta sekä aktiivista keskustelua asiakkaan kanssa. Ketterä lähestymistapa (agile approach) sai virallisesti alkunsa vuonna 2001, kun ryhmä ohjelmistokehittäjiä muodostivat Agile Alliance –nimisen järjestön, jonka tarkoituksena oli luoda ketterälle kehittämiselle yhteinen ajattelumalli, joka vastaisi perinteisen ohjelmistokehityksen, kuten vesiputousmallin, ongelmakohtiin (Beck ym., 2001a). Ketterät kehittämismenetelmät eroavat vesiputousmallin ta ”raskaammista” prosessikeskeisistä menetelmistä esimerkiksi siten, että ne keskittyvät enemmän arvoihin ja periaatteisiin kuin prosesseihin (Hussain ym., 2012). Agile Alliance -järjestön perustamisen tuloksena syntyikin ketterän ohjelmistokehityksen manifesti (Agile Manifesto), joka perustuu neljään arvoon ja kahteentoista periaatteeseen. Arvot on ilmaistu seuraavalla tavalla (Beck ym., 2001a):

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

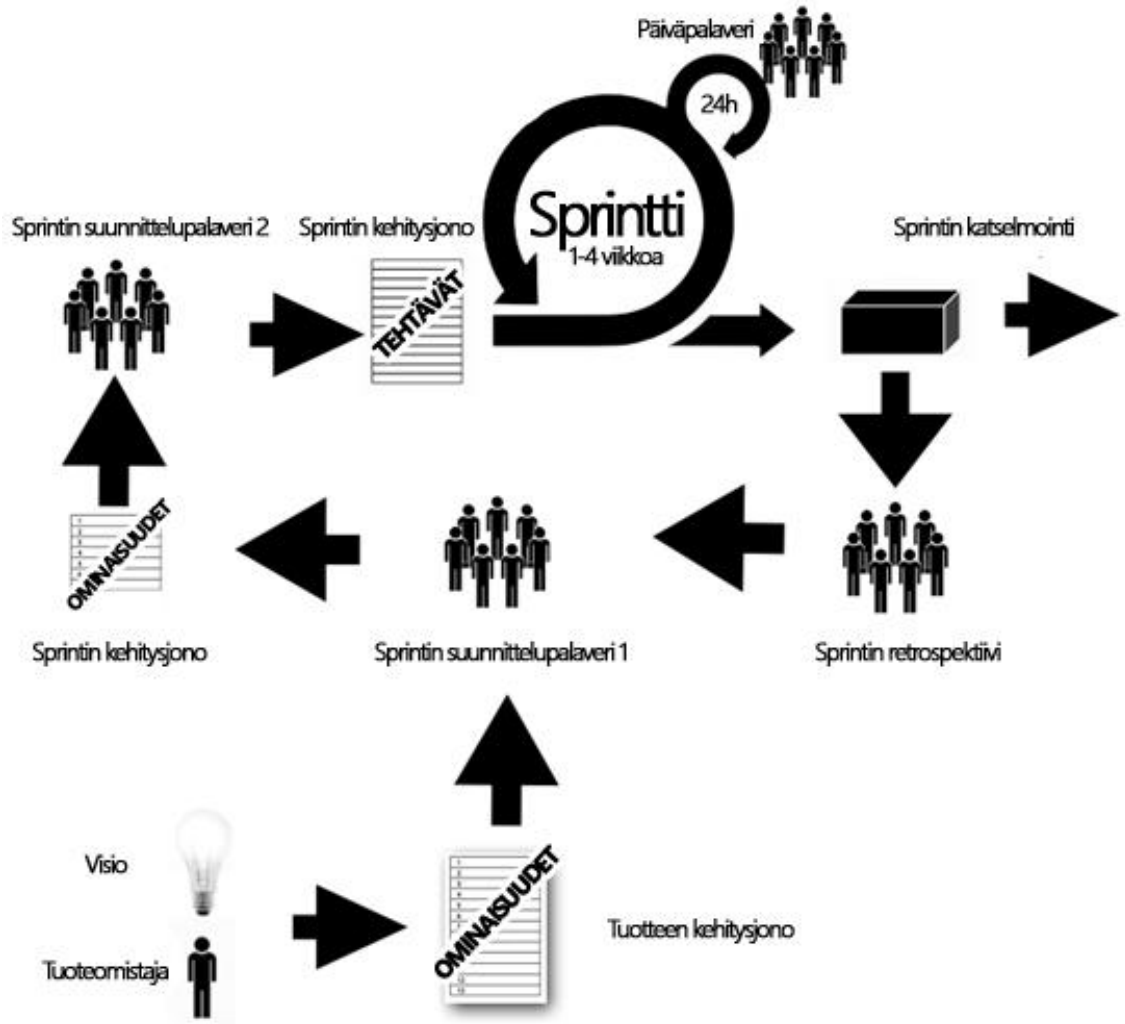
Ketterän ohjelmistokehityksen manifesti loi pohjan nykyään tunnetulle ketterälle ohjelmistokehitykselle (Huo ym., 2004). Mikäli ohjelmistokehitysmenetelmä toteuttaa manifestin arvoja ja periaatteita, voidaan se lukea kuuluvaksi ketteriin menetelmiin (Ambler, 2005). Lukuisista ketteristä menetelmistä yleisimmät ovat Scrum ja XP (Silva da Silva, 2012; VersionOne, 2013). Seuraavassa kuvataan lyhyesti niitä molempia.

2.2 Scrum

Scrum on empiiriseen prosessinhallintaan perustuva, monimutkaisten tuotteiden kehittämiseen ja ylläpitoon tarkoitettu viitekehys, jonka sisällä voidaan käyttää useita prosesseja ja tekniikoita. Se on kevyt ja helppo ymmärtää, mutta vaikea hallita. (Schwaber & Sutherland, 2013.) Scrum soveltuu sellaisiin projekteihin, joissa vaatimukset muuttuvat usein. Vaikka menetelmää käytetäänkin eniten ohjelmistokehityksessä, voidaan sen metodologiaa hyödyntää myös muissa erilaisissa projekteissa. Scrumissa kehitysprosessi tapahtuu iteratiivisesti sprinteissä, jotka kestävät yhdestä neljään viikkoon. Jokainen sprintti alkaa suunnittelupalaverilla ja päättyy katselmointiin. (Cohn, 2011.) Scrum koostuu Scrum-tiimeistä ja niiden rooleista sekä tapahtumista, tuotoksista ja säännöistä. (Schwaber & Sutherland, 2013). Seuraavaksi kuvataan prosessia ja rooleja.

2.2.1 Prosessi

Scrum-projektissa toteutetaan tuoteomistajan (product owner) laatimaa visiota projektista. Vision perusteella luodaan tuotteen kehitysjono (product backlog), joka sisältää listan vaadituista toiminnallisuuksista tärkeysjärjestyksessä. Tärkeysjärjestys muodostuu siten, että sellaiset toiminnallisuudet, jotka tuottavat eniten arvoa asiakkaalle tai liiketoiminnalle, toteutetaan ensin ja ovat siten listan alussa. Kehitysjono muuttuu ja kehittyy projektin edetessä. (Sutherland, 2010.) Scrumissa kaikki tapahtumat ovat ennalta suunniteltuja ja aikarajattuja projektin säännöllisyyden takaamiseksi (Schwaber & Sutherland, 2013). Kuviossa 1 on esitetty Scrumin prosessin vaiheet:



KUVIO 1 Scrumin prosessimalli (Sutherland, 2010)

Scrumin mukainen ohjelmistokehitys perustuu tyypillisesti 1-4 viikon mittaisiin työsykleihin, joita kutsutaan sprinteiksi (Sutherland, 2010). Sprintin aikana tuotetaan valmis, vaatimukset täyttävä ja julkaisukelpoinen versio tuotteesta (Schwaber & Sutherland, 2013). Sprintillä on aina tarkka kesto ja päättymispäivämäärä, jota ei voida ylittää edes silloin, vaikka työ olisi vielä keskeneräinen (Sutherland, 2010). Sprintit sisältävät sprintin suunnittelupalaverin, päiväpalaverin, itse kehitystyön, sprintin katselmoinnin sekä sprintin retrospektiivin (Schwaber & Sutherland, 2013).

Jokaisen sprintin alussa pidetään kaksiosainen sprintin suunnittelupalaveri, jossa Scrum-tiimin kesken laaditaan sprintin tavoitteet ja suunnitellaan sprintin aikana tehtävä työ (Schwaber & Sutherland, 2013). Palaverin alussa Scrum-tiimi valitsee tuotteen kehitysjonosta toteutettavaksi ne tehtävät, jotka he pystyvät saamaan valmiiksi sprintin aikana. Palaverin loppupuolella suunnitellaan yksityiskohtaisesti, kuinka valitut tehtävät aiotaan konkreettisesti toteuttaa. Kehitysjonosta valitut ominaisuudet pilkotaan pienemmäksi joukoksi tehtäviä ja ne kirjataan sprintin kehitysjonoon (sprint backlog). (Sutherland, 2010.)

Sprintin aikana pidetään kerran päivässä lyhyt, maksimissaan 15 minuutin mittainen kokous, jota kutsutaan päiväpalaveriksi (Daily Scrum). Siinä kehitystiimi tarkastelee, mitä edellisen palaverin jälkeen on saatu aikaiseksi, ja toisaalta sitä, mitä asioita tiimi pystyy toteuttamaan seuraavaan päiväpalaveriin mennessä. Päiväpalaverin ansiosta voidaan seurata työn edistymistä ja sprintin kehitysjonon toteutumista sekä taata se, että kehitystiimi pääsee sprintin alussa asetettuihin tavoitteisiin. (Schwaber & Sutherland, 2013.)

Sprintin päätyttyä Scrum-tiimi ja sidosryhmät kokoontuvat sprintin katselmointiin (sprint review), jossa keskustellaan siitä, mitä sprintin aikana saatiin tehdyksi ja mitä voitaisiin kehittää seuraavaksi (Sutherland, 2010). Katselmointi on epämuodollinen palaveri, jonka tavoitteena on saada palautetta ja edistää sidosryhmien välistä kommunikointia. Tuloksena on tarkistettu tuotteen kehitysjono, joka toimii pohjana seuraaville sprintin suunnittelupalavereille. (Schwaber & Sutherland, 2013.)

Sprintin katselmoinnin jälkeen on vielä jäljellä sprintin retrospektiivi (sprint retrospective), jossa Scrum-tiimillä on mahdollisuus tarkastella ja keskustella edellisen sprintin aikana havaituista asioista (Sutherland, 2010). Tällaisia asioita voivat olla esimerkiksi sprintissä hyvin sujuneet asiat sekä parannusehdotukset seuraavaa sprinttiä ajatellen. Retrospektiivin loppuun mennessä Scrum-tiimi on tunnistanut kehitysprosessin parannustarpeet ja luonut suunnitelman niiden toteuttamiseksi seuraavan sprintin aikana. (Schwaber & Sutherland, 2013.)

2.2.2 Roolit

Scrum-tiimit ovat itseohjautuvia tiimejä, jossa ryhmän jäsenet itse päättävät heille parhaiten soveltuvat työtavat tehtävien suorittamiseksi. Vaadittavat tehtävät ja toiminnallisuudet tuotetaan iteratiivisesti ja inkrementaalisesti. Lisäksi ryhmän jäsenet ovat moniosaajia, jotta työ voidaan suorittaa ryhmän sisällä riippumatta ulkopuolisista tekijöistä tai henkilöistä. Nämä seikat takaavat tiimille parhaan joustavuuden, tuottavuuden ja luovuuden. Scrum-tiimi koostuu tuoteomistajasta, kehitystiimistä sekä Scrum-mestarista. (Schwaber & Sutherland, 2013.)

Tuoteomistaja on Scrum-tiimin avainhenkilö ja kehitysprojektin sidosryhmien edustaja. Hänen vastuullaan on projektin vision välittäminen kehitystiimin jäsenille (Cohn, 2011). Lisäksi tuoteomistaja vastaa kehitettävän tuotteen arvon ja kehitystiimin työn laadun maksimoimisesta. Hän on myös ainoa henkilö, joka on vastuussa tuotteen kehitysjonon hallinnasta. (Schwaber & Sutherland, 2013.)

Kehitystiimin jäsenet ovat ammattilaisia, jotka työskentelevät yhdessä saavuttaakseen etukäteen sovitut tavoitteet sprintin aikana. Mikään muu taho kehitystiimin lisäksi ei osallistu tuoteversion kehitykseen. Scrumissa kaikki kehitystiimin jäsenet ovat poikkeuksetta titteliltään kehittäjiä. (Schwaber & Sutherland, 2013.)

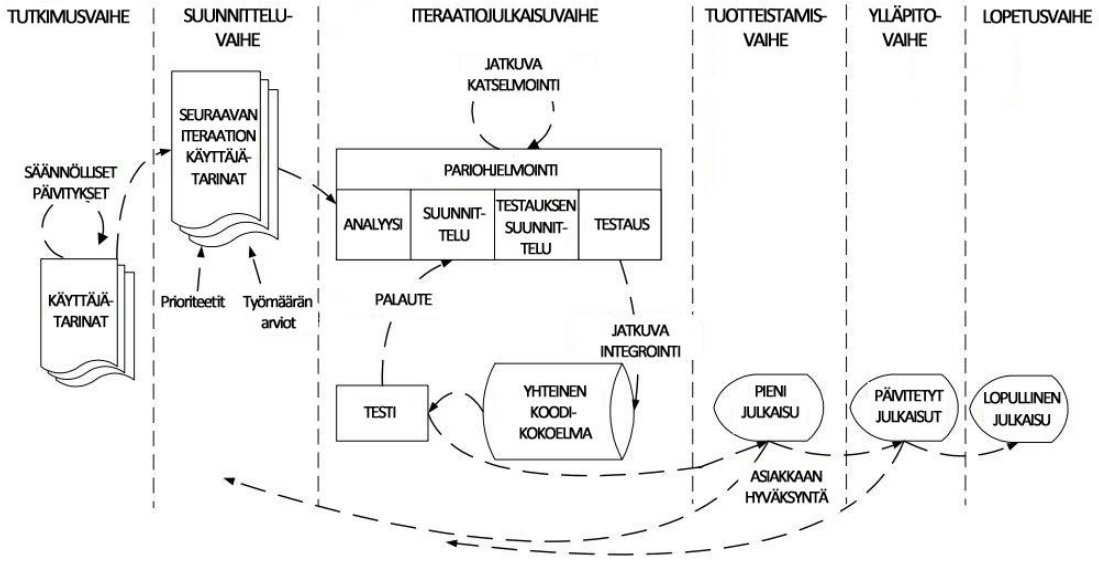
Scrum-tiimin kolmas osapuoli on Scrum-mestari, jota usein pidetään tiimin ”valmentajana”. Hänen tehtävänsä on varmistaa, että tiimi työskentelee mahdollisimman tehokkaasti. (Cohn, 2011.) Lisäksi hän on vastuussa siitä, että kaikki ryhmän jäsenet ymmärtävät ja noudattavat Scrumin sääntöjä ja periaatteita (Schwaber & Sutherland, 2013).

2.3 Extreme Programming (XP)

Extreme Programming eli XP on yksi tunnetuimmista ketteristä kehitysmenetelmistä (Hussain ym., 2012). Kent Beckin, Ward Cunninghamin ja Ronn Jeffriesin esittämän XP-ajattelumallin lähtökohtana oli selvittää, mikä tekee ohjelmiston toteuttamisesta helppoa ja mikä tekee siitä hankalaa. Se keskittyy ohjelmointitekniikoiden vahvaan soveltamiseen, selkeään viestintään sekä tiimityöskentelyyn. Alun perin XP oli pienille ja keskisuurille tiimeille kevyt tapa kehittää ohjelmistoja silloin, kun vaatimukset muuttuvat nopeasti tai ne ovat epämääräisiä. Nykyään XP:tä voidaan kuitenkin käyttää kaiken kokoisissa tiimeissä. Se sisältää viisi perusarvoa kehityksen tukemiseksi, jotka ovat kommunikaatio, yksinkertaisuus, palaute, rohkeus ja kunnioitus. (Beck & Andres, 2004a.) Seuraavassa kuvataan XP:n prosessi ja roolit.

2.3.1 Prosessi

XP:n prosessin elinkaari koostuu viidestä eri vaiheesta, jotka ovat tutkimus, suunnittelu, iteraatiojulkaisu, tuotteistaminen, ylläpito ja lopetus (Abrahamsson, Salo, Ronkainen & Warsta, 2002). Kuvio 2 esittää prosessin vaiheet:



KUVIO 2 XP:n prosessin elinkaari vaiheittain (Abrahamsson ym., 2002)

Tutkimusvaiheessa projektitiimi tutustuu projektissa tarvittaviin työkaluihin, teknologioihin ja käytäntöihin. Tiimi rakentaa prototyypin kehitettävästä järjestelmästä testatakseen käytössä olevia teknologioita sekä arkkitehtuurisia mahdollisuuksia. Samaan aikaan asiakkaat kirjaavat käyttäjätarinoita eli asioita, joita he toivovat toteutettavaksi ensimmäisessä julkaisussa. Tämä vaihe voi kestää muutamasta viikosta pariin kuukauteen. (Beck, 1999.)

Suunnitteluvaiheessa arvioidaan jokainen käyttäjätarina työmäärän kannalta, jonka jälkeen tarinat järjestetään tärkeysjärjestykseen. Samalla sovitaan projektin aikataulusta ja ensimmäisen julkaisun sisällöstä. Vaihe kestää yleensä muutaman päivän. (Beck, 1999.)

Iteraatiojulkaisussa projektin aikataulu pilkotaan iteraatioihin. Jokaisen tällaisen iteraation toteuttaminen kestää yhdestä neljään viikkoa. Ensimmäisessä iteraatiossa suunnitellaan arkkitehtuuri järjestelmälle. Asiakkaan päätettävänä on, mitkä toiminnallisuudet toteutetaan missäkin iteraatiossa. Jokaisen iteraation päätteeksi testataan asiakkaan toimesta tuotteen toiminnallisuus. Viimeisen iteraation lopussa järjestelmä on valmis tuotantoon. (Beck, 1999.)

Tuotteistamisvaiheessa tutkitaan järjestelmän suorituskykyä ja suoritetaan lisätestausta ennen kuin järjestelmä voidaan julkaista. Tässä vaiheessa iteraatioiden kestoa saatetaan joutua lyhentämään. Tuotteistamisessa on mahdollista, että järjestelmässä havaitaan vielä uusia muutoksia. Tällöin on päätettävä, sisällytetäänkö ne nykyiseen julkaisuun. Järjestelmään liittyvät ehdotukset ja esitetyt ideat dokumentoidaan, jotta ne voidaan myöhemmin toteuttaa esimerkiksi ylläpitovaiheessa. (Beck, 1999.)

Ylläpitovaiheessa järjestelmä on jo tuotteistettu. XP-projektin on pidettävä tuotanto käynnissä ja samanaikaisesti tuotettava uusia iteraatioita. Tässä vaiheessa kehittämisen tahti yleensä hidastuu johtuen muista asiakastuen tehtävistä. (Beck, 1999.)

Lopetusvaihe alkaa, kun asiakkaalla ei ole enää toteutettavia käyttäjätarinoita ja kun järjestelmä täyttää muutenkin asiakkaan vaatimukset, esimerkiksi järjestelmän suorituskyvyn tai toimintavarmuuden suhteen. Lopetusvaiheessa järjestelmän arkkitehtuuriin, suunnitelmiin tai ohjelmakoodiin ei enää tehdä muutoksia. Tämä mahdollistaa lopullisen järjestelmän dokumentaation kirjoittamisen. (Beck, 1999.)

2.3.2 Roolit

XP:ssä on omat roolinsa eri tehtäville kehitysprosessin aikana (Abrahamsson ym., 2002). Nämä roolit ovat ohjelmoija, käyttäjä, testaaja, projektipäällikkö, tuotepäällikkö, johtaja, arkkitehti, interaktiosuunnittelija sekä tekninen kirjoittaja. Beck ja Andres (2004) määrittelevät XP:n roolit seuraavasti:

Ohjelmoijan tehtävänä on arvioida ja jakaa käyttäjätarinat yksittäisiin tehtäviin, kirjoittaa testejä ja ohjelmakoodia, automatisoida kehitysprosessia sekä asteittain parantaa järjestelmän suunnitelmia.

XP-tiimin käyttäjät auttavat kirjoittamaan ja valitsemaan käyttäjäkertomuksia sekä priorisoimaan ne kehityksen aikana. Lisäksi he kirjoittavat testejä sekä päättävät, milloin kukin vaatimus on täyttynyt.

Testaaja auttaa asiakkaita valitsemaan ja kirjoittamaan testejä. Lisäksi testaaja ajaa testejä säännöllisesti, julkaisee testituloksia ja ylläpitää testaustyökaluja.

Projektipäällikkö helpottaa tiimin sisäistä viestintää sekä koordinoi kommunikaatiota asiakkaiden, toimittajien ja muun muiden organisaation osapuolten välillä. Projektipäällikkö on vastuussa tiiminsä edistymisestä ja suunnitelmien toteutumisesta projektissa.

Tuotepäällikön tehtävänä on kirjoittaa käyttäjäkertomuksia, valita teemoja ja kertomuksia neljännesvuosittaisiin sykleihin, valita tarinoita viikoittaisiin sykleihin ja vastata projektin aikana ilmeneviin kysymyksiin. Hänen tehtäväänään on myös käyttäjien ja ohjelmoijien välisen kommunikaation tukeminen sekä auttaa tiimiä prioriteettien valinnoissa.

Johtaja toimii tiimin motivoijana, rohkaisijana ja luottohenkilönä. Lisäksi hän valvoo, että tiimin toiminta ja koko prosessi kehittyvät jatkuvasti. Johtaja odottaa tiimiltään rehellisyyttä ja selkeitä vastauksia kaikissa päätöksentekoon liittyvissä asioissa.

Arkkitehti suorittaa suuren mittakaavan refaktorointia, kirjoittaa järjestelmätason arkkitehtuuria koskevia testejä sekä toteuttaa käyttäjätarinoita. Hänen tehtävänään on myös jakaa järjestelmä pienempiin osiin osajärjestelmiksi.

Interaktiosuunnittelija valitsee järjestelmän yleiset metaforat ja arvioi käyttöönotetun järjestelmän käyttöä uusien tarinoiden löytämiseksi. Lisäksi hän työskentelee asiakkaiden kanssa auttaen heitä kirjoittamaan ja selventämään käyttäjätarinoita.

Tekninen kirjoittaja antaa palautetta järjestelmän ominaisuuksista aikaisessa vaiheessa. Hänen tehtävänään on myös luoda läheiset välit käyttäjien kanssa esimerkiksi kuuntelemalla heidän antamaansa palautetta tuotteesta. Lisäksi tekninen kirjoittaja on vastuussa projektin dokumentoinnista.

2.4 Yhteenveto

Tässä luvussa käsiteltiin ketterän ohjelmistokehityksen taustaa ja esiteltiin ketteristä menetelmistä Scrum ja XP. Molemmat menetelmät esiteltiin niiden prosessien ja roolien näkökulmasta. Scrumille ja XP:lle on yhteistä iteratiivinen ja inkrementaalinen kehitysprosessi. Scrumin prosessi perustuu sprintteihin kutsuttuihin työsykleihin. XP:n kehitysprosessi on sen sijaan viisivaiheinen. Scrum sisältää kolme pääroolia, kun taas XP:ssä eri rooleja on yhdeksän.

3 KÄYTTÄJÄKESKEINEN SUUNNITTELU

Käyttäjäkeskeinen suunnittelu (User-Centered Design, UCD) on järjestelmäkehityksessä käytettävä lähestymistapa, jonka tavoitteena on tuottaa mahdollisimman käytettäviä vuorovaikutteisia järjestelmiä (ISO 13407:1999). Siinä loppukäyttäjän tarpeet ja toivomukset otetaan erityisesti huomioon, jotta ohjelmistosta saadaan helppokäyttöinen ja käytännöllinen. UCD keskittyy käytettävyyteen koko kehittämisprosessin ja järjestelmän elinkaaren ajan (Gulliksen ym., 2003). Seuraavassa määritellään käytettävyyden käsite, esitellään käyttäjäkeskeisen suunnittelun kaksitoista perusperiaatetta ja sen suunnitteluprosessi.

3.1 Käytettävyys ja käyttäjäkeskeisen suunnittelun periaatteet

Nielsenin (1993) mukaan käytettävyys koostuu useista tekijöistä. Hän määrittelee käytettävyyden viiden ominaisuuden kautta, jotka ovat opittavuus, tehokkuus, muistettavuus, virheet ja tyytyväisyys. Seuraavassa määritellään nämä ominaisuudet:

- *Opittavuus* (learnability): Järjestelmän tulee olla niin helppokäyttöinen, että uusikin käyttäjä pystyy sisäistämään sen toimintaperiaatteen ja käytön.
- *Tehokkuus* (efficiency): Tehokkuus tarkoittaa sitä, että käyttäjä pystyy oppimiseen jälkeen käyttämään järjestelmää nopeasti ja tehokkaasti.
- *Muistettavuus* (memorability): Järjestelmän tulee olla helposti muistettava, jotta käyttäjän ei tarvitse opetella kaikkea uudelleen, kun järjestelmän käytössä on ollut tauko.
- *Virheet* (errors): Järjestelmän tulee sisältää mahdollisimman vähän virheitä, jotta käyttäjien tekemien virheiden määrä pysyy alhaisena. Lisäksi järjestelmän tulee palautua nopeasti mahdollisista käyttäjän virheistä.

- *Tyytyväisyys* (satisfaction): Järjestelmän tulee olla miellyttävä käyttää, jotta käyttäjät ovat siihen tyytyväisiä.

Myös toinen Nielsenin määritelmän ohella paljon käytetty käytettävyyden määritelmä on annettu ISO 9241-11 -standardissa (Jokela, Iivari & Tornberg, 2004). Standardin mukaan käytettävyys on mitta, jolla tuotteen määrätyt käyttäjät saavuttavat määritellyt tavoitteet tietyssä käyttökontekstissa tarkasti, tehokkaasti ja tyytyväisesti (ISO 9241-11:1998).

Tarkkuudella (effectiveness) tarkoitetaan, kuinka kokonaisvaltaisesti käyttäjä saavuttaa määritellyt tavoitteet. Tehokkuus (efficiency) tarkoittaa käytettyjen resurssien suhdetta saavutettuihin tavoitteisiin nähden. Tyytyväisyys (satisfaction) kuvaa käyttäjän positiivista suhtautumista tuotteen käyttöön. (ISO 9241-11:1998.)

Käyttäjäkeskeinen suunnittelu perustuu Gulliksenin ym. (2003) mukaan kahteentoista perusperiaatteeseen, joiden tarkoituksena on tukea kehittämisprosessia sekä helpottaa kommunikaatiota ja UCD-prosessin arviointia. Seuraavassa määritellään periaatteet Gulliksenin ym. (2003) mukaan:

Käyttäjäkeskeisyys: Kaikkien projektin jäsenten tulee ymmärtää esimerkiksi toiminnan tavoitteet, käyttökonteksti, käyttäjien tavoitteet, tehtävät ja tarpeet sekä kuinka käyttäjät kommunikoivat ja tekevät yhteistyötä. Tämän seurauksena voidaan keskittyä teknisen fokuksen sijaan käyttäjän tarpeisiin.

Aktiivinen käyttäjän osallistuminen: Osallistuvien käyttäjien täytyy aktiivisesti olla mukana projektissa koko kehitysprosessin sekä järjestelmän elinkaaren ajan. Lisäksi käyttäjien tulee edustaa tavoiteltuja käyttäjäryhmiä. Suunnitelmat käyttäjien mukaan ottamiseksi täytyy määritellä heti projektin alussa. On myös tärkeää, että käyttäjät tavataan oikeassa käyttöympäristössä, esimerkiksi heidän työpaikallaan.

Evolutionaarinen järjestelmäkehitys: Kehityksen tulee olla sekä iteratiivista että inkrementaalista. Iteraation tulee sisältää käyttäjien tarpeiden ja käyttöympäristön kattava analyysi, suunnitteluvaihe, arvio muutosehdotuksista sekä uusi suunnitelma arvion tulosten perusteella. Inkrementaalisessa suunnittelussa järjestelmälle asetetaan prioriteetit ja se jaetaan julkaisukelpoisiin osiin. Inkrementtien arvioinnin tulee vaikuttaa seuraavien inkrementtien suunnitteluun, jolloin ohjelmistosta kehittyy lopulta valmis tuote.

Prototyypitys: Prototyyppien käyttö tukee luovaa prosessia. Sen avulla voidaan tuoda esille vaatimuksia sekä visualisoida ideoita ja ratkaisuja. Prototyyppien suunnittelussa on tärkeää aloittaa ensin karkean tason suunnitelmista, esimerkiksi paperille luonnosteltuna, ennen varsinaisen ohjelmakoodin kirjoittamista. Prototyyppejä voidaan luoda myös rinnakkain, mikä auttaa suunnittelijoita pysymään avoimina ja luovina suunnittelun ajan.

Yksinkertaiset suunnitteluratkaisut: Suunnitelmien esitystavan ja käsitteistön tulee olla sellaisia, että ne ovat helposti ymmärrettävissä. Tavoitteena on, että kaikki projektin käyttäjät ja sidosryhmät ymmärtävät, mitä projektissa kehitetään. Ymmärtämistä helpottaa esimerkiksi prototyypitys ja simulointi.

Käyttökontekstin arviointi: Tärkeimmät käytettävyystavoitteet tulee olla määritelty ja suunnittelun tulee perustua suunnittelukriteereihin. Suunnitteluratkaisut arvioidaan oikeassa käyttöympäristössä yhdessä käyttäjien kanssa. Käyttäjien reaktioita, käytöstä, mielipiteitä ja ideoita tulee tarkkailla, tallentaa ja analysoida. Tulosten perusteella määritellään käytettävyyden kannalta tärkeiden näkökulmien tavoitteet.

Selkeät ja harkitut suunnitteluaktiviteetit: Järjestelmässä käyttäjän vuorovaikutus ja käytettävyys ovat tietoisten ja harkittujen suunnitteluaktiviteettien tulosta.

Ammattimainen ote: Järjestelmän suunnittelussa ja kehittämisprosessissa vaaditaan usein monenlaisia taitoja ja asiantuntemusta. Tämän vuoksi analyysi-, suunnittelu- ja kehitystyö tulee suorittaa valtuutetun monitaitoisen tiimin toimesta. Työssä vaaditaan ammattimaista otetta sekä yhteistyökykyä ja tehokkuutta.

Kokeneen käytettävyysekspertin mukaan ottaminen: Kehitystiimissä tulee olla mukana joko kokenut käytettävyysasiantuntija tai käytettävyystiimi. Käytettävyysasiantuntijalla on oltava valta päättää järjestelmän käytettävyyteen liittyvistä asioista. Hänen on oltava omistautunut projektille alusta saakka koko kehitysprojektin ja järjestelmän elinkaaren ajan.

Holistinen suunnittelu: Järjestelmää tulee kehittää samanaikaisesti kaikista eri näkökulmista. Näitä näkökulmia ovat muun muassa työkäytänteet, työympäristö, käyttöliittymä ja vuorovaikutus, käyttäjäkoulutus sekä turvallisuus. Yhdellä henkilöllä tai tiimillä tulee olla vastuu kaikkien näkökulmien yhdistämisestä.

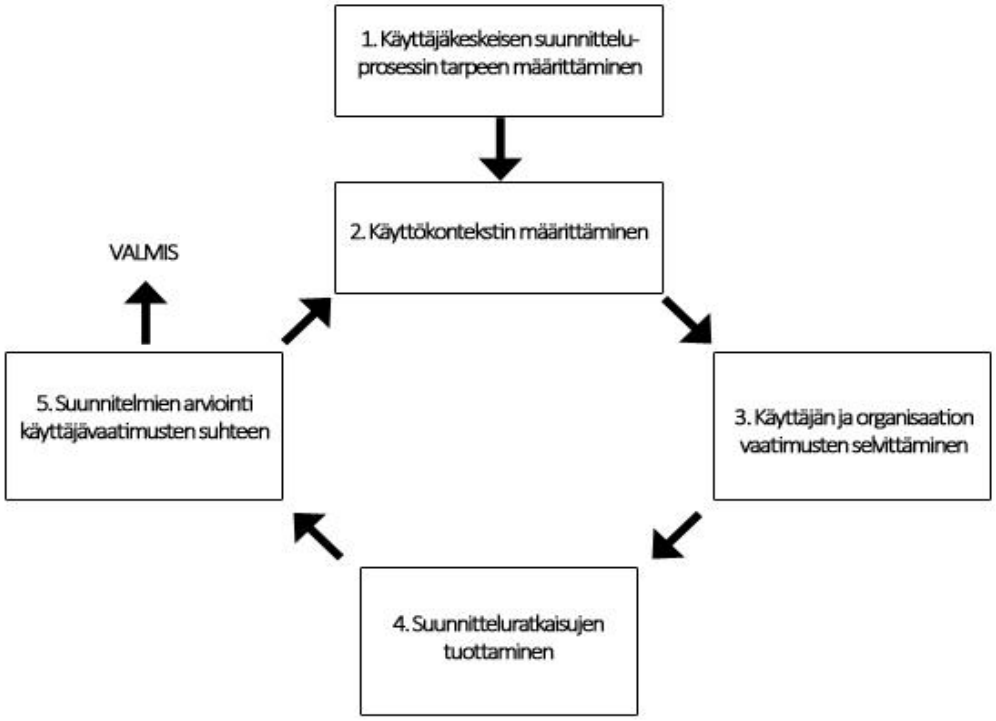
UCD-prosessin muokkaaminen: Käyttäjäkeskeinen kehittämisprosessi tulee kustomoida ja mukauttaa sopivaksi jokaiselle organisaatiolle sen erityistarpeisiin perustuen.

Käyttäjäkeskeisen asenteen vakiinnuttaminen: Projektitiimin, kehitysorganisaation ja asiakasorganisaation täytyy tiedostaa ja ymmärtää käytettävyyden ja käyttäjän mukanaolon tärkeys projektissa. Kaikilta osapuolilta odotetaan käyttäjäkeskeistä asennetta.

Edellä mainittujen periaatteiden noudattamisesta saadaan useita hyötyjä. Periaatteet auttavat esimerkiksi säilyttämään fokuksen käyttäjissä ja käytettävyydessä koko kehittämisprosessin ajan. (Gulliksen ym., 2003.)

3.2 Suunnitteluprosessi

Käyttäjäkeskeinen suunnittelu koostuu seuraavista vaiheista: 1) käyttäjäkeskeisen suunnitteluprosessin tarpeen määrittäminen, 2) käyttökontekstin määrittäminen, 3) käyttäjän ja organisaation vaatimusten selvittäminen, 4) suunnitteluratkaisujen tuottaminen ja 5) suunnitelmien arviointi käyttäjävaatimusten suhteen. Kuvio 3 esittää UCD:n suunnitteluprosessin etenemisen. (ISO 13407:1999.)



KUVIO 3 UCD:n suunnitteluprosessi (ISO 13407:1999)

Käyttäjäkeskeisen suunnitteluprosessin tarpeen määrittäminen käynnistää UCD-prosessin. Tässä vaiheessa asetetaan järjestelmän tavoitteet sekä määritellään käytettävyysvaatimukset.

Käyttökontekstin määrittämisen tarkoituksena on oppia tuntemaan käyttöympäristö, jossa järjestelmää käytetään. Lisäksi tavoitteena on selvittää, millainen on järjestelmän käyttäjä ja kuinka hän operoi järjestelmän kanssa.

Käyttäjän ja organisaation vaatimusten selvittämisessä määritellään järjestelmän kannalta oleelliset toiminnalliset vaatimukset niin käyttäjän kuin organisaationkin näkökulmasta. Lisäksi vaiheessa esitetään käyttäjäkeskeisen suunnittelun tavoitteet sekä määrätään suositukset ja rajoitukset suunnitelmien luomiselle.

Suunnitteluratkaisujen tuottamisessa hyödynnetään aikaisempaa tietämystä esimerkiksi käytettävyydestä tai visuaalisesta suunnittelusta. Nimensä mukaisesti vaiheen tarkoituksena on kehittää suunnitteluratkaisuja asetettujen tavoitteiden saavuttamiseksi. Tässä vaiheessa suunnitelma esitellään käyttäjille. Käyttäjiltä saadun palautteen perustella järjestelmästä voidaan luoda konkreettisempia prototyyppejä tai simulaatioita.

Suunnitelmien arviointi käyttäjävaatimusten suhteen on tärkeä vaihe käyttäjäkeskeisessä suunnittelussa. Sen tulee tapahtua kaikissa järjestelmän elinkaaren vaiheissa. Arvioinnin tarkoituksena on hankkia palautetta, jonka avulla voidaan parantaa suunnitelmia. Lisäksi arvioidaan, ovatko asetetut käyttäjä- ja organisatoriset vaatimukset saavutettu. (ISO 13407:1999.)

4 KÄYTTÄJÄKESKEISEN SUUNNITELUN INTEGROINTI KETTERIIN MENETELMIIN

Tässä luvussa esitellään tutkimuksia, joissa on tarkasteltu ketterän ohjelmistokehityksen ja käyttäjäkeskeisen suunnittelun piirteiden integrointia. Ketterien menetelmien joukosta on käsittelyssä Scrum ja XP. Aluksi esitellään esimerkkitapauksia aihepiirin aikaisemmista tutkimuksista, jonka jälkeen luvussa kootaan yhteen esiin nousseita yhtäläisyyksiä menetelmien integrointitavoissa. Lopuksi esitellään menetelmien liittämisestä mahdollisesti koituvia haasteita.

4.1 Esimerkkitutkimuksia integroinnista

Singh (2008) esittelee tutkimuksessaan U-Scrum -menetelmän, joka on muunnos perinteisestä Scrum-menetelmästä. Siinä keskitytään erityisesti käytettävyyden edistämiseen. Singhin (2008) mukaan Scrum ei itsessään sovellu tuotteille, joissa käytettävyys on tärkeää, koska se ei kykene täyttämään asiakkaiden käytettävyystarpeita riittävän hyvin. Tämä johtuu muun muassa siitä, että Scrumin tuoteomistajan markkinointiin ja myyntiin liittyvät huolet estävät riittävän keskittymisen käytettävyyteen. Ongelman ratkaisemiseksi U-Scrumissa tuoteomistajan rooli on jaettu kahdelle eri henkilölle, joista toinen on vastuussa erityisesti tuotteen käytettävyydestä ja käyttäjäkokemuksesta. Molemmat tuoteomistajat ovat projektissa tasa-arvoisia, mikä tarkoittaa sitä, että kummankaan tuoteomistajan näkemykset tai huolet eivät ole toisen näkemyksiä tai huolia alempiarvoisempia. Tuoteomistajat sopivat yhdessä projektisuunnitelman ja tuotteen kehitysjonon sisällöstä. Kehitysjono vastaa rakenteeltaan perinteistä tuotteen kehitysjonoa, mutta siinä otetaan enemmän huomioon käytettävyyden näkökulma. Lisäksi tuoteomistajat sopivat käyttäjäkokemuksen visiosta kuitenkin niin, että käytettävyystuoteomistaja on päävastuussa vision luomisesta. Käytettävyystuoteomistajan vastuulla on myös tuottaa käyttäjäprofiileja, jotka kuvastavat tuotteen tavoiteltuja käyttäjiä. Visio esitetään projektin sidosryhmille, jotta kaikki osapuolet ymmärtävät projektin keskeiset tavoitteet. Tämä hel-

pottaa sidosryhmien välistä kommunikaatiota sekä parantaa yleisesti käytettävyyttä ja tuottavuutta.

Ungar ja White (2008) ehdottavat tapaustutkimuksessaan UCD:n ja ketterän kehityksen yhdistämistä hyödyntämällä yhden päivän kestävän työpajan, design studion, käyttöönottamista. Siinä projektin eri osapuolet, mukaan lukien kehitystiimi ja UCD-tiimi, kokoontuvat yhteen keskustelemaan jokaisen osapuolen tehtävistä ja näkemyksistä. Tilaisuudessa osallistujat tuottavat karkeita luonnoksia suunnitelmista, jonka jälkeen he yhdistävät ideansa yhdeksi ratkaisuksi, joka toimii lähtökohtana tulevalle kehitystyölle. Design studio on nopea prosessi, joka mahdollistaa suunnittelijoiden, kehittäjien ja sidosryhmien välisen yhteistyön ja auttaa vaihtoehtoisten suunnitteluratkaisujen löytämisessä. Design studion ansiosta tuotteen suunnittelua voidaan suorittaa kehitystä edellä, jolloin suunnittelutiimillä on riittävästi aikaa käyttäjätutkimukselle ja suunnitelmien tekemiselle. Lisäksi työpajan käyttö edistää tiimien välistä yhteistyötä ja auttaa löytämään yhteisen suunnitteluvision.

Budwig, Jeong ja Kelkar (2009) kertovat tapaustutkimuksessaan yrityksensä UX-tiimin kokemuksista laajan Scrum-projektin yhteydessä. Projekti toteutettiin kolmessa eri vaiheessa, joiden aikana kehittämisprosessissa ja rooleissa havaittiin useita puutteita. Näihin ongelmakohtiin kehitettiin joka vaiheen jälkeen uusia toimintatapoja, jotta UX-tiimi pystyi hyödyntämään Scrum-prosessia tehokkaammin. Prosessiin lisättiin normaalien sprinttien tueksi erilliset neljännesvuosittaiset sprintit suunnittelun visioimista varten. Tällä tavalla UX-tiimi pystyy kokonaisvaltaisesti arvioimaan projektin tulevaisuutta pidemmällä aikavälillä. Lisäksi projektissa todettiin hyödylliseksi, että UX-tiimi työskentelee yhden tai kaksi sprinttiä kehitystiimiä edellä. Myös UX-tiimin erottaminen omaksi Scrum-tiimikseen kehitystiimin rinnalle koettiin hyödylliseksi. Tämä tarkoittaa käytännössä sitä, että myös UX-tiimillä on oma tuoteomistaja sekä tuotteen kehitysjono. Prosessiin tehdyt muutokset näkyivät muun muassa UX-tiimin ja kehitystiimin parantuneena yhteistyönä sekä laadukkaampina ja yhtenäisempinä suunnitelmina.

Myös Chamberlainin, Sharpin ja Maidenin (2006) kenttätutkimus selvittää UCD:n ja ketterien menetelmien yhdistämistä kolmessa eri ohjelmistokehitysprojektissa. Kaikissa projekteissa oli kehitystiimin lisäksi mukana erillinen suunnittelutiimi. Tutkimuksen pohjalta kehitettiin viitekehys, joka auttaa edellä mainittujen menetelmien integroimisessa. Sen periaatteita ovat esimerkiksi käyttäjän mukanaolo kehitysprosessissa sekä käytettävyys- ja kehitystiimin välinen tiivis yhteistyö ja kommunikaatio. Lisäksi menetelmien integrointia helpottaa prototyyppien käyttö suunnittelun yhteydessä sekä se, että UCD:lle on varattu riittävästi aikaa käyttäjien tarpeiden selvittämiseksi. Jokaisessa projektissa hyödynnettiinkin etukäteissuunnittelua ennen itse kehittämisen alkamista. Tutkimuksen mukaan viitekehyksen noudattaminen hyödyttää sekä yritystä että käyttäjää.

Najafi ja Toyoshiba (2008) kertovat tapaustutkimuksessaan käyttäjäkokemussuunnittelun (UX) periaatteiden liittämisestä ketteriin menetelmiin tuotteen käytettävyyden parantamiseksi. Tämä saavutettiin ottamalla erillinen UX-tiimi

mukaan kehitykseen, jolloin käyttäjien tarpeet ja odotukset pystyttiin huomioimaan entistä paremmin. Näin käyttäjätutkimusta ja testaamista voitiin hyödyntää tuotteen kehitysjonon ominaisuuksien priorisoinnissa. Samalla myös järjestelmän suunnitelma kehittyi iteratiivisesti ja saavutettiin parempi käytettävyys. UX-tiimin liittäminen ketterään prosessiin vaatii Najafin ja Toyoshiban (2008) mukaan täysivaltaista yhteistyötä ketterän kehitystiimin kanssa. Sen osallistuminen muun muassa Scrumin päiväpalavereihin on erittäin tärkeää kehitysprosessin kannalta, jotta kehitystiimin tekemä työ saadaan keskitettyä siten, että se täyttää käyttäjien vaatimukset.

Silva da Silvan (2012) ketterien menetelmien ja UCD:n integroimiseen liittyvässä tutkimuksessa havaittiin, että järjestelmän suunnittelu ja mallintaminen kannattaa aloittaa hieman ennen toteuttamisvaiheen alkua tai jopa ennen projektin virallista käynnistämistä. Samalla määritellään järjestelmän sisältö ja rakenne. Tämä mahdollistaa sen, että käytettävyystiimi on aina yhden sprintin edellä kehitystiimiä käyttäjätutkimus- ja suunnitelma-asioissa. Toinen tutkimuksessa esiin noussut seikka oli käytettävyystiimin ja kehitystiimin välinen tiivis yhteistyö. Käytettävyystiimin tehtävänä on tukea kehitystiimiä sprintin aikaisessa suunnittelussa, antaa palautetta sekä suorittaa tarkastusarviointeja sen hetkisen sprintin toteuttamiseen liittyen. Käyttäjäkeskeisen suunnittelun hyödyntäminen ketterässä kontekstissa helpottuu, kun interaktiosuunnittelijat ovat kokoaikaisia tiimin jäseniä, jotka työskentelevät samoissa tiloissa kehittäjien kanssa.

Sy (2007) teki kirjoituksessaan vastaavanlaisia havaintoja kuin Silva da Silva (2012). Hänen lähestymistavassaan suunnittelun ja toteutuksen iterointi tapahtuu erillään, mutta rinnakkain. Tämä tarkoittaa sitä, että ennen varsinaisia iteraatioita käytettävyystiimi pitää lyhyen sprintin, nollasprintin, jossa suunnitellaan ja kerätään käytettävyysvaatimuksiin liittyvää dataa. Ensimmäisen sprintin alkaessa kehitystiimi alkaa toteuttaa kyseisen sprintin kehitysjonoa sillä välin, kun käytettävyystiimi suunnittelee jo seuraavan sprintin käytettävyyttä. Kun kehitystiimi saa ensimmäisen sprintin valmiiksi, on sillä jo valmiina käytettävyysdataa seuraavan sprintin toteuttamista varten.

Memmel, Gundelsweiler ja Reiterer (2007) esittelevät tutkimuksessaan XP:hen pohjautuvan CRUISER-mallin, joka yhdistää ohjelmistokehityksen sekä käytettävyyssuunnittelun ketterässä ympäristössä. Malli hyödyntää menetelmien samankaltaisuuksia, joita löytyy muun muassa niiden perusperiaatteista ja käytännöistä. Tutkimuksen mukaan skenaarioiden ja prototyyppien käyttö prosessin keskeisenä artefaktina edistää käyttäjien ja sidosryhmien osallistumista suunnitteluprosessiin.

Fox, Sillito ja Maurer (2008) loivat tutkimuksessaan aikaisempiin alan tutkimuksiin pohjautuvan laajennetun, yleisemmän tason mallin UCD:n ja ketterien menetelmien yhdistämiseksi. Malli sisältää kolme erilaista lähestymistapaa integroinnin saavuttamiseksi. Niitä kutsutaan yleiseksi (generalist), spesialisti- (specialist) ja hybridi-lähestymistavaksi (hybrid).

Spesialisti-lähestymistapa koostuu kolmesta päärooliista: käyttäjistä, UCD-asiantuntijasta ja kehitystiimistä. Lähestymistavan alkuvaiheessa UCD-

asiantuntija tekee taustatutkimusta ja luo karkean tason prototyyppejä. Prototyypin pohjalta luodaan suunnitelma käyttöliittymälle, jonka jälkeen kehitystiimi vahvistaa, että suunnitelma on teknisesti toteutettavissa. Sillä välin, kun kehitystiimi toteuttaa iteraatiossa määrättyjä ominaisuuksia, tekee UCD-asiantuntija jo lisää käyttäjätutkimusta seuraavaa iteraatiota varten. Iteraation päätteeksi UCD-asiantuntija suorittaa vielä käytettävyystestejä ja mikäli toteutetut ominaisuudet vastaavat käyttäjän odotuksia, on iteraatio valmis. (Fox ym., 2008.)

Yleisessä lähestymistavassa on käytössä vain kaksi pääroolia: käyttäjät sekä kehittäjät, jotka toimivat samanaikaisesti myös UCD-asiantuntijoina ilman virallista koulutusta UCD:stä. Spesialisti-lähestymistavasta poiketen yleisessä lähestymistavassa käytetään useita UCD-asiantuntijoita. Lisäksi projektin alkuvaihe kestää spesialisti-lähestymistapaa lyhyemmän ajan. Yleisessä lähestymistavassa prototyypityksen jälkeen pidetään suunnittelupalaveri, jossa valitaan prioriteetin mukaan seuraavassa iteraatiossa toteutettavat ominaisuudet. Tämän jälkeen aloitetaan iteraation toteutusvaihe. Mahdolliset käytettävyysongelmat voidaan ratkaista heti ilman muita osapuolia, sillä kehittäjät huolehtivat itse tuotteen käytettävyydestä. (Fox ym., 2008.)

Hybridi-lähestymistapa on hyvin samankaltainen edellä mainittujen lähestymistapojen kanssa. Se noudattaa samoja käytänteitä projektin aloitusvaiheessa sekä iteraatioiden aikana. Tässä lähestymistavassa yhdellä tiimin jäsenellä oli aikaisempaa kokemusta sekä UCD-asiantuntijana että kehittäjänä toimimisesta. (Fox ym., 2008.)

4.2 Yhteenvetoa tutkimuksista

Oleellinen toimintapa, joka esiintyi useissa aihepiirin tutkimuksissa, oli käytettävyystiimin työskentely ajallisesti kehitystiimiä edellä. Käytännössä tämä tarkoittaa, että käytettävyystiimin tulisi suorittaa käyttäjätutkimusta, suunnittelua ja testausta keskimäärin yksi tai kaksi sprinttiä toteuttamista edellä. Lisäksi ennen toteuttamisen aloittamista pidettävä nollasprintti, jossa käytettävyystiimi määrittelee projektin vaatimuksia sekä luo alustavia käyttäjäkertomuksia, mainittiin useassa tutkimuksessa. Edellä mainitut tavat helpottavat toteuttamisprosessia ja vähentävät sitä kautta turhaa työtä. (esim. Felker ym., 2012; Sy, 2007; Silva da Silva, 2012.)

Budwig ym. (2009) esittivät lisäksi ketterän kehittämisprosessin tukemiseksi neljännesvuosittain tapahtuvia sprinttejä, joissa visioidaan ja suunnitellaan pitkän aikavälin tavoitteita kehittämisprojektissa. Myös prototyyppien käyttö suunnittelun yhteydessä on havaittu hyödylliseksi (Chamberlain ym., 2006; Memmel ym., 2007).

Ohjelmistokehityksen organisoimiseksi esitettiin käytettävyysasiantuntijan (Singh, 2008) tai erillisen käytettävyystiimin (Budwig ym., 2009; Najafi ym., 2008) käyttämistä ketterän kehittämisprosessin yhteydessä. Tällä tavalla käyttäjien tarpeet pystytään ottamaan paremmin huomioon ja tätä kautta myös

tuotteen käytettävyyttä voidaan parantaa. Lisäksi esimerkiksi tuoteomistajan rooli voidaan jakaa kehitys- ja käytettävyystiimin kesken, jolloin toinen tuoteomistajista keskittyy yksinomaan käytettävyyden vaalimiseen (Singh, 2008).

Toinen tutkimuksissa esiintynyt seikka oli käytettävyys- ja kehitystiimin välinen yhteistyö (Najafi ym., 2008). Käytettävyystiimin tulee toimia tiiviissä yhteistyössä kehitystiimin kanssa. Lisäksi on suositeltavaa, että käytettävyystiimi työskentelee kokoaikaisesti samassa paikassa kehitystiimin kanssa (Silva da Silva, 2012). Sujuva yhteistyö vaatii hyvää kommunikaatiota ja halukkuutta tiimityöskentelyyn. Esimerkiksi osallistuminen yhteisiin päiväpalavereihin on erittäin tärkeää. Kommunikointia voidaan edistää esimerkiksi pitämällä erillinen suunnittelupalaveri, jossa osapuolet kokoontuvat yhteen kertomaan omista näkemyksistään pyrkien yhteisymmärrykseen (Ungar ym., 2008). Hyvän tiimienvälisen kommunikaation ansiosta suunnittelussa voidaan välttää väärinymmärryksiä ja tärkeiden tietojen häviämistä prosessin aikana (Sy & Miller, 2008).

4.3 Integroinnin haasteet

Vaikka käyttäjäkeskeisen suunnittelun integroiminen ketteriin menetelmiin koetaan hyödylliseksi, on menetelmien yhdistäminen samalla haasteellista. Seuraavaksi kerrotaan tutkimuksissa esiin nousseita ongelmakohtia.

UX-tiimillä ei välttämättä ole varattu tarpeeksi aikaa suunnittelua varten, jolloin suunnitelmien laatu saattaa heikentyä. Tämä näkyy muun muassa siinä, että kehitystiimi joutuu odottamaan suunnittelun valmistumista, mikä saattaa hidastaa kehitysprosessin kulkua. Toinen ongelma, johon käytettävyystiimi saattaa törmätä, on sprinttien liian lyhyt kesto ja vähäinen suunnittelu. Tällöin suunnittelua ei ehditä saada valmiiksi ajoissa, eikä aikaa jää esimerkiksi käytettävyystestaukselle tai asiakaskontakteille. Lisäksi riittämätön tai liian myöhään annettu käyttäjäpalaute saattaa aiheuttaa ongelmia prosessin edetessä. (Sy & Miller, 2008.)

UCD:n integrointi ketteriin menetelmiin saattaa koitua hankalaksi myös, mikäli UX-tiimi työskentelee yhtä aikaa useammassa kuin yhdessä projektissa (Sy & Miller, 2008). Varsinkin ulkomailla työskentelevien tiimien koordinointi saattaa olla haastavaa (Budwig ym., 2009) Tällöin itse suunnitteluun ja iteraatioihin tarkoitettu aika kuluu helposti muihin asioihin, kuten kokouksiin. Siksi onkin tärkeää, että UX-tiimi työskentelee kokoaikaisesti vain yhdessä projektissa kerrallaan. (Sy & Miller, 2008.)

Ketterien menetelmien ja UCD:n integroimisesta tekee vaikeaa muun muassa tiimin puutteellinen kommunikaatio (Sohaib & Khan, 2010). Kommunikaatio-ongelmat voivat johtaa usein tärkeän tiedon häviämiseen ja väärinymmärryksiin suunnittelun suhteen. Tämän seurauksena kehitystiimi ei välttämättä usko UX-tiimin esittämiin suunnitelmiin (Sy & Miller, 2008).

5 YHTEENVETO

Tämän tutkielman tarkoituksena oli selvittää, kuinka käyttäjäkeskeisen suunnittelun piirteitä voidaan liittää ketterän kehittämisen prosessiin ja rooleihin. Tutkielman alussa kuvattiin ketterän ohjelmistokehityksen taustaa ja esiteltiin Scrum- ja XP-menetelmät. Menetelmistä kerrottiin niiden prosesseista ja rooleista. Scrumin prosessi perustuu lyhyisiin työsykleihin, joita kutsutaan sprinteiksi. Scrum-tiimi koostuu tuoteomistajasta, kehitystiimistä ja Scrum-mestarista. XP:n prosessi koostuu viidestä vaiheesta ja sen rooleja ovat ohjelmoija, käyttäjä, testaaja, projektipäällikkö, tuotepäällikkö, johtaja, arkkitehti, interaktiosuunnittelija sekä tekninen kirjoittaja.

Toiseksi tutkielmassa tarkasteltiin käytettävyyden määritelmiä ja käyttäjäkeskeistä suunnittelua. Samalla esiteltiin käyttäjäkeskeisen suunnittelun kaksitoista pääperiaatetta sekä UCD:n suunnitteluprosessi.

Kolmanneksi kuvattiin eri tutkimuksissa esitettyjä ehdotuksia ketterien menetelmien ja käyttäjäkeskeisen suunnittelun integroimiseksi. Tarkastelun keskiössä olivat prosessia ja rooleja koskevat ehdotukset.

Tutkimusten pohjalta nousi esiin muutamia yhteneviä keinoja, joilla käyttäjäkeskeistä suunnittelua integroitiin ketterän kehittämisen prosessiin ja rooleihin. Ensimmäinen tällainen toimintatapa oli käytettävyysasiantuntijan tai erillisen käytettävyystiimin ottaminen mukaan kehittämisprosessiin. Tällä tavalla uskotaan pystyttävän huomiomaan käyttäjien tarpeet paremmin, jonka myötä myös tuotteen käytettävyys paranee. Toinen tutkimuksissa korostettu seikka oli käytettävyys- ja kehitystiimin välinen yhteistyö. Hyvän tiimien välisen kommunikaation ansiosta vältytään väärinymmärryksiltä eikä tärkeää tietoa häviä prosessin aikana. Kolmas toimintatapa, jonka uskotaan tukevan integrointia, on käytettävyystiimin työskentely ajallisesti kehitystiimin edellä. Käyttäjätutkimuksen, suunnittelun ja testauksen suorittaminen yksi tai kaksi sprinttiä ennen toteuttamista helpottaa koko toteuttamisprosessia ja vähentää turhaa työtä. Lisäksi integroinnin tueksi on ehdotettu prototyypityksen ja neljännesvuosittaisten suunnitteluspринttien hyödyntämistä.

Käyttäjäkeskeisen suunnittelun integroiminen ketteriin menetelmiin tuo myös omat haasteensa. Esimerkiksi käytettävyystiimille varattu liian vähäinen

suunnitteluaika saattaa näkyä tuotteen heikentyneenä laatuna. Myös käytettävyystiimin työskentely useissa projekteissa samanaikaisesti hankaloittaa integraation onnistumista. Tämän lisäksi myös puutteellinen kommunikaatio tiimien välillä saattaa aiheuttaa väärinymmärryksiä kehitysprosessissa.

Tutkielman tuloksia voidaan hyödyntää sellaisissa ohjelmistoprojekteissa, joissa ketterien menetelmien lisäksi halutaan korostaa käytettävyyden näkökulmaa. Menetelmien integroimisesta saatavat hyödyt näkyvät ohjelmiston parantuneena käytettävyytenä.

Tässä tutkielmassa tarkasteltiin käyttäjäkeskeisen suunnittelun integrointia ketterään kehittämiseen sen prosessin ja roolien näkökulmasta. Jatkossa menetelmien yhdistämistä voisi tutkia laajemmin myös muista näkökulmista.

Moni käsitellyistä tutkimuksista koski vain yksittäistä projektia tai tiettyä yritystä. Vaikka tutkimuksissa nousikin esiin samankaltaisia havaintoja, selkeitä yleisohjeita tai suosituksia menetelmien onnistuneeseen integroimiseen ei tuntunut löytyvän. Jatkossa olisikin mielenkiintoista tutkia, voisiko tutkielman tulosten perusteella muodostaa yleispätevän mallin, jota voisi hyödyntää universaalisti kaikenlaisissa projekteissa.

LÄHTEET

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2002). *Agile software development methods - Review and analysis.* VTT Publications (478). Espoo: Otamedia Oy.

Ambler, S. (2005). Quality in an agile world. *Software Quality Professional, 7(4)*, 34-40.

Beck, K. (1999). *Extreme Programming Explained: Embrace Change*. Boston: Addison-Wesley.

Beck, K. & Andres, C. (2004). *Extreme Programming Explained: Embrace Change* (2. painos). Boston: Addison-Wesley.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001a). Manifesto for Agile Software Development. Haettu 15.12.2013 osoitteesta http://agilemanifesto.org/

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001b). Principles behind the Agile Manifesto. Haettu 15.12.2013 osoitteesta http://agilemanifesto.org/principles.html

Budwig, M., Jeong, S. & Kelkar, K. (2009). When User Experience Met Agile: A Case Study. Teoksessa D.R. Olsen, Jr. (toim.), Proceedings of the 27th international conference extended abstracts on Human factors in computing systems, Boston, April 4 – 9 (s. 3075-3084). New York, NY: ACM Press.

Chamberlain, S., Sharp, H., & Maiden, N. (2006). Towards a Framework for Integrating Agile Development and User-Centred Design. Teoksessa Abrahamsson, P., Marchesi, M., Giancarlo, S. (toim.), *Extreme Programming and Agile Processes in Software Engineering: Proceedings of the 7th International Conference (XP 2006)* (s. 143–153). Berlin: Springer.

Cohn, M. (2011). Mountain Goat Software: Scrum Overview for Agile Software Development. Haettu 11.4.2014 osoitteesta http://www.mountaingoatsoftware.com/agile/scrum/overview

Felker, C., Slamova, R. & Davis, J. (2012). Integrating UX with Scrum in an Undergraduate Software Development Project. Teoksessa *SIGCSE'12 Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 301-306.

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

Huo, M., Verner, J., Zhu, L. & Babar, M. (2004). Software quality and agile methods. Teoksessa *Proceedings of the 28th Annual International Computer*

Software and Applications Conference (COMPSAC'04) (s. 520-525). Hong Kong: IEEE Computer Society.

Hussain, Z., Lechner, M., Milchrahm, H., Shahzad, S., Slany, W., Umgeher, M., Vlk, T., Köffel, C., Tscheligi, M. & Wolkerstorfer, P. (2012). Practical Usability in XP Software Development Processes. Teoksessa *The Fifth International Conference on Advances in Computer-Human Interactions ACHI 2012* (s. 208-217).

International Organization for Standardization. (1999). *ISO 13407:1999 Human-centred design processes for interactive systems*.

International Organization for Standardization. (1998). *ISO 9241-11: Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs): Part 11: Guidance on Usability*.

Jokela, T., Iivari, N. & Tornberg, V. (2004). Using the ISO 9241-11 definition of usability in requirements determination: case studies. Teoksessa A. Dearden & L. Watts (toim.), *Proceedings of HCI2004: Design for life, the 18th British HCI group annual conference vol. II* (s. 155-156). UK*:* Leeds Metropolitan University.

Memmel, T., Gundelsweiler, F. & Reiterer, H. (2007). Agile Human-Centered Software Engineering. Teoksessa L.J. Ball, M. Sasse, C. Sas, T.C. Ormerod, A. Dix, P. Bagnall & T. McEwan (toim.), *Proceedings of the 21st British HCI Group Annual Conference on People and Computers BCS-HCI '07, September 3–7* (s. 167-175).

Najafi, M. & Toyoshiba, L. (2008). Two Case Studies of User Experience Design and Agile Development. Teoksessa G. Melnik, P. Kruchten & M. Poppendieck (toim.), *Proceedings of AGILE 2008 Conference, August 4-8* (s. 531-536).

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

Patton, J. (2002). Hitting the Target: Adding Interaction Design to Agile Software Development. Teoksessa *Proceedings of OOPSLA*, (s. 1-7).

Rodriguez P., Markkula J., Oivo M., Turula K. (2012). Survey on agile and lean usage in Finnish software industry. Teoksessa *International Symposium on Empirical Software Engineering and Measurement (ESEM'12)*, (s. 139-148).

Schwaber, K., Sutherland, J. (2013). The Scrum Guide. Haettu 20.4.2014 osoitteesta https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide-FI.pdf

Silva da Silva, T. (2012). *A framework for integrating interaction design and agile methods*. Doctoral Thesis. University of Rio Grande do Sul.

Singh, M. (2008). U-SCRUM: An Agile Methodology for Promoting Usability. Teoksessa G. Melnik, P. Kruchten & M. Poppendieck (toim.), *Proceedings of AGILE 2008 Conference.*

Sohaib, O. & Khan, K. (2010). Integrating Usability Engineering and Agile Software Development: A Literature Review. Teoksessa J. Wang & B. Wang (toim.), *Proceedings of 2010 International Conference On Computer Design And Applications (ICCDA 2010)*, (s. 32-38).

Sutherland, J. (2010). Scrum Handbook. Haettu 21.4.2014 osoitteesta http://jeffsutherland.com/scrumhandbook.pdf

Sy, D. (2007). Adapting Usability Investigations for Agile User-Centered Design. *Journal of Usability Studies, 2(3),* 112-132.

Sy, D. & Miller, L. (2008).Optimizing Agile User-Centred Design. Teoksessa M. Burnett, M.F. Costabile, T. Catarci, B. de Ruyter, D. Tan, M. Czerwinski & A. Lund (toim.), *Proceedings Conference on Human factors in computing systems (CHI '08),* (s. 3987-3900).

Ungar, J. & White, J. (2008). Agile User Centered Design: Enter the Design Studio - A Case Study. Teoksessa M. Burnett, M.F. Costabile, T. Catarci, B. de Ruyter, D. Tan, M. Czerwinski & A. Lund (toim.), *Proceedings Conference on Human factors in computing systems (CHI '08).*

VersionOne. (2013). 8th Annual State of Agile Development Survey. Haettu 2.5.2014 osoitteesta http://www.versionone.com/pdf/2013-state-of-agile-survey.pdf