Tampereen yliopisto

Topi Kalermo

FULL STACK -SUUNNITTELIJAN ROOLIN TARPEELLISUUS KETTERÄSSÄ OHJELMISTOKEHITYKSESSÄ

Työntekijän näkökulma

Informaatioteknologian ja viestinnän tiedekunta
Pro gradu -tutkielma
Elokuu 2024

TIIVISTELMÄ

Topi Kalermo: Full stack -suunnittelijan roolin tarpeellisuus ketterässä ohjelmistokehityksessä: Työntekijän näkökulma
Pro gradu -tutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Elokuu 2024

Full stack -suunnittelu ja full stack -suunnittelijan rooli ovat melko tuntemattomia termejä ohjelmistokehityksessä. Tämän tutkielman tavoitteena on selvittää, mitä hyötyä full stack -suunnittelijasta on ketterälle ohjelmistokehitysprojektille työntekijän näkökulmasta. Tämä tutkimus rakentuu kahdesta pääosasta: kirjallisuuskatsauksesta ja empiirisestä tutkimuksesta. Yhdessä nämä osat tarjoavat kattavan ja syvällisen tarkastelun tutkimusaiheesta. Kirjallisuuskatsauksessa analysoidaan aiempia tutkimuksia ja teoreettista taustaa liittyen käyttäjäkeskeisen suunnittelun rooleista ja sen integroimisesta ketteriin menetelmiin. Kirjallisuuskatsauksen tulosten perusteella laadittiin empiirisen haastattelun teemat ja niihin liittyvät haastattelukysymykset. Empiiriseen materiaalin luomiseen osallistui kolme suunnittelijaa, kaksi full stack -kehittäjää sekä yksi testaaja ja yksi applikaatiokonsultti.

Tehdyn empiirisen tutkimuksen havainnot olivat suurimmaksi osaksi yhteneviä kirjallisuuskatsauksen tulosten kanssa, mutta niistä löytyi myös kiinnostavia poikkeamia. Tämän tutkimuksen kirjallisuuskatsauksessa todettiin, että full stack -suunnittelijalla ei ole vielä vakiintunutta roolia ketterässä ohjelmistokehityksessä, eikä sen hyödyistä ja haasteista ole selvää kuvaa. Haastattelut vahvistivat tätä käsitystä, sillä kukaan haastatelluista ei ollut aiemmin kuullut roolista, vaikka osa osasi arvella sen merkitystä. Haastateltavat näkivät full stack -suunnittelijan potentiaalisen hyödyn projektien kokonaisuuden hallinnassa ja ajansäästössä, mutta toivat esiin myös haasteita, kuten ajanhallinnan vaikeuden ja riskin tulla pullonkaulaksi suuremmissa projekteissa.

Kirjallisuuskatsaus ja haastattelut korostivat myös käyttäjäkeskeisen suunnittelun ja ketterän kehityksen integraation haasteita, joita voidaan lieventää tiiviillä yhteistyöllä ja etupainotteisella suunnittelulla. Palvelumuotoilijan rooli nousi esiin erityisen tärkeänä projektien alkuvaiheessa palvelukonseptin määrittelyssä. Keskeinen tulos on, että full stack -suunnittelijan rooli on hyödyllinen pienissä projekteissa, mutta suurissa projekteissa se voi aiheuttaa ajankäytöllisiä haasteita, vaikka kehittäjien ja suunnittelijoiden välinen yhteistyö sujuu yleensä hyvin, kun käyttäjäkokemussuunnittelu on integroitu kunnolla.

Avainsanat: ohjelmistokehitys, ketterä kehitys, full stack -suunnittelija, full stack -suunnittelu

Opinnäytteessäni käytetyt tekoälytyökalut ja niiden käyttötarkoitukset on kuvailtu alla:

Työkalun nimi (ja versio): ChatGPT GPT-4o mini

Käyttötarkoitus ja osio, jossa työkalua käytettiin: Haastattelutulosten analysointi ja teemoittelu luvussa 4.

Olen tietoinen siitä, että olen täysin vastuussa koko opinnäytteeni sisällöstä, mukaan lukien tekoälyllä tuotetut osat, ja hyväksyn vastuun mahdollisista eettisten ohjeiden rikkomuksista.

1 Johdanto

Tehtävänimike ”full stack -suunnittelija” on melko tuntematon termi suomalaisessa ohjelmistokehityksessä. Viitanen [2018] suoritti tutkimuksen full stack -suunnittelijan hyödyistä ja haitoista ketterässä ohjelmistokehityksessä vuonna 2018, mutta tämän jälkeen termiä ei ole tutkittu Suomessa eikä tehtävänimikkeellä löydy töitä Suomesta. Nykyisessä kokemuspainotteisessa maailmassa ketterä käyttäjäkokemussuunnittelu on ajankohtainen ja suosittu aihe [Silva ym., 2018].

Vaikka full stack -suunnittelijan roolia on tutkittu vain vähän, ketterä ohjelmistokehitys on sen sijaan vakiinnuttanut asemansa ohjelmistokehityksen keskeisenä menetelmänä. [VersionOne, 2020] Ketterä ohjelmistokehitys (engl. Agile Software Development, ASD) on saavuttanut suosiota, koska sen menetelmät soveltuvat erilaisiin ohjelmointiparadigmoihin sekä se hyödyntää erilaisia kehyksiä, menetelmiä ja tekniikoita ohjelmiston laadun parantamiseksi. Se muodostaa olennaisen osan ohjelmistokehitysprosessia, painottaen inkrementaalista toimitusta, tiimityöskentelyä, jatkuvaa suunnittelua ja oppimista koko kehitysprosessin ajan. Ennen kuin ketterä kehitys viimeistelee mitään, se käy läpi useita iteraatioita palautteen perusteella. Tämän seurauksena prosessi muuttuu dynaamisemmaksi, kun kaikki osallistuvat yhteisen päämäärän saavuttamiseen. [Hooda, 2023] Ketterä kehitys ei kuitenkaan usein takaa käyttäjäystävällistä lopputulosta, jos ohjelmistoa on kehitetty nopeassa aikataulussa tai pelkästään tekninen toteutus mielessä. Täten onkin tärkeää suorittaa tutkimusta asiakkaan tarpeista sekä tehdä vaatimusmääritelmä ennen kehityksen alkua sekä suorittaa käytettävyystutkimuksia kehityksen aikana. Näin voidaan varmistaa, että kehitettävästä ohjelmistosta tulee sellainen, joka toimii asiakkaan sekä käyttäjien tarpeiden mukaan. [Brhel ym., 2015]

Ketterän kehityksen merkityksen ymmärtämisen ohella on tärkeää tutkia, kuinka full stack -suunnittelijan rooli istuu tähän kehykseen. Tämän tutkielman tarkoituksena on selvittää full stack -suunnittelijan roolin tarpeellisuutta ketterässä ohjelmistokehityksessä työntekijöiden näkökulmasta. Lisäksi full stack -suunnittelijan rooli tullaan täsmentämään tämän työn aikana. Tutkimuskysymys on seuraava:

- Miten työntekijät kokevat full stack -suunnittelijan roolin ketterässä ohjelmistokehityksessä?

Tähän tutkimuskysymykseen vastataan kirjallisuuskatsauksella, jossa tutkitaan aiempia tutkimuksia aiheesta tai siihen liittyvistä aiheista, sekä tämän tutkielman aikana tehtävällä empiirisellä tutkimuksella, jossa haastatellaan alan ammattilaisia aiheeseen liittyen. Jotta

tutkimuksen laajuus ei olisi liian suuri, aihe rajataan työntekijöiden näkökulmaan, joiden projekteihin on integroituna käyttäjäkeskeinen suunnittelu. Haastattelemalla suunnittelijoita, kehittäjiä sekä muita alan ammattilaisia voidaan selvittää, miten nykyisissä projekteissa käyttäjäkeskeinen suunnittelu toimii ja miten työntekijät kokevat full stack -suunnittelijan roolin ketterässä ohjelmistokehityksessä.

Full stack -suunnittelijan roolia ketterässä ohjelmistokehityksessä on tutkittu vähän, vaikka käyttäjäkeskeisen suunnittelun integroimista ketteriin menetelmiin on käsitelty laajasti. Tämän tutkimuksen tavoitteena on selvittää, millaisia vaikutuksia full stack -suunnittelijalla on ketterässä ohjelmistokehityksessä ja miten muut ketterän kehityksen roolit kokevat kyseisen roolin. Vaikka full stack -suunnittelijasta löytyy runsaasti blogikirjoituksia ja verkkosisältöä, tieteellistä tutkimusta aiheesta on huomattavasti vähemmän.

Tutkielman luvussa 2 käsitellään tutkimuksen kannalta tärkeitä käsitteitä käyttäjäkeskeisestä suunnittelusta ja ketterästä ohjelmistokehityksestä sekä tarkastellaan, kuinka nykyisessä ketterässä ohjelmistokehityksessä otetaan huomioon käyttäjäkeskeinen suunnittelu. Näiden lisäksi kerrotaan mitä full stack -suunnittelulla tarkoitetaan sekä määritellään full stack -suunnittelijan rooli ja selitetään, että mitä sillä tarkoitetaan tässä tutkielmassa.

Luvussa 3 esitellään tämän tutkimuksen tutkimusmenetelmät ja haastattelu aineistonkeruumenetelmänä. Lisäksi käsitellään haastattelusuunnitelma, kuvataan haastattelujen toteutustapa ja valmistellaan niiden analysointi.

Luvussa 4 käydään läpi haastattelun tulokset teemoittelun avulla. Teemoittelussa etsitään aineistosta yhteneväisiä sekä eroavia tekijöitä, jotka muodostavat tutkielman tulokset. Näiden teemojen pohjalta analysoidaan aineistoa syvällisemmin ja tarkastellaan, miten tulokset vastaavat tutkimuskysymyksiin. Teemoittelun avulla pyritään myös tunnistamaan mahdollisia uusia ilmiöitä, jotka eivät olleet ennakkoon tiedossa.

Luvussa 5 kerrotaan tutkielman johtopäätökset, pohditaan tutkielman onnistumista sekä kerrotaan jatkotutkimusmahdollisuuksista. Johtopäätöksissä esitetään keskeiset löydökset ja niiden merkitys tutkittavan ilmiön kannalta. Lisäksi arvioidaan tutkimuksen rajoituksia ja pohditaan, miten ne ovat saattaneet vaikuttaa tuloksiin. Jatkotutkimusmahdollisuuksien osalta esitetään uusia tutkimuskysymyksiä ja suosituksia tuleville tutkimuksille.

Luvussa 6 esitellään tutkielman yhteenveto. Yhteenveto kokoaa yhteen tutkielman keskeisimmät havainnot ja päätelmät, tarjoten lukijalle selkeän käsityksen tutkimuksen kokonaisuudesta. Tässä luvussa korostetaan myös tutkimuksen käytännön merkitystä ja sen mahdollisia vaikutuksia alaan liittyvissä käytännöissä ja päätöksenteossa.

2 Kirjallisuuskatsaus

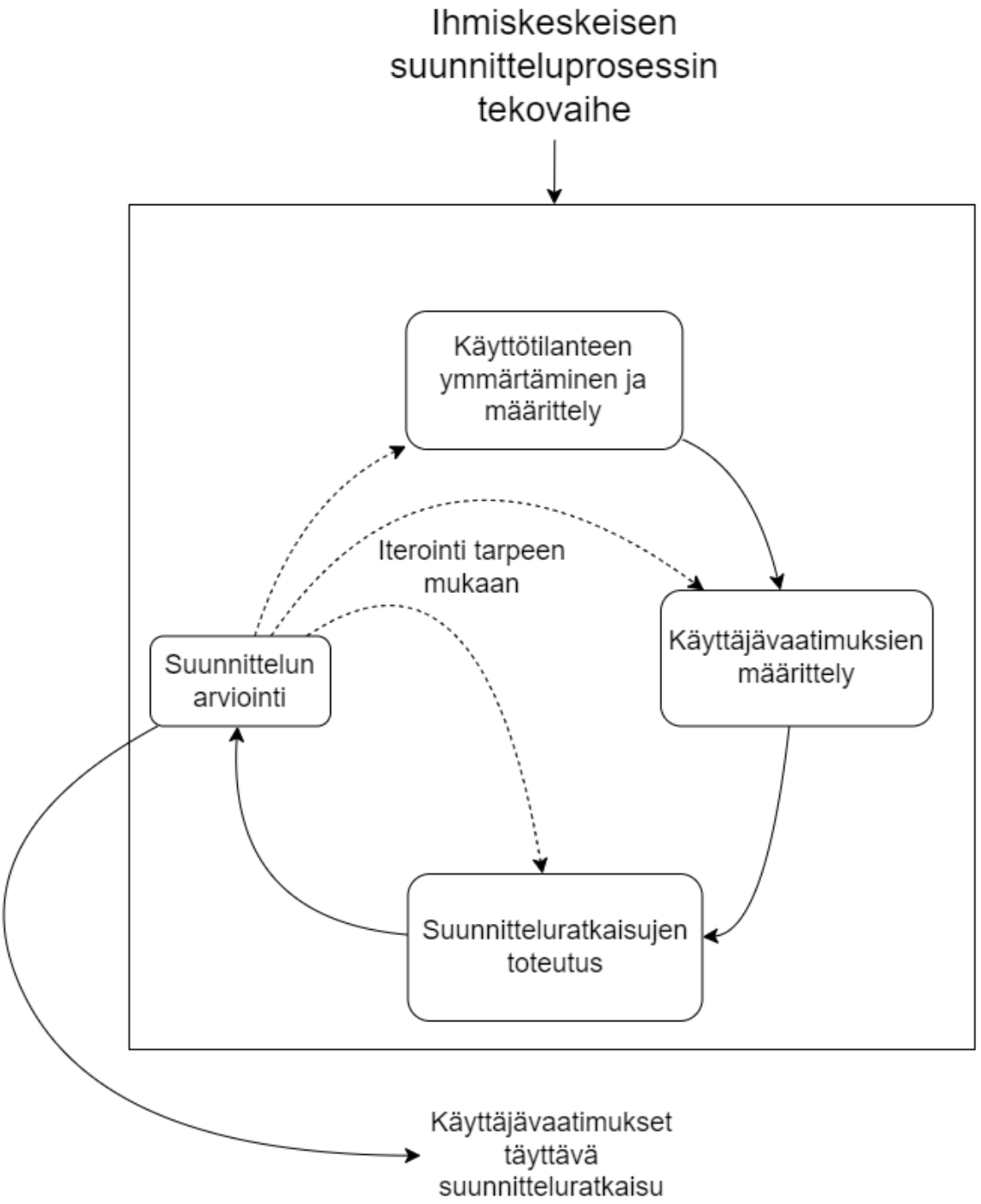
Tässä luvussa käydään läpi tärkeitä termejä tutkielman aiheeseen liittyen. Ensimmäisenä määritellään käyttäjäkeskeinen suunnittelu sekä tuodaan ilmi sen periaatteita. Lisäksi luvussa esitellään menetelmiä, joita käytetään käyttäjäkeskeisen suunnittelun toteuttamiseen ketterässä ohjelmistokehityksessä. Sen jälkeen määritellään ketterä ohjelmistokehitys ja sen synty, jonka jälkeen avataan Scrum-menetelmää. Lopuksi määritellään full stack -suunnittelijan rooli ja mitä sillä tarkoitetaan tässä työssä.

2.1 Käyttäjäkeskeisen suunnittelun määritelmä

ISO-standardi 9241-210:2019 kuvailee käyttäjäkeskeisen suunnittelun toimintoja, suosituksia ja periaatteita, joiden on tarkoitus edistää ihmisen ja järjestelmän välistä vuorovaikutusta. Näin järjestelmien käytettävyyttä voidaan parantaa. Standardin mukaan käyttökelpoisia ja hyödyllisiä järjestelmiä saadaan kehitettyä lähestymällä vuorovaikutteisia järjestelmiä ihmiskeskeisellä suunnittelulla, joka keskittyy käyttäjiin, heidän tarpeisiinsa ja vaatimuksiinsa. Lisäksi se soveltaa käytettävyyteen, ergonomiaan sekä inhimillisiin tekijöihin liittyvää tietoa ja tekniikoita. [SFS, 2021]

Käyttäjäkeskeisen suunnittelun periaatteet on määritelty Nadikattun [2016] mukaan seuraavanlaisesti:

- Käyttäjien sekä heidän odotusten ja tarpeiden ymmärtäminen
- Käyttäjien ja ympäristöjen ymmärtäminen
- Käyttäjät pidettävä mukana koko suunnittelu- ja kehitysprosessin ajan
- Suunnittelu on iteratiivista
- Suunnitelmat ovat monitieteellisiä



Kuva 1. Käyttäjäkeskeinen suunnitteluprosessi [SFS, 2021].

Kuva 1 esittää käyttäjäkeskeisen suunnittelun prosessin vaiheet. Prosessi sisältää neljä päävaihetta: käyttötilanteen ymmärtäminen ja määrittely, käyttäjävaatimusten määrittely, suunnitteluratkaisujen luominen ja suunnittelun arviointi. Ensimmäisessä vaiheessa, käyttötilanteen ymmärtämisessä ja määrittelyssä, keskitytään järjestelmän käyttötilanteen syvälliseen ymmärtämiseen. Tässä vaiheessa tarkastellaan käyttäjiä, heidän suoritettavia

tehtäviä sekä ympäröivää byrokraattista ja fyysistä ympäristöä. Toisessa vaiheessa, käyttäjävaatimusten määrittelyssä, havainnoidaan käyttäjien tarpeet ja määritellään tuotteen tai järjestelmän toiminnalliset ja muut vaatimukset. Tämä vaihe sisältää myös käyttäjävaatimusten määrittelyn suhteutettuna järjestelmän liiketoiminnan tavoitteisiin sekä käyttötilanteeseen. Suunnitteluratkaisujen toteutus vaiheessa tehdään useampaa eri asiaa [SFS, 2021]:

1. Käyttäjän ja järjestelmän vuorovaikutussuunnittelu
2. Käyttäjätehtävien määrittely
3. Käyttöliittymän suunnittelu
4. Suunnitteluratkaisujen konkretisointi

Se, että kuinka käyttäjät voivat suorittaa tehtäviä järjestelmän avulla, pyritään päättämään vuorovaikutussuunnittelun aikana. Suunnitteluratkaisuja tehdessä on huomioitava, etteivät suunnittelun eri vaiheet ole juuri tässä järjestyksessä, vaan ne hyödyntävät toistensa tuloksia. Viimeisessä vaiheessa arvioidaan suunnittelun lopputulosta ja suunnittelua iteroidaan, kunnes suunnitteluratkaisu täyttää käyttäjävaatimukset. [SFS, 2021]

2.2 Käyttäjäkeskeinen suunnittelu ketterässä kehityksessä

Tässä kohdassa tarkastellaan käyttäjäkeskeistä suunnittelua ketterässä ohjelmistokehityksessä sekä selvitetään, mitä eri rooleja käyttäjäkokemussuunnittelijalla voi olla ja kuinka tehokasta työskentely on muiden ketterän kehityksen roolien kanssa.

Ketterä käyttäjäkokemussuunnittelu (engl. Agile User eXperience Design) on trendikäs ja ajankohtainen aihe ohjelmistokehityksessä, sillä elämme kokemusvetoisessa maailmassa. Kun ohjelmistotuote pääsee loppukäyttäjien saataville, tuotteen käyttäjäkokemus määrää tuotteen onnistumisen tai epäonnistumisen markkinoilla [Silva ym., 2018]. Kahden ensimmäisen vuosikymmenen aikana käyttäjäkeskeisen suunnittelun parissa strukturoidut metodologiat tarjosivat selkeän prosessin, joka mahdollisti käyttäjäkeskeisten suunnittelumenetelmien integroimisen ilman merkittäviä integraatiohaasteita, jotka eivät pääosin johtuneet prosessirakenteista. Tämä tilanne muuttui ketterän ohjelmistokehityksen lähestymistavan yleistyessä, mikä eroaa olennaisesti strukturoidusta kehityksestä [Cockton ym., 2016]. Ketterän kehityksen tarkoitus on keskittyä siihen, kuinka hyödyllisiä ohjelmistoja voidaan kehittää, käyttäjäkokemussuunnittelun tarkoituksena on varmistaa loppukäyttäjien tarpeiden ja tavoitteiden huomioiminen tuotekehityksessä. Molemmat menetelmät ovat ristiriidassa toistensa kanssa. Käyttäjäkokemussuunnittelun lähestymistavat pyrkivät lisäämään ennakkoanalyysi- ja suunnittelutyötä, kun taas ketterät menetelmät yrittävät vähentää sekä rajoittaa niiden tarvetta. [Silva ym., 2018]

Vaikkakin yhteisöissä on todettu, etteivät ketterä kehitys ja käyttäjäkokemussuunnittelu voi olla kaksi erillistä prosessia ja että käyttäjäkokemuksen toiminnot on integroitava ketterään kehitykseen [Silva ym., 2018], useita haasteita on silti edelleen ratkaisematta [Cockton ym., 2016; Argumanis ym., 2020]. Haasteita ovat muun muassa [Argumanis ym., 2020; Curcio ym., 2019]:

- Riittämätön viestintä käyttäjäkokemussuunnittelijoiden ja kehittäjien välillä
- Asiakkaat yrittävät edustaa loppukäyttäjiä tietämättä heidän todellisia tarpeitaan
- Käyttäjäkokemukselle on annettu liian vähän painoarvoa ketterässä kehityksessä
- Ajan puute suorittaa ennakkosuunnittelua sekä testejä oikeiden käyttäjien kanssa
- Valtataistelu kehittäjien ja käyttäjäkokemussuunnittelijoiden välillä
- Koko UX-projektin näkemyksen puute
- Priorisoinnin vaikeus UX tehtävissä
- Dokumentoinnin puute

Lähestymistapoja käyttäjäkeskeisen suunnittelun integroimiseksi ketterään kehitykseen on keksitty muutamia. Yhtenä tapana on kevyiden ja inkrementaalisien prosessien lisääminen ketterään kehitykseen sekä yksinkertaistettua ääneen ajattelua testiparikäyttäjien kanssa yhdistettynä heuristiseen arviointiin. [Teka ym., 2018] Toisena tapana integroida ketterä kehitys käyttäjäkeskeiseen suunnitteluun on olla yksi iteraatio edellä ketterää kehitystä suunnittelemalla tarvittava käyttöliittymä seuraavalle iteraatiokierrokselle [Sy, 2007]. Ennen ohjelmoinnin aloittamista, asiakkaan data kerätään, suunnittelu tehdään aina yhtä sprinttiä eteenpäin sekä kyseisen toiminnon käyttäjätestaus suoritetaan seuraavassa sprintissä. Silva ym. [2011] ehdottavat systemaattisessa katsauksessaan kolmea asiaa tehtäväksi:

1. Pienen suunnittelun tekeminen etukäteen
2. Keskitytään kehittäjien sekä käytettävyyssuunnittelijoiden tiiviiseen yhteistyöhön
3. Käytettävyyssuunnittelijat työskentelevät yhtä sprinttiä kehittäjiä edellä

2.2.1 Käyttäjäkokemussuunnittelijan roolit ketterässä kehityksessä

Käyttäjäkokemussuunnittelijan rooleja ketterässä ohjelmistokehityksessä on tutkittu Silva ym. [2013] puolesta, saadaksemme paremman käsityksen roolien merkityksistä.
Myös aikaisemmin käyttäjäkokemussuunnittelijan roolin integroimista ketterään ohjelmistokehitykseen on tutkittu [Hussain, 2009], mutta tutkimuksia ei ole käsitelty asianmukaisesti, sillä niissä ei kerrota tarkemmin käyttäjäkokemussuunnittelijan roolista kehityk-

sessä [Silva ym., 2013]. Silva ym. [2013], saivat myös tulokseksi, että käyttäjäkokemussuunnittelijan rooli voi muuttua useasti projektin aikana sekä se voi olla sidonnainen kontekstiin paljon.

Silvan ym. [2013] tutkimuksessa kerrotaan kolme käyttäjäkokemussuunnittelijan roolia:

- Vuorovaikutussuunnittelija (engl. interaction designer)
- Käyttäjäkokemussuunnittelija (engl. user experience designer / UX-designer)
- Käyttöliittymäkehittäjä (engl. user interface developer / UI-developer)

Vuorovaikutuksen suunnittelu sekä arviointi ovat vuorovaikutussuunnittelijan vastuulla. Tutkimus osoitti, että projektin alkuvaiheessa tehdyn tutkimuksen avulla käyttäjäkokemussuunnittelija voi rakentaa oman käyttäjäkokemus kehitysjonon (engl. UX Backlog), jolloin vuorovaikutussuunnittelija voi suunnitella tai jopa luoda prototyypin yhden iteroinnin ennen kehitystiimiä. Jokainen vuorovaikutussuunnittelijan rautalankamalli, design tai luonnos voi auttaa käyttäjäkokemussuunnittelijaa aloittamaan arviointiprosessin, joiden avulla voidaan määrittää mitä rakennetaan sekä välttää turha uudelleen työstäminen. [Silva ym., 2013]

Käyttäjäkokemussuunnittelijan tarkoituksena on ymmärtää käyttäjien tarpeet. Tutkimuksen mukaan, eräässä yrityksessä oli tarkoitus kerätä dataa suorituskyvyn mittaamiseen, persoonien määrittelyyn sekä kohderyhmien johtamiseen. Selvisi, että yrityksen markkinointitiimi oli kerännyt dataa käyttäjiltä käyttäjäkokemustiimille, mutta tämä data koski ennemmin käyttäjien toiveita kuin heidän tarpeitansa. Tämän vuoksi on erittäin tärkeää, että käyttäjäkokemussuunnittelija suorittaa kyseisen tutkimuksen, sillä he ovat saaneet siihen koulutuksen. [Silva ym., 2013]

Käyttöliittymäkehittäjät työskentelevät sekä suunnittelijoiden että kehittäjien kanssa. Heidän vastuullaan on käyttöliittymän graafinen toteutus sekä siihen liittyvistä elementeistä. Tutkimuksen perusteella käyttöliittymäkehittäjistä vain harva on suunnittelijoita, sillä suunnittelijoilla ei ole kokemusta käyttöliittymäkehityksestä ja kokevat sen hankalaksi. Kuitenkin joissain tapauksissa yrityksessä saattaa olla käyttäjäkokemussuunnittelijoita, jotka osaavat toteuttaa prototyyppejä ohjelmoimalla. [Silva ym., 2013] Kaikkien kolmen eri roolin vastuut ja taidot esitetään taulukossa 1.

Rooli	Vuorovaikutussuunnittelija	Käyttäjäkokemussuunnittelija	Käyttöliittymäkehittäjä
Vastuu	Vastuussa käyttäjävuorovaikutuksen suunnittelusta ja arvioinnista tuotteiden tai palvelujen yhteydessä sekä prototyypeissä, että kehitetyssä järjestelmässä.	Käyttäjäymmärrys.	Graafisen käyttöliittymän (engl. Graphical User Interface, GUI) kehittäminen ja graafisten elementtien suunnittelu.
Taidot	Vuorovaikutussuunnittelu, nopea prototyypitys, käyttäjäkokemussuunnittelu, tuotesuunnittelu, sissitestauksen luonnostelu, käytettävyystestaus, ideointi, yhteistyösuunnittelu, prosessivirrat, tietoarkkitehtuuri, palvelusuunnittelu, suunnitteluajattelu.	Käyttäjätutkimus, etnografiset tutkimukset, käyttäjäkokemuksen suunnittelu, käyttäjäprofilointi, ideointi, kilpailija-analyysi, suunnitteluajattelu, asiakaspolun kartoitus.	Nopea prototyypitys, yhteistoiminnallinen suunnittelu, informaatioarkkitehtuuri, visuaalinen suunnittelu, graafisen käyttöliittymän suunnittelu, palvelumuotoilu, suunnitteluajattelu.

Taulukko 1. Käyttäjäkokemussuunnittelijan eri rooleja ketterässä ohjelmistokehityksessä

2.2.2 Yhteistyön tehokkuus muiden ketterän kehityksen roolien kanssa

Kuusisen [2015] tutkimuksessa tutkittiin kuutta erilaista projektia viiden eri yrityksen välillä yhden julkaisusyklin aikana ja tutkimuksessa saatiin tunnistettua kolme erilaista yhteistyötyyppiä:

1. Minimaalinen yhteistyö
2. Tuoteomistajan ja käyttäjäkokemussuunnittelijoiden läheinen yhteistyö
3. Sovelluskehittäjien ja käyttäjäkokemussuunnittelijoiden läheinen yhteistyö

Näistä parhaimmaksi ja halutuimmaksi yhteistyöksi koettiin kolmas yhteistyötyyppi eli sovelluskehittäjien ja käyttäjäkokemussuunnittelijoiden välinen yhteistyö. Jotta tämä toimintatapa toimisi oikein, on sovelluskehittäjien sekä käyttäjäkokemussuunnittelijoiden oltava kokeneita ja halukkaita työskentelemään yhdessä. Lisäksi suunnittelutehtävien allokointi voi olla haastavaa käyttäjäkokemussuunnittelijoille [Salah ym., 2014]. Tiimit,

jotka ovat kokemattomampia, voisivat hyötyä paremmin kakkostyypistä eli tuoteomistajan ja käyttäjäkokemussuunnittelijoiden välisestä yhteistyöstä, sillä tuoteomista pitää kehityksen hallinnassa koko kehitysprosessin ajan. Tämä tapa toimii myös silloin, jos projektin laajuus ei ole tiedossa tai on epäselvä. Tapa kaksi toimii kuitenkin vain silloin, jos tuoteomistaja arvostaa ja ymmärtää käyttäjäkokemustyötä. [Kuusinen, 2015]

Ensimmäistä tapaa eli sitä etteivät tuoteomistaja eikä kehittäjät ole missään yhteistyössä käyttäjäkokemussuunnittelijan kanssa suositellaan vältettävän, sillä siitä syntyy yleensä enemmän ongelmia kuin hyötyä. [Kuusinen ym., 2012] Tätä on myös tutkittu aikaisemmin Ferreira ym. toimesta [2010]

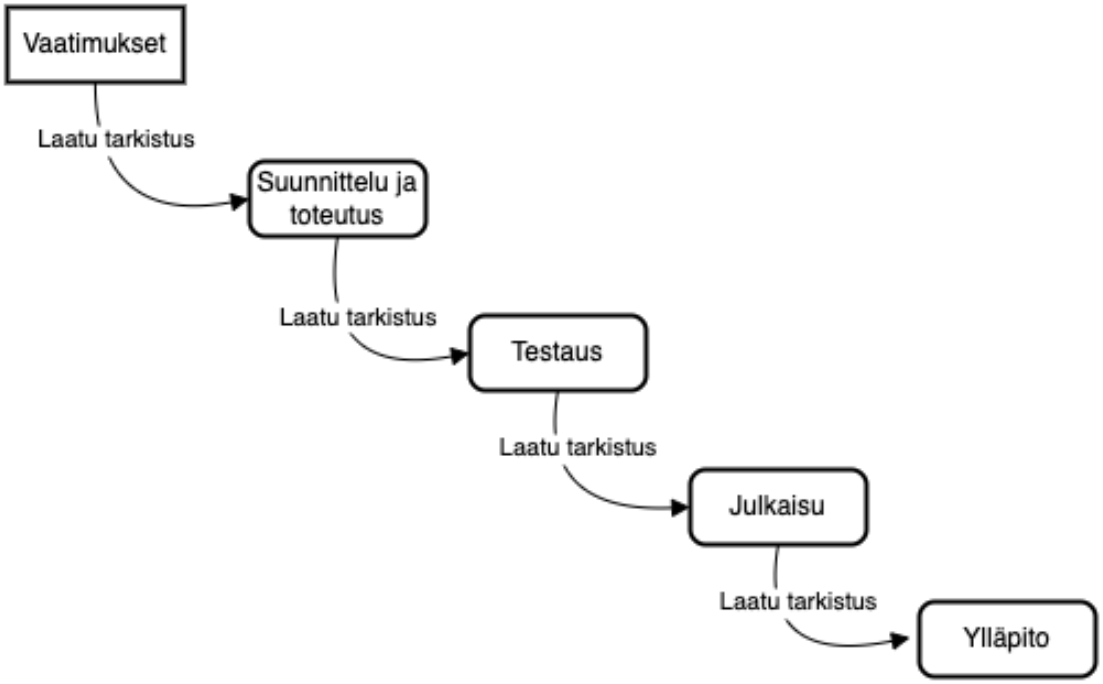
Ketterät menetelmät kuten Scrum ei lähtökohtaisesti tue käyttäjäkokemussuunnittelua tarpeeksi tehokkaasti vaan ne ovat tarkoitettu palvelemaan asiakkaita mahdollisimman hyvin. Tämän takia käyttäjäkokemussuunnittelijat saattavat kokea olevansa lisäosana ketterässä kehityksessä, vaikka he ovatkin tärkeitä ohjelmistoprojektien menestykselle [Schmitz ym., 2019].

2.3 Ketterä ohjelmistokehitys

Tässä kappaleessa syvennymme tarkemmin ketterien menetelmien, erityisesti Scrumin, perusperiaatteisiin ja miten ne eroavat perinteisestä vesiputousmallista. Näin saamme kokonaisvaltaisemman kuvan siitä, miksi ketterä kehitys on noussut suosioon ja miten se mahdollistaa käyttäjäkeskeisen suunnittelun tehokkaan toteutuksen.

Ketterät menetelmät ovat saavuttaneet laajan hyväksynnän maailmalla [Dingsør ym., 2012], joista Scrum nousi esiin suosituimpana menetelmänä [VersionOne, 2020]. Tässä luvussa esitellään ketterän ohjelmistokehityksen periaatteita, keskittyen lähinnä Scrum-menetelmään tunnettavuutensa vuoksi. Seuraavaksi vertaillaan ketterää mallia vesiputousmalliin, jota pidetään ketterän kehityksen vastakohtana ja josta ketterä kehitysmalli on saanut alkunsa, sillä ohjelmistokehittäjät halusivat luoda vaihtoehtoisen projektinhallintamallin perinteisen mallin tilalle [Beck ym., 2001].

Vesiputousmallissa, joka on perinteinen projektinhallintamalli, asiakas määrittelee odotetut tulokset selkeästi projektin alkuvaiheessa. Projekti suunnitellaan kokonaisuudessaan alusta loppuun asti, mukaan lukien työpaketit, vastuut ja aikataulut, jotta se voidaan toteuttaa tavoitteellisesti ja suunnitelmallisesti. Vesiputousmallissa tärkeintä on noudattaa alkuperäistä suunnitelmaa mahdollisimman tarkasti, mikä luo vakautta ja rakennetta. Tämä lähestymistapa tarjoaa ennakoitavat resurssit ja dokumentoidun suunnitelman. [Thesing ym., 2021]



Kuva 2. Vesiputousmallin vaiheet perustuen Petersenin ja muiden [2009] mukaan.

Vesiputousmallin organisoitu ja looginen toiminta aiheuttaa kuitenkin myös ongelmia. Niitä ovat muun muassa vesiputousmallin jäykkyys eli kehitysprosessin aikana ei voida tehdä muutoksia juuri ollenkaan, jolloin testausvaiheessa ohjelmiston laatu voi olla arvaamaton. Tämä taas johtaa siihen, että ohjelmistoa joudutaan muokkaamaan uudestaan myöhemmässä vaiheessa, jolloin ylimääräisen työn tekeminen on välttämätöntä. [Sommerville, 2011]

Ongelmat vesiputousmallissa saivat ohjelmistokehittäjät miettimään vaihtoehtoista kehitysmallia, jolla vesiputousmallin ongelmat voitaisiin välttää. Vuonna 2001 ketterä kehitys sai alkunsa, kun ohjelmistokehittäjien ryhmä kokoontui kertomaan omia näkemyksiään ohjelmistokehityksestä. Kokoontumisen seurauksena syntyi Ketterän ohjelmistokehityksen julistus (engl. Agile Manifesto). Julistuksessa kiteytetään ketterän kehityksen 12 periaatetta, joita ovat [Beck ym., 2001]:

1. Tarjota asiakkaille heidän tarpeitaan vastaavia ohjelmistoverisoita varhaisessa vaiheessa ja säännöllisin väliajoin
2. Muuttuvien vaatimusten vastaanotto myös myöhäisessä vaiheessa
3. Toimiva versio ohjelmasta toimitettuna asiakkaalle lyhyin väliajoin
4. Liiketoiminnan edustajien ja ohjelmistokehittäjien päivittäinen yhteistyö koko projektin ajan
5. Projektien rakentaminen motivoituneiden yksilöiden ympärille

6. Kasvotusten käytävä kommunikointi, jotta tieto saadaan tehokkaimmin sekä toimivimmin kehitystiimille
7. Toimiva ohjelmisto on edistymisen ensisijainen mittari
8. Ketterät menetelmät mahdollistavat pitkäkestoisen kehityksen
9. Teknisen laadun sekä ohjelmiston jatkuva huomiointi
10. Yksinkertaisuus
11. Itseorganisoituvat tiimit
12. Tiimin säännöllinen tarkastelu

Julistuksessa luetellaan myös ketterässä kehityksessä arvostettuja arvoja [Beck ym., 2001]:

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

Julistus kattaa useita kehitysmenetelmiä, muun muassa Scrum-menetelmän, mutta ei ole itsessään kehitysmenetelmä tai prosessi.

Ketterä kehitys nojaa vahvasti itseorganisoituviin ja monitaitoisiin tiimeihin, jotka kykenevät tekemään päätöksiä projektikohtaisella tasolla itsenäisesti sekä iteroimaan nopeasti suoraan asiakkaan tai sidosryhmien kanssa. Näin voidaan mahdollistaa valmis tuote nopeammin ilman, että aikaa olisi käytetty liikaa vaatimusten määrittelyyn. Ketterät menetelmät, kuten Scrum tai Kanban, eivät painota kattavaa ennakkosuunnittelua ja lineaarista, tarkkaa etenemistä. Sen sijaan ne mahdollistavat ratkaisun kehittämisen vaiheittain ja tiiviin yhteistyön asiakkaan kanssa lyhyissä sykleissä. Tämä lähestymistapa perustuu siihen, että asiakas tai projektin lopullinen käyttäjä määrittelee yleiset vaatimukset, mutta yksityiskohdat hahmottuvat vasta projektin edetessä. [Thesing ym., 2021]

2.4 Scrum-menetelmä

Alkuperäisesti Scrum sopi parhaiten pienille ohjelmistokehittäjäryhmille, jotka toimivat samasta sijainnista työskennellen tiiviisti asiakkaan kanssa koko kehitysprosessin ajan. Nykyään Scrumia käytetään laajasti aina ohjelmistokehitysprojekteista kokonaisten yritysten johtamiseen. [Hron ym., 2022]

Alun perin vuonna 1997 Schwaber esitteli Scrum-metodologian yksinkertaiseksi kehitysmenetelmäksi, joka perustuu iteratiivisiin ja inkrementaalisiin kehitysperiaatteisiin.

Scrum sisältää erilaisia seremonioita tai toiselta nimeltään rituaaleja tai menettelyjä sekä esineitä ja rooleja, ja sen luojat ovat jatkuvasti päivittäneet sitä aktiivisesti. Marraskuussa 2020 on julkaistu viimeisin versio virallisesta Scrum-oppaasta.

2.4.1 Scrumin arvot

Scrum-oppaan mukaan Scrum-tiimin jäsenet ovat sitoutuneita yhteisten tavoitteiden saavuttamiseen ja toistensa tukemiseen. He omistautuvat ensisijaisesti sprinttityöhön, jotta voivat tehokkaimmin edistää tiimin päämääriä. Avoin kommunikaatio ja haasteiden jakaminen ovat keskeisiä niin tiimin sisällä kuin ulkopuolisten sidosryhmienkin kanssa. Jäsenten keskinäinen kunnioitus ja arvostus ovat perustana yhteistyölle, ja he myös odottavat vastavuoroista kunnioitusta muilta yhteistyökumppaneiltaan. Scrum-tiimin jäsenillä on rohkeutta tarttua haastaviin tehtäviin ja työskennellä niiden ratkaisemiseksi parhaaksi katsomallaan tavalla. Näin ollen Scrumin onnistunut käyttökokemus nojaa vahvasti viiteen eri arvoon; sitoutuminen, keskittyminen, avoimuus, kunnioitus ja rohkeus. [Schwaber ym., 2020]

Nämä arvot toimivat ohjeistuksena Scrum-tiimin työlle, toiminnalle ja käyttäytymiselle. Kaikkien päätösten ja Scrumin käytön tulisi vahvistaa näitä arvoja eikä heikentää niitä. Scrum-tiimin jäsenet tutkivat ja oppivat näitä arvoja työskennellessään Scrumin tapahtumien ja tulosten parissa. Kun tiimin jäsenet ja heidän yhteistyökumppaninsa ilmaisevat näitä arvoja, Scrumin empiirinen perusta - läpinäkyvyys, tarkastelu ja mukauttaminen - saavuttaa täyden potentiaalinsa ja rakentaa luottamusta. [Schwaber ym., 2020]

2.4.2 Scrum-tiimi, sen tehtävät ja roolit

Scrum-tiimissä ei ole hierarkia-asetelmaa vaan se on ammattilaisten yksikkö, joka on monialainen ja itseohjautuva sekä riittävän pieni ollakseen jouheva ja tarpeeksi suuri valmistaakseen riittävän työkokonaisuuden yhden sprintin aikana. Tyypillisesti Scrum-tiimissä on kuitenkin enintään kymmenen jäsentä, sillä pienemmät tiimit kommunikoivat keskenään paremmin sekä tuottavat tuotteelle enemmän arvoa. Tiimin vastuualueeseen kuuluu kaikki tekeminen mikä liittyy tuotteeseen. Näitä tehtäviä voivat olla muun muassa:

- Laadun varmistaminen
- Tuotanto
- Ylläpito
- Sidosryhmäviestintä
- Kehitystyö
- Kokeilu
- Tutkimusten laatiminen

Scrum-oppaassa on määritelty kolme vastuualuetta tiimin sisällä, jotka ovat Scrum Master, tuoteomistaja (engl. product owner) sekä kehittäjät (engl. development team). Scrum Masterin vastuu on taata Scrumin toteutuksesta oppaan mukaisesti sekä auttaa ymmärtämään tiimiä sekä koko organisaatiota Scrumin käytännöt ja teoria. Scrum Master palvelee näin kolmea eri tahoa, tiimiä, tuoteomistajaa sekä organisaatiota. Scrum Master auttaa tiimiä toimimaan itsenäisesti ja monialaisesti sekä fokusoimaan loppukäyttäjille arvokkaiden inkrementtien toteuttamiseen. Tämän lisäksi Scrum Master varmistaa, että tiimin etenemiselle ei ole esteitä ja että kaikki Scrumille tärkeät tapahtumat pidetään, ovat tuottavia, pysyvät aikamääreissä ja toteutuvat positiivisessa hengessä. [Schwaber ym., 2020]

Tuoteomistaja on yksittäinen henkilö, jonka tavoitteena on saada Scrum-tiimi tuottamaan mahdollisimman suuren arvon tekemälleen työlle. Kehitysjonon hallinta on tuoteomistajan päävastuualue ja siihen kuuluu kehitysjonon valmistelu, järjestely, viestintä sekä sen läpinäkyvyyden, ymmärrettävyyden ja saatavuuden varmistaminen. Tuoteomistaja myös edustaa eri sidosryhmien edustajia ja heidän tarpeitaan tuotteen kehitysjonon avulla. Koko organisaation tulee arvostaa tuoteomistajien päätöksiään, jotta tuoteomistajat voivat onnistua tehtävässään. [Schwaber ym., 2020]

Kehittäjät vastaavat sprintin kehitysjonosta ja ovat sitoutuneita tuottamaan kaiken tarvittavan käyttökelpoisen inkrementin jokaisessa sprintissä. Näin ollen heillä täytyy olla usein olla monipuoliset taidot, jotka voivat vaihdella toimialojen mukaan. [Schwaber ym., 2020] Kehitystiimi voi pitää sisällään esimerkiksi suunnittelijoita, kehittäjiä, testaajia, arkkitehtejä ja dokumentoijia. [Sutherland, 2015]

2.4.3 Scrum prosessi

Scrumin prosessi on suunniteltu läpinäkyviksi, jotta toimintaa voidaan tarkastella ja tarpeen mukaan muokata. Sprintti on Scrumin ainoa tapahtuma, joka näin ollen sisältää kaikki muut tapahtumat. Niitä ovat sprintin suunnittelu, päivittäispalaveri (engl. daily stand-up), sprintin katselmointi ja retrospektiivi. Yhden sprintin kesto on aina vakio ja se kestää enintään kuukauden. Sen aikana ei tehdä muutoksia, jotka vaarantavat sprintin tavoitteen sekä laadun tulee säilyä korkeana. Uusi sprintti alkaa heti edellisen sprintin päätyttyä. Tuotteen kehitysjonoa jalostetaan tarpeen mukaan ja sprintin sisältöä voidaan tarkentaa sekä siitä voidaan neuvotella tuoteomistajan kanssa yhteisymmärryksessä. [Schwaber ym., 2020]

Sprintin suunnittelun tavoitteena on määritellä sprintin tavoite ja sprintille valitut kehityskohdat. Suunnitelma niiden toteuttamiseksi muodostavat sprintin kehitysjonon. Suunnittelussa vastataan kolmeen kysymykseen:

1. Miksi tämä sprintti on arvokas?
2. Mitä tässä sprintissä voidaan saada valmiiksi?
3. Miten valittu työ saadaan valmiiksi?

Aluksi tuoteomistaja ehdottaa tiimille, että miten tuotteen hyödyllisyyttä sekä arvoa voisi nostaa sprintin aikana, jonka jälkeen Scrum-tiimi yhdessä määrittelee tavoitteen, joka kertoo, miksi tämä sprintti on arvokas sidosryhmille. Sprintin suunnitteluun käytetään tavallisesti enintään kahdeksan tuntia aikaa, jos sprintti on neljän viikon mittainen. [Schwaber ym., 2020]

Päivittäispalavereissa tarkastellaan, kuinka kehitystiimi on edistynyt kohti sprintin päämäärää ja tarvittaessa sprintin kehitysjonoa muokataan sopivaksi. Jos tuoteomistaja tai Scrum Master toteuttavat kehitysjonon tehtäviä, he osallistuvat päivittäispalaveriin kehittäjän roolissa. Päivittäispalaverin pituus on 15 minuuttia, joka pidetään aina samaan aikaan samassa paikassa jokaisena sprintin työskentely päivänä, jotta työskentely olisi mahdollisimman yksinkertaista. Päivittäispalaverien on tarkoitus parantaa tiimin vuorovaikutusta sekä nostaa esiin ongelmankohtia ja lisätä kyvykkyyttä nopeaan päätöksentekoon, jolloin tarvetta muille kokouksille ei välttämättä ole. [Schwaber ym., 2020]

Scrum-tiimi ja sidosryhmien edustajat tarkistavat sprintissä saavutetut toiminnot sekä muutokset sprintin katselmoinnin aikana, joka pidetään sprintin lopussa. Tämän perusteella he voivat keskustella tulevista mahdollisista muutostarpeista sekä siitä, että mitä kannattaisi tehdä seuraavaksi. Katselmoinnin tarkoituksena on kerätä palautetta ja kasvattaa kommunikaatiota tiimin ja sidosryhmien välillä. [Schwaber ym., 2020; Sutherland, 2015]

Sprintin päättävä tapahtuma on sprintin retrospektiivi, jonka tarkoituksena on mahdollistaa omien toimintatapojen reflektointi ja luomaan mahdollisesti parempia toimintatapoja seuraavaan sprinttiin. Scrum-tiimi arvioi kuluneen sprintin kulkua ihmisten, yhteistyön, prosessien, ja työkalujen sekä valmiin määritelmän näkökulmasta, tarkasteltavat asiat vaihtelevat työn luonteen mukaan. Mahdolliset harhaanjohtavat oletukset tunnistetaan ja niiden alkuperä selvitetään. Lisäksi tiimi käy läpi, mikä sujui hyvin sprintin aikana, millaisia ongelmia ilmeni, ja miten nämä ongelmat ratkaistiin tai jäivät ratkaisematta. [Schwaber ym., 2020; Sutherland, 2015]

2.5 Full stack -suunnittelijan määritelmä

Aikaisemmissa kohdissa on käsitelty käyttäjäkeskeistä suunnittelua ketterissä menetelmissä sekä ketterän ohjelmistokehityksen perusperiaatteita, erityisesti Scrumin näkökulmasta. Tässä kontekstissa on tärkeää ymmärtää, kuinka eri lähestymistavat ja roolit, kuten "full stack -suunnittelija" (engl. full stack -designer), integroituvat ketteriin kehitysprosesseihin. Full stack -suunnittelija edustaa laaja-alaisempaa lähestymistapaa, jossa yhdistyy käyttäjäkeskeinen suunnittelu ja tekninen osaaminen, mikä on olennaista ketterässä ympäristössä, jossa tiimit toimivat tiiviissä yhteistyössä ja moniosaaminen on eduksi.

”Full stack -suunnittelija” ei ole vakiintunut termi, vaikkakin onkin muodostunut yhteinen käsitys, että mitä sillä tarkoitetaan [Abid ym., 2023, Kulchinskiy, 2021, Larocca, 2023]. Tässä luvussa määritellään, mitä full stack -suunnittelijalla tarkoitetaan tässä tutkielmassa ja mitä full stack -suunnittelu itsessään tarkoittaa.

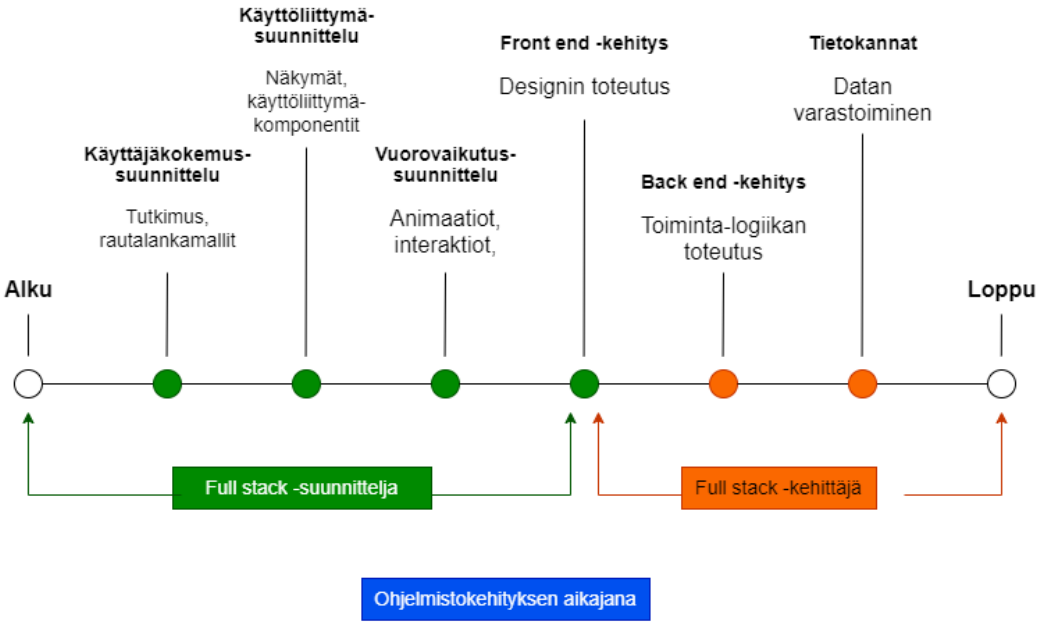
Yritykset, jotka tuottavat prototyyppejä suunnitelluille käyttöliittymille, suosivat yleisesti digitaalisia prototyyppejä. Prototyyppien luontiin käytetään suosittuja käyttöliittymäsuunnittelutyökaluja kuten Figmaa tai Adobe XD:tä. Digitaaliset prototyypit mahdollistavat suunnitelmalle realistisen vaikutelman vuorovaikutuksineen ja käyttöliittymineen, mutta suunnitelman muuttaminen koodiksi tuottaa usein ongelmia. Esimerkiksi jotkin käyttöliittymäelementit eivät ole yhteensopivia tai vuorovaikutusten tuottaminen suunnitelman mukaisesti on mahdotonta, jolloin prototyypin luominen täytyy aloittaa alusta. [Canziba, 2018]

Toinen, mutta monimutkaisempi tapa luoda prototyyppejä on koodata niitä. Prototyyppien koodaus mahdollistaa lähes täysin realistiset tuotteet ja interaktiot niiden kanssa. Full stack -suunnittelu on saanut viime vuosina paljon huomiota ja monet yritykset rakentavat omia viitekehyksiään (engl. frameworks), jotta he pystyvät olemaan ajan tasalla heidän tuotteidensa käyttöliittymistä. Full stack -suunnittelija antaa animaatiot, säännökset, vuorovaikutukseen vaikuttavat tekijät, käyttöliittymäelementit sekä graafiset materiaalit yrityksen tuotteille käytettäviksi. Useimmille yrityksille prototyyppien koodaus on iso investointi, sillä henkilön palkkaaminen, joka osaa sekä suunnitella että koodata, on kallista. Canziba, 2018]

Termi ”full stack” kääntyy englannista ”täysi pino”, joka usein yhdistetään full stack -kehittäjään. Full stack -kehittäjä hallitsee web-kehityksen eri teknologiakerrokset front end:n ja back end:n. Kun puhutaan full stack -suunnittelijasta, tarkoitetaan, että suunnittelija osaa koodausprototyypittelyn eri kerrokset:

- Vuorovaikutussuunnittelun
- Käyttöliittymäsuunnittelun
- Käyttäjäkokemussuunnittelun
- Front end -kehityksen

Front end -kehitys pitää sisällään HTML- ja CSS-merkkausten osaaminen käyttöliittymän visuaalisen ilmeen toteutukseen sekä JavaScriptin osaaminen käyttöliittymän animaatioiden sekä vuorovaikutusten toteuttamiseen. Kuvan 3 mukaisesti käyttäjäkokemussuunnittelussa suoritetaan tutkimus sekä tehdään rautalankamallit käyttöliittymästä. Käyttöliittymäsuunnittelussa toteutetaan käyttöliittymän tarkat näkymät, ikonit ja materiaali. Animaatiot, interaktiot sekä viimeiset prototyypit valmistetaan vuorovaikutussuunnittelun aikana. [Canziba, 2018]



Kuva 3. Full stack -suunnittelijan ja kehittäjän eroavaisuudet [Canziba, 2018].

Canziba [2018] muistuttaa, ettei full stackillä tarkoiteta sitä, että henkilö tekee kaiken alusta loppuun suunnittelusta tietokantojen ylläpitoon, vaan että henkilö on kehittänyt laajasti eri taitoja, joiden avulla hänen olisi mahdollista tuottaa täydellinen suunnitelma tai kehitysprojekti loppuun asti yksin. Full stack -suunnittelijan on tarkoitus olla tuotteen suunnittelun elinkaaren ajan aina tutkimuksen laatimisesta animaatioiden ja interaktioiden tuottamiseen front endissä. Full stack -suunnittelijan ei ole tarkoitus olla erityisen loistava koodaamaan, mutta hänen täytyy ymmärtää mitä teknologioita siinä käytetään ja kuinka se toimii, sillä kehittäjän tehtävänä on yhdistää ohjelman back end -suunnittelijan tuottamaan viitekehykseen. Hyvä suunnittelija ei keskity pelkästään käyttöliittymään tai

suunnitteluun vaan ottaa myös huomioon asiat, jotka ovat liitettynä siihen, jolloin suunnittelija saa aikaiseksi kokonaisuuden, josta kehittäjien on hyvä jatkaa työtä. Perinteisen suunnittelijan ja full stack -suunnittelijan erona onkin se, että full stack -suunnittelijalla on kyky ajatella kokonaiskuvaa, sillä hän on mukana koko kehitysprosessin ajan, toisin kuin perinteinen suunnittelija on vain osana jotain tiettyä kohtaa suunnittelussa. Full stack -kehittäjään verrattuna, kehittäjän tarkoituksena on ymmärtää laaja spektri eri teknologiota, jotta kehitys olisi mahdollisimman jouhevaa. Full stack -suunnittelija keskittyy enemmän tuottamaan laadukasta suunnitelmaa, käytettävyyttä sekä tuotetta. [Canziba, 2018]

Parhaimmillaan full stack -suunnittelija osaa hyvissä ajoin tunnistaa projektin odotukset ja rajoitukset suunnittelun aikana, jolloin tiedetään, mitkä asiat eivät tule toimimaan ja saadaan realistiset odotukset projektiin. Näin säästetään aikaa ja rahaa projektin myöhemmissä vaiheissa, kun koodia tai ensimmäisiä visuaalisia komponentteja aletaan tuottamaan. Full stack -suunnittelija tuo monipuolista hyötyä suunnittelutiimille ja koko organisaatiolle. Heidän laaja-alaisen osaamisensa ansiosta he ymmärtävät tarkasti, mitä odottaa koodausvaiheessa ja mikä parantaa yhteistyötä kehitystiimin kanssa. Tämä tietoisuus mahdollistaa paremman visuaalisen suunnittelun, huomioiden tekniset mahdollisuudet ja käyttöliittymän tarpeet. [Canziba, 2018]

Full stack -suunnittelijoita voidaan kutsua myös ”yksisarvisiksi” (engl. unicorn), koska suunnittelijoita, jotka osaavat sekä suunnitella että koodata on vaikea löytää [Abid ym., 2023; Botha, 2018; Kulchinskiy, 2021; Ng, 2024]. Myös yksisarvisen sarveen yhdistetään full stack -suunnittelijan eri suunnittelupinot, kuten käytettävyys-, käyttöliittymä- ja vuorovaikutussuunnittelu sekä front end -kehitys, jotka ovat esillä kuvassa 3. [Botha, 2018]

Useaan eri lähteeseen perustuen on alla listaus full stack -kehittäjän hyvistä sekä huonoista puolista [Botha, 2018; Canziba, 2018; Kulchinskiy, 2021]:

Hyvät puolet:

- Kokonaiskuvan ymmärtäminen
- Eri alueiden ymmärtäminen
- Tehtävien vaihtelevuus
- Hyvät mahdollisuudet freelancerina työskentelyyn
- Täyteläinen portfolio
- Hyvä lisäys projektin hallintaan ja tiimin vetäjille
- Mahdollisuus säästää aikaa ja rahaa projekteissa
- Prototyyppien testaus oikeilla käyttäjillä

Huonot puolet:

- Vaikea olla hyvä kaikessa
- Vaikea pysyä mukana teknologiakehityksessä
- Ajanhallinta
- Vaikea kehittyä uralla
- Full stack -suunnittelija usein huono koodaamaan

2.6 Yhteenveto

Kirjallisuuskatsauksesta päätellen full stack -suunnittelijalle ei ole löydetty vielä vakiintunutta roolia ketterässä ohjelmistokehityksessä, eikä näin ollen osata sanoa, mitkä tämän roolin hyödyt tai haasteet voivat olla tai että onko koko roolille edes tarvetta. Käyttäjäkokemussuunnittelun yhdistäminen ketteriin menetelmiin on hankala, sillä ne eivät lähtökohtaisesti tue käyttäjiä, vaan asiakkaita. Suurimmiksi ongelmiksi nousivat puutteellinen kommunikointi kehittäjien ja suunnittelijoiden välillä, liian pieni painoarvo käyttäjäkeskeiselle suunnittelulle sekä ajan puute suorittaa ennakkosuunnittelua sekä testejä oikeiden käyttäjien kanssa.

Yhdeksi ratkaisuksi ehdotettiin, että suunnittelutyö olisi yhtä iteraatiota edellä ketterää kehitystä, jolloin suunnittelu- ja kehitystiimien aikataulutus olisi paremmassa kunnossa. Ennen kuin varsinainen ohjelmointi aloitettaisiin, dataa olisi pitänyt olla kerättynä asiakkaalta ja suunnittelu tehtynä datan pohjalta sekä tiimien väliseen kommunikointiin panostettaisiin enemmän. Full stack -suunnittelijalle nämä ratkaisut voivat silti aiheuttaa ajanhallintaongelmia.

3 Tutkimusmenetelmä

Tässä luvussa käydään läpi aluksi tähän tutkimukseen valittu tutkimusmenetelmä ja aineiston tyyppi. Tämän jälkeen kerrotaan haastateltavista henkilöistä, haastattelun suunnitelma, haastattelujen toteuma sekä lopuksi haastattelutulosten analysointi.

3.1 Laadullinen tutkimus

Tähän pro gradu tutkielmaan valittiin kvalitatiivinen eli laadullinen tutkimus tutkimusmenetelmäksi. Tämä menetelmä valittiin, koska tässä tutkimuksessa halutaan ymmärtää mahdollisimman kokonaisvaltaisesti full stack -suunnittelijan roolin tarpeellisuus ketterässä ohjelmistokehityksessä. Hirsjärvi ym. [2007] kertovat, että laadullisen tutkimuksen tarkoituksena on tutkia tutkittavaa ilmiötä mahdollisimman kattavasti sekä monipuolisesti. Tiedon keruussa luotetaan enemmän tutkijan omiin havaintoihin sekä keskusteluihin ihmisten kanssa, kuin mitattavissa olevaan tietoon, siksi kohderyhmä haastattelulle valittiin tarkasti.

Laadulliset tutkimukset nojaavat aiempiin tutkimuksiin tutkittavasta aiheesta, empiiriseen aineistoon ja tutkijan omiin päättelyihin ja ajatuksiin, kun taas määrälliset tutkimukset pohjautuvat pääasiassa kerättyyn aineistoon ja sen perusteella tehtyihin mittaustuloksiin, sekä tutkijan omaan pohdintaan. Laadullisten ja määrällisten tutkimusten erot tulevat esiin niiden erilaisissa tutkimusasetelmissa eli siinä, miten tutkimusongelmiin kerätään aineistoa käyttäen tiettyjä menetelmiä. Laadullisessa tutkimuksessa aineistonkeruumenetelmänä voi olla esimerkiksi haastattelut, havainnointi, kirjeet sekä elämänkerrat. [Saaranen-Kauppinen ym., 2006]

Tämän tutkimuksen laadullista tutkimusmenetelmää voidaan kuvailla monin eri tavoin. Tutkimuksessa ei käytetä kokeellista lähestymistapaa, vaan pyritään ymmärtämään tutkittavien henkilöiden moninaisuutta ja syvyyttä. Jokaisen osallistujan yksilölliset näkökulmat ja kokemukset otetaan tarkasti huomioon, mikä mahdollistaa monipuolisen aineiston keräämisen. Vaikka aineistokoko on suhteellisen pieni, se mahdollistaa syvällisemmän analyysin ja yksityiskohtaisemman tulkinnan tutkittavista ilmiöistä.

3.2 Haastattelu aineistona

Tämän tutkimuksen tavoitteena on selvittää full stack -suunnittelijan roolia ketterässä ohjelmistokehityksessä, mitä hyötyjä tai haasteita roolista on ja koetaanko rooli tarpeelliseksi. Kirjallisuuskatsauksessa selvitettiin roolille ominaisia piirteitä, mitä työtehtäviä roolilla on ja miten rooli toimii muiden roolien kanssa ketterässä ohjelmistokehityksessä.

Haastatteluilla on tarkoitus selvittää ohjelmistokehittäjien sekä suunnittelijoiden omakohtaisia kokemuksia nykyisestä ketterästä kehityksestä ja miten he kokevat full stack -suunnittelijan roolin heidän projekteissaan. Haastattelut valittiin siksi, että saataisiin mahdollisimman tarkkoja vastauksia. Kirjallisuuskatsauksen alakohdassa 2.2.1 kerrottiin, mitä eri käyttäjäkokemussuunnittelijan rooleja on ketterässä kehityksessä ja mitä eri tehtäviä niillä on [Silva ym., 2013]. Lisäksi kohdassa 2.5 puhuttiin full stack -suunnittelusta, mitä se käytännössä on, mitä full stack -suunnittelijalla tarkoitetaan tässä tutkielmassa ja mitä hyviä sekä huonoja puolia tästä roolista on [Botha, 2018; Kulchinskiy, 2021; Canziba, 2018]. Näiden kirjallisuuskatsauksesta löydettyjen tietojen perusteella muodostettiin haastatteluteemoja ja haastattelukysymykset, jotta haastattelulla saataisiin vastattua mahdollisimman hyvin tutkimuskysymykseen.

Laadulliselle tutkimukselle ominaiset aineistonkeruumenetelmät ovat haastattelut ja havainnointi [Saaranen-Kauppinen ym., 2006]. Tähän tutkielmaan valittiin aineistonkeruumenetelmäksi haastattelut, sillä tutkimuksessa halutaan tuoda esille vahvasti työntekijöiden omakohtaisia kokemuksia, aihe on melko tuntematon sekä halutaan saada tarkkoja vastauksia. Hirsjärvi ym. [2007] perustelevat haastattelun käyttöä, kun halutaan keskittyä ihmiseen tutkimuskohteena, ennakoitavien vastausten suunta on epävarma ja pyritään syventymään mielipiteiden taustalla oleviin perusteluihin. Laadullisessa tutkimuksessa aineistona voi olla yhden henkilön haastattelu tai joukko yksilöhaastatteluja, sillä tutkimuksen tarkoitus ei ole löytää tilastollisia säännönmukaisuuksia tai keskiarvoisia yhteyksiä. Riittävä aineisto voidaan kuvata, kun haastatteluissa alkaa toistua samat teemat tai asiat [Hirsjärvi ym., 2007]. Tarkoituksena on saada kahdeksan haastattelua, mutta jos aineistossa havaitaan puutteita, suoritetaan tarvittaessa lisää haastatteluja.

Tutkittaessa ihmisiin liittyviä asioita, tietojenkäsittelytieteissä käytetään aineistonkeruumenetelmänä usein haastatteluja. Tutkimushaastatteluissa tunnistetaan kaksi päätyyppiä: teemahaastattelu ja lomakehaastattelu eli strukturoitu haastattelu. Teemahaastattelussa keskustelu etenee vapaasti haastattelijan asettamien aiheiden pohjalta, kun taas strukturoidussa haastattelussa haastattelija esittää valmiiksi laaditun kysymysluettelon mukaisia kysymyksiä haastateltavalle. Tähän tutkimukseen sopivin haastattelutyyppi on teemahaastattelu, sillä haastateltavia on alle kymmenen sekä vastauksista halutaan saada kokonaisvaltaisempi kuva. Ennalta määritellyt teemat ohjaavat keskustelua tutkimuskysymysten ytimiin, mikä estää epäolennaisten asioiden esiintymisen haastattelussa. Tämä menetelmä tarjoaa tilaa haastateltavalle ilmaista omia ajatuksiaan ja perustelujaan, mikä edistää kokonaisvaltaista ilmiön ymmärtämistä. Lisäksi teemahaastattelu mahdollistaa syventymisen integraation onnistumiseen ilman strukturoitua rakennetta, mikä tuottaa tietoa,

joka on luonteeltaan paljon syvällisempää kuin strukturoitujen haastattelujen tuottama tieto [Tiainen, 2014].

3.3 Haastateltavat

Halutun aineiston edellytyksenä on, että haastateltavat valitaan tarkoitustenmukaisesti. Laadullisessa tutkimuksessa pyritään ymmärtämään ilmiötä kattavasti, ja tämän tavoitteen saavuttaminen usein onnistuu parhaiten hyödyntämällä heterogeenista haastateltavajoukkoa [Tiainen, 2014]. Aineiston rajauksen ja valinnan on oltava perusteltuja kaikissa tutkimuksissa. Kun suunnitellaan haastatteluja, on tärkeää vastata kysymykseen: miksi juuri nämä henkilöt valitaan tutkimuksen kohteeksi? [Saaranen-Kauppinen ym., 2006]

Tämän tutkielman tutkimuskysymyksiin pystyy vastaamaan vain henkilöt, jotka tekevät töitä ketterän ohjelmistokehityksen parissa joko suunnittelijana tai kehittäjänä tai että heillä on kokemusta siitä. Myös opiskelijat, joilla on kokemusta kursseilta käyttäjäkeskeisestä suunnittelusta tai ohjelmistokehityksestä ovat päteviä henkilöitä osallistumaan tähän tutkimukseen. Näiden kriteerien täyttyminen varmistetaan haastattelun esitietolomakkeessa (Liite 1), jossa tiedustellaan perustietojen lisäksi henkilön kokemusta. Perustiedot ovat perusmuuttujia, joita kysytään, vaikka niitä ei välttämättä käytetä haastateltavien valintakriteereinä. Perustietoja ovat:

- Syntymävuosi
- Korkein suorittama tutkinto
- Kuinka suuressa yrityksessä ja tiimissä työskentelee

Haastateltaviksi henkilöiksi päätyi Solitalla työskenteleviä ohjelmistokehittäjiä ja suunnittelijoita. Solita on suomalainen teknologiakonserni, joka keskittyy tuottamaan IT-ratkaisuja yrityksille sekä julkisen hallinnon organisaatioille. Solita työllistää Suomessa sekä tytäryhtiöissä Ruotsissa, Norjassa, Virossa, Saksassa, Tanskassa, Belgiassa, Puolassa ja Sveitsissä yhteensä 1500 henkilöä. Solitan henkilöstöllä on kokemusta useista erilaisista asiakasprojekteista, joissa ohjelmistoa on kehitetty ketterin menetelmin ja suunnittelu on otettu hyvin huomioon [Solita, 2024]. Nämä tiedot riittivät perusteluksi valita Solita yritykseksi, josta haastatella työntekijöitä tutkimusta varten. Haastateltaviksi henkilöiksi päätyi myös tietojenkäsittelytieteen opiskelijoita, jotka ovat opintojensa loppusuoralla ja saaneet kokemusta paljon erilaisista ohjelmointi ja käyttäjäkeskeisistä kursseista.

Haastateltavien joukossa oli ohjelmistokehittäjiä, UI/UX-suunnittelijoita, applikaatiokonsultti ja automaatiotestaajaharjoittelija. Osa haastateltavista opiskelee samanaikaisesti myös ihmisen ja teknologian vuorovaikutusta (engl. Human-Technology Interaction).

Haastateltavat toimivat pääosin isoissa yli 250 henkilön yrityksissä, ohjelmisto- ja pankkialalla, ohjelmistotuotannon parissa. Kaikilla paitsi yhdellä haastateltavalla oli vähintään vuoden kokemus käyttäjäkokemussuunnittelusta joko nykyisessä yrityksessä, aikaisemmissa yrityksissä tai opinnoista. Suurin osa haastateltavista työskenteli yhdessä projektissa kerrallaan paria haastateltavaa lukuun ottamatta. Osalla haastateltavista saattoi olla myös muita rooleja ilmoitetun roolin lisäksi. Jotta haastateltavien tiedot pysyvät anonyymina, yritysten henkilöstön määrä on ilmoitettu suuntaa antavana ja työnimikkeet on yksinkertaistettu. Taulukossa 2 on annettu jokaiselle haastateltavalle oma tunniste.

Haastateltavan tunniste	**Syntymävuosi**	**Koulutus**
Suunnittelija 1	1981	Ylempi korkeakoulututkinto
Suunnittelija 2	1982	Ylempi korkeakoulututkinto
Suunnittelija 3	1979	Alempi korkeakoulututkinto
Kehittäjä 1	1982	Alempi korkeakoulututkinto
Kehittäjä 2	1976	Ylempi korkeakoulututkinto
Konsultti	1997	Alempi korkeakoulututkinto
Testaaja	1998	Alempi korkeakoulututkinto

Taulukko 2. Haastateltavien syntymävuodet ja korkein suoritettu tutkinto.

3.4 Haastattelun suunnitelma

Haastatteluilla oli tarkoitus löytää haastateltavien omia mielipiteitä ja kokemuksia ketterässä ohjelmistokehityksessä työskentelevien roolien yhteistyöstä, mitä rooleja he valitsisivat omaan toiveprojektiinsa, käyttäjäkokemussuunnittelusta, full stack -suunnittelijan yleisyydestä sekä sen hyödyistä että haitoista verraten heidän yritysten tarjoamiin projekteihin tai yliopiston kursseihin viitaten. Näitä kysymällä koitettiin luoda käsitys siitä, kuinka tärkeäksi koetaan henkilö, joka osaisi käyttäjäkokemussuunnittelun sekä suunnitelman toteutuksen ketterässä ohjelmistoprojektissa.

Haastatteluteemat muodostuivat seuraavanlaisesti:

1. Ketterän kehityksen roolit
2. Suunnittelutyön sisältö
3. Suunnittelu sekä kehitys samanaikaisesti
4. Full stack -suunnittelu

Ensimmäinen teema, ketterän kehityksen roolit, syntyi kirjallisuuskatsauksen pohjalta, kun siinä tarkasteltiin suosituinta ketterää menetelmää Scrumia [VersionOne, 2020] sekä käyttäjäkeskeistä suunnittelua. Näiden kahden menetelmän yhdistämisessä korostuvat tietyt keskeiset roolit, jotka ovat olennaisia onnistuneen projektin kannalta. Silva ym. [2013] tutkivat käyttäjäkeskeisen suunnittelun rooleja ja havaitsivat, että käyttäjäkokemussuunnittelijan rooli voi muuttua projektin aikana useita kertoja. Tämä joustavuus korostaa roolien merkitystä ja moninaisuutta ketterän kehityksen yhteydessä. Haastattelussa kysytään haastateltavilta, mitä rooleja he ottaisivat heidän unelmaprojektiinsa.

Toinen teema, suunnittelutyön sisältö, nousi esiin osana ketterän kehityksen rooleja käsittelevää tutkimusta. Silva ym. [2013] keskustelivat suunnittelutyön laajuudesta ja siitä, kuinka se muotoutuu projektin edetessä. Suunnittelutyö ei ole aina kiveen hakattu, vaan se sisältää monia eri osa-alueita, jotka voivat vaihdella projektin tarpeiden mukaan. Suunnittelutyön sisällön tarkastelu on tärkeää, koska se määrittelee, millaisia taitoja ja resursseja projektissa tarvitaan, ja miten ne jaetaan eri vaiheiden välillä. [Silva, 2013] Haastateltavilta kysyttiin, että mitä suunnittelutyötä he ovat tehneet ja mitä se on käytännössä pitänyt sisällään.

Kolmas teema käsittelee suunnittelun ja kehityksen samanaikaisuutta. Kirjallisuuskatsauksessa luvussa 2.5 Botha [2018], Kulchinskiy [2021], ja Canziba [2018] toivat esiin näkemyksiä siitä, millaisia etuja ja haasteita suunnittelun ja kehityksen yhdistäminen samanaikaisesti tuo mukanaan. Tämä voi nopeuttaa projektin valmistumista ja mahdollistaa joustavamman reagoinnin muutoksiin, mutta toisaalta se voi aiheuttaa haasteita kommunikaatiossa ja työprosessien hallinnassa. Tämä teema on keskeinen ymmärrettäessä, miten ketterä kehitys voidaan toteuttaa tehokkaasti. Haastattelussa kysyttiin esimerkiksi, että kuinka laajoja työtehtäviä eri rooleilla saisi olla ja missä tilanteissa olisi hyvä olla erilliset henkilöt suunnittelutyölle ja toteutukselle.

Neljäs teema käsittelee full stack -suunnittelua. Canziba [2018] määritteli full stack -suunnittelijan tehtäväalueet, jotka kattavat vuorovaikutus-, käyttöliittymä- ja käyttäjäkokemussuunnittelun sekä front end -kehityksen. Full stack -suunnittelija on monipuolinen

osaaja, joka kykenee yhdistämään suunnittelun ja ohjelmoinnin taitoja prototyypittelyn avulla. Tämä rooli on erityisen arvokas ketterissä projekteissa, joissa tarvitaan laaja-alaista osaamista ja kykyä mukautua nopeasti muuttuviin vaatimuksiin. Haastateltavilta kysyttiin aluksi, onko rooli tuttu ja että mitä se voisi tarkoittaa. Lopuksi kysyttiin hyötyjä sekä haasteita roolista, kun se oli avattu haastateltavalle kunnolla.

Jokaisen haastattelun tavoitepituudeksi oli asetettu 30 minuuttia, mutta jos haastateltavalla on pitkiä vastauksia tai keskustelua riittää aiheesta, niin maksimipituudeksi oli asetettu 45 minuuttia. Jos aika silti loppuisi kesken, niin sovitaan aika uudelle haastattelulle. Haastattelu alkaa lyhyellä esittäytymisellä sekä kiittämisellä haastatteluun osallistumisesta. Tämän jälkeen nauhoituslupa varmistettiin vielä toistamiseen ennen varsinaisen haastattelun alkamista. Haastattelu alkoi kysymällä haastateltavan omista kokemuksista, jonka jälkeen siirrytään haastateltavan omiin toiveisiin ja lopuksi käsiteltiin full stack -suunnittelijan roolia. Koko haastattelun ajan haastattelijalla oli vieressään haastattelurunko, josta hän pystyi varmistamaan haastattelun kulun sekä aiheessa pysymisen. Lopuksi kiitettiin vielä toistamiseen haastateltavaa osallistumisesta ja kerrottiin, että valmis tutkielma lähetetään samaan sähköpostiosoitteeseen kuin kutsulinkki, jonka saa lukea halutessaan. Haastattelurunko löytyy liitteestä 4.

3.5 Haastattelujen toteuma

Haastattelut oli sovittu erikseen etukäteen joko suullisesti sopimalla, sähköpostilla tai lähestymällä haastateltavia Telegram-keskustelusovelluksen kautta. Haastattelukutsuissa oli lyhyt esittelyteksti tutkijasta ja tutkittavasta aiheesta. Ennen varsinaista haastattelua lähetettiin toinen sähköposti haastateltavalle, jossa oli kutsu haastatteluun. Kutsu piti sisällään linkit haastateltavan esitietolomakkeeseen sekä haastattelun tallennuksen suostumuslomakkeeseen. Haastattelut suoritettiin etäyhteydellä käyttäen Microsoft Teams -kokoussovellusta.

Haastateltavia löydettiin alun perin kahdeksan kappaletta. Haastattelujen oli tarkoitus toteutua 8.5.2024 – 31.5.2024 välillä, joista lähti kutsut haastateltaville 6.5.2024. Tutkielman haastattelut toteutettiin 8.5.2024 – 16.5.2024 välisenä aikana ja haastattelut suoritettiin kello 11.00–20.00 välillä. Ajanjakson aikana haastateltiin seitsemää eri henkilöä. Haastateltavien henkilöiden määrä koettiin tarpeelliseksi, sillä haastatteluissa nousivat esiin samat asiat, minkä avulla saatiin varmuus aineiston pätevyydelle. Haastatteluaikojen sopiminen onnistui hyvin, sillä haastattelut pidettiin etäyhteydellä.

Haastateltavat saivat sähköpostiin kutsulinkin haastatteluun, jossa kerrottiin, että kutsulinkin mukana tulevissa lisätiedoissa olivat linkit tallennussuostumukseen sekä esitie-

tolomakkeeseen, jotka piti täyttää ennen haastatteluun tuloa. Microsoft Teams -sovelluksen luomassa viestissä ei kuitenkaan tullut tarpeeksi hyvin esille haastattelun lisätietoosio, jonka takia jouduttiin lähes kaikille haastateltaville lähettämään vielä yksi muistutusviesti lomakkeiden täyttämistä varten. Kaikki haastateltavat kuitenkin vastasivat lomakkeisiin ennen haastattelua.

Haastattelut itsessään olivat hyvin saman tyyppisiä ja aiheessa pysyttiin jokaisessa haastattelussa. Jokainen haastateltava suostui haastattelun tallennukseen, eikä kehenkään haastateltavaan tarvinnut olla uudestaan yhteydessä epäselvyyksien vuoksi. Vastauksia saatiin myös hyvin, eikä lisäkysymyksiä juuri tarvinnut esittää. Myös tavoiteajassa pysyttiin, eikä missään haastattelussa ylitetty haastattelulle varattua aikaa. Lyhin haastattelu kesti 16:24 minuuttia ja pisin 27:13 minuuttia.

3.6 Haastattelutulosten analysointi

Toukokuussa 2024 aloitettiin aineiston analysointi. Haastattelut litteroitiin, jonka jälkeen ne vielä tarkistettiin kirjoitusvirheiden osalta. Myös jokainen litteroitu haastattelu koodattiin omilla haastattelulyhenteillä, esimerkiksi Suunnittelija 3. Haastattelussa käytettiin termien lyhenteitä tekstin lyhentämiseksi sekä lukemisen helpottamiseksi: UI-suunnittelu (käyttöliittymäsuunnittelu), UX-suunnittelu (käyttäjäkokemussuunnittelu) ja PO (tuoteomistaja) (engl. Product owner). Myös suunnittelun englannin kielistä termiä ”design” sekä kehittäjän englannin kielistä termiä ”developer” käytettiin, sillä ne ovat ammattisanastossa tunnetumpia termejä.

Teemahaastattelujen aineiston käsittelyssä teemoittelu on yleinen analyysitapa. Tämä tarkoittaa keskeisten aiheiden eli teemojen tunnistamista aineistosta löytämällä yhdistäviä tai erottavia elementtejä. Teemat voivat muistuttaa haastattelurungon aiheita, mutta aineisto saattaa myös paljastaa uusia teemoja. Teemojen luomisessa voidaan hyödyntää koodausta, jossa aineistoon tehdään jäsentäviä merkintöjä, esimerkiksi alleviivauksia eri väreillä. Näin aineistosta löydetään keskeiset asiat, jotka voidaan ryhmitellä teemoiksi. Aineisto järjestetään sitten näiden teemojen mukaan, jolloin kerätään yhteen kaikki kohdat, joissa teemaa käsitellään [Saaranen-Kauppinen ym., 2006].

Teemoitteluun voidaan yhdistää sisällönanalyysi, joka tarkastelee aineiston yhtäläisyyksiä ja eroja. Sen avulla pyritään luomaan tiivis kuvaus tutkittavasta aiheesta ja yhdistämään tulokset laajempaan kontekstiin sekä muihin tutkimuksiin. Sisällönanalyysi voi olla teorialähtöistä, teoriaohjaavaa, tai aineistolähtöistä. Vaikka aineiston analyysi ei perustu suoraan teoriaan, siitä voidaan löytää teoreettisia yhteyksiä. Teoria voi tarjota vahvistusta

ja selityksiä aineiston löydöksille [Saaranen-Kauppinen ym., 2006]. Tässä tutkimuksessa analyysi suoritettiin teoriasidonnaisesti, perustuen kirjallisuuskatsauksen tuloksiin.

Tässä tutkimuksessa analysointimenetelminä käytettiin teemoittelua ja teoriasidonnaista sisällönanalyysiä. Tavoitteena oli löytää tutkimuskysymyksen kannalta keskeiset yhdistävät aiheet ja ymmärtää millaisia hyötyjä sekä haittoja full stack -suunnittelijasta on ketterässä ohjelmistokehityksessä. Aluksi etsittiin aineistosta teemoja ja alateemoja koodauksen avulla.

Teemoja lähdettiin etsimään aluksi kirjallisuuskatsauksen mukaisesti, jolloin havainnoista löydettiin myös poikkeamia niistä. Ensimmäiseksi teemaksi valikoitui suunnittelutyö ketterässä kehityksessä, jonka alateemoina olivat yhteistyön toimivuus eri roolien välillä sekä roolien jaottelu. Myöhemmässä vaiheessa neljänneksi teemaksi muodostui projektin jäsenten roolit perustuen haastattelussa kysyttävään kysymykseen, jossa haastateltavalla on tarkoitus kertoa oma mielipide täydellisestä projektin roolien jaottelusta. Pian tämän jälkeen huomattiin, että vastausten tulokset ovat samantyyppisiä ensimmäisen teeman kanssa, joten teemat yhdistettiin.

Haastattelutulokset avattiin tekstinkäsittelyohjelmassa ja saman teeman tulokset merkittiin samalla värillä. Samankaltaiset aiheet yhdistettiin ja muodostettiin teemat. Teemat avattiin ja niitä tukevat haastattelulainaukset koottiin yhteen. Lopuksi verrattiin aineistoa kirjallisuuskatsauksen tuloksiin, luoden näin vahvemman pohjan johtopäätöksille.

Haastattelu antoi paljon tietoa ketterästä kehityksestä, suunnittelusta ketterässä kehityksessä, sekä mielipiteitä full stack -suunnittelijasta osana ketterää kehitystä. Haastatteluissa selvisi, että suunniteltu teemahaastattelurunko pysyi lähes samana toteutuneesta haastattelujen perusteella, mutta yksi teema nousi lisänä esille: Projektityypin vaikutus suunnittelutyölle ja full stack -suunnittelijalle. Tästä kerrotaan lisää kohdassa 4.6.

4 Haastattelun tulokset

Teemoittelu alkaa aluksi ylätasolta ketterän kehityksen rooleista, johon on integroituna käyttäjäkeskeinen suunnittelu sekä hahmotellaan, millaisia eri rooleja olisi hyvin toimivassa ketterässä ohjelmistokehitysprojektissa mukana. Sen jälkeen selvitetään, mitä suunnittelutyö pitää sisällään ja mitä mielipiteitä samanaikainen suunnittelu- ja kehitystyö jakaa. Lopuksi selvitetään full stack -suunnittelijan roolia ketterässä kehityksessä ja kuinka projektin tyyppi voi vaikuttaa full stack -suunnitteluun. Haastattelun alkuperäiset neljä teemaa ja lisäksi haastattelujen sisällön perusteella muodostetut viides sekä kuudes teema:

1. Ketterän kehityksen roolit
    a. Suunnittelijoiden roolit
    b. Tuoteomistajan rooli
    c. Kehittäjien roolit
2. Roolien yhteistyö projektinhallinnassa
    a. Suunnittelijoiden ja kehittäjien yhteistyö
    b. Yhteistyön haasteet
    c. Roolit projektin eri vaiheissa
3. Suunnittelutyön sisältö
4. Suunnittelu sekä kehitys samanaikaisesti
    a. Haastateltavien kokemuksia
    b. Hyödyt
    c. Haitat
5. Full stack -suunnittelu
    a. Ennakkoluulot
    b. Hyödyt
    c. Haitat
6. Projektityypin vaikutus full stack -suunnittelulle

Seuraavissa alakohdissa käsitellään teemat yksitellen. Teemojen kuvauksissa esitellään sitaatteja haastatteluista, jotta saadaan käsitys, miten teemoista käytännössä puhutaan.

4.1 Ketterän kehityksen roolit

Ensimmäisenä teemana selkeytyi, mitä rooleja on ketterissä ohjelmistokehitysprojekteissa. Teema rajautuu tunnistamaan ketterän kehityksen eri roolit. Haastatteluissa ilmeni, että suunnittelijoiden roolit voivat vaihdella projektin koosta ja luonteesta riippuen, mutta

monipuolinen osaaminen on tärkeää. Optimaalisessa tiimissä voisi olla esimerkiksi käyttöliittymäsuunnittelija ja palvelumuotoilija, jotka yhdessä varmistavat käyttäjäkokemuksen laadun. Tuoteomistajan rooli nähtiin keskeisenä projektin onnistumiselle, erityisesti hänen yhteistyönsä tuotesuunnittelijan (engl. product designer) kanssa. Kehittäjien roolijaosta oli vaihtelevia näkemyksiä: osa korosti front end -osaamisen tärkeyttä, kun taas toiset pitivät full stack -kehittäjiä riittävinä. Suurissa projekteissa erillisten front- ja back end -kehittäjien roolit voivat olla hyödyllisiä.

4.1.1 Suunnittelijoiden roolit

Haastateltavien kommenteista ilmeni selkeästi, että suunnittelijoiden määrä projektissa voi vaihdella yhdestä kolmeen riippuen projektin luonteesta. Haastatteluissa ilmeni tarve monipuoliselle osaamiselle suunnittelutiimissä. Monien mielestä optimaalinen kokoonpano sisältäisi kaksi suunnittelijaa: toisen, joka keskittyy digitaaliseen käyttöliittymäsuunnitteluun, ja toisen, joka toimii palvelumuotoilijana ja keskittyy projektin alkuvaiheen määrittelyihin. Tämä mahdollistaa synergiaedut, kun käyttöliittymäsuunnittelija voi hyödyntää palvelumuotoilijan keräämää tietoa.

”Mieluiten kaksi suunnittelijaa, että toinen ois tämmönen minunlainen digisuunnittelija, joka suunnittelee käyttöliittymää ja sitten palvelumuotoilija, joka keskittyy enemmän siihen alkupään määrittelyyn ja mä ammennan tietoa häneltä mitä voin käyttää” – Suunnittelija 1

Yksi haastateltavista ehdotti, että product designerin lisäksi tiimiin tulisi ottaa UX-designer, jolla on visuaalinen UX-osaaminen, jotta käyttäjäkokemus ja visuaalinen ilme saavat riittävästi huomiota.

”Product designerin kaveriksi ottaisin vielä UX-designerin, jolla on visuaalinen UX-osaaminen” – Suunnittelija 2

Myös asiakas- ja käyttäjäymmärryksen merkitystä korostettiin, ja tämän osaamisalueen hallitsijan nähtiin olevan olennainen osa suurten projektien onnistumista.

”Design-puolella niitä rooleja voi olla, sanotaanko yhdestä kolmeen. Sellanen niinku asiakasymmärrys tai käyttäjäymmärrys pitää olla eli joku, joka hallitsee sen. Et pystyy dokumentoimaan sen sellaiseksi, että siihen pystyy muu tiimi tukeutumaan isossa projektissa.” – Suunnittelija 3

Yhteistyö full stack -kehittäjien ja yhden tai kahden UX-tyypin kanssa mainittiin toimivaksi yhdistelmäksi, ja jotkut painottivat, että UI- ja UX-suunnittelijoiden roolit olisi hyvä pitää erillään. Kokonaisuudessaan suunnittelutiimin kokoonpano ja roolit tulee määritellä projektin tarpeiden mukaan.

”UI/UX-suunnittelijat, mieluusti erillänsä” – Konsultti

”Ois hyvä olla joku UX designer tyyppi tietenkin vähän projektista riippuen” – Testaaja

4.1.2 Tuoteomistajan rooli

Haastatteluissa korostettiin tuoteomistajan tärkeyttä projektin onnistumisessa, erityisesti hänen jatkuvalla ajantasaisuudellaan ja yhteistyöllään product designerin kanssa, joka vastaa käyttöliittymän ja käyttäjäkokemuksen suunnittelusta. Suurissa projekteissa tuoteomistajan rooli korostuu entisestään, ja usein palvelumuotoilua voidaan käsitellä erillisenä projektina, mikä varmistaa suunnittelutyön kattavuuden ja laadun. Seuraavaksi käydään läpi tarkemmin tuoteomistajan vaikutuksia projektin onnistumiselle.

Kirjallisuuskatsauksen alakohdassa 2.2.2 Kuusinen [2015] tunnisti kolme erilaista yhteistyötyyppiä käyttäjäkokemussuunnittelijoiden sekä muiden ketterässä kehityksessä työskentelevien roolien välillä. Yksi kolmesta tyypistä oli tuoteomistajan sekä käyttäjäkokemussuunnittelijoiden läheinen yhteistyö. Tuoteomistajan rooli liitettiin läheisesti suunnittelutyöhön, sillä tuoteomistajan osallistuminen suunnitteluun voi vaihdella projektin tarpeiden mukaan. Joissain projekteissa tuoteomistaja on ollut aktiivisesti mukana front end -suunnittelussa, tuoden esiin liiketoiminnan tarpeet ja varmistaen, että suunnittelutyö linjaa projektin tavoitteiden kanssa. Myös eräs haastateltavista koki, että tärkeintä koko projektissa on se, että tuoteomistaja tekee juuri sitä työtä mitä hänen pitäisi tehdä.

”Jokaisessa tiimissä on yksi Product owner joka on enemmän tai vähemmän niissä designhommissa mukana” – Kehittäjä 2

Kun haastateltavilta kysyttiin mielipiteitä ohjelmistoprojektista, jossa olisi mukana roolit, jotka he kokivat välttämättömiksi ja tarpeellisiksi, vastaukset olivat hyvin yhtenäisiä. Haastateltavat korostivat tuoteomistajan roolin tärkeyttä.

”PO:n rooli on tosi tärkeä.” – Kehittäjä 1

Tuoteomistajan rooli nähtiin keskeisenä, koska hän pystyy pitämään projektin oikealla kurssilla ja varmistamaan, että kehitetty tuote tai palvelu tuottaa mahdollisimman paljon

arvoa. Lisäksi yksi haastateltavista mainitsi, että asiakkaan tuotteenomistaja tulisi olla jatkuvasti ajan tasalla ja hänen rinnallaan tulisi olla product designer, joka vastaa käyttöliittymän ja käyttäjäkokemuksen suunnittelusta.

”Asiakkaan PO pitää olla ajan tasalla ja hänen kaveriksi product designer joka tekee UI/UX työtä.” – Suunnittelija 2

Toisen haastateltavan mielestä tärkeintä oli, että tuotteenomistaja tekee juuri sitä työtä, mitä hänen rooliinsa kuuluu, varmistaen näin projektin menestyksen.

”Kaikkein tärkeintä koko projektissa on, että on PO, joka tekee sitä työtä mitä PO:n pitäisi tehdä” – Suunnittelija 3

Jos projekti on kokoluokaltaan suuri, UX-suunnittelija ja palvelumuotoilija voivat tehdä toistensa töitä, jolloin palvelumuotoilija on suunnittelutyössä jatkuvasti mukana. Suurissa projekteissa on usein oma suunnittelutiimi, joka tuottaa suunnitelmia kehitystiimille, tai palvelumuotoilu on saatettu toteuttaa erillisenä projektina ennen varsinaisen kehitysprojektin alkamista.

4.1.3 Kehittäjien roolit

Tässä alakohdassa käsitellään kehitystiimin roolijakoa ja siihen liittyviä mielipiteitä. Haastatteluissa ilmeni, että kehittäjien roolijako front end- ja back end -osaajien välillä tai full stack -kehittäjien käyttö jakoi mielipiteitä. Jotkut kokivat, että erillisten roolien merkitys on vähäinen, kun taas toiset korostivat front end -osaamisen tärkeyttä, erityisesti käyttöliittymäsuunnittelussa ja testauksessa. Erityisesti front end -suunnittelijan rooli nähtiin hyödyllisenä, sillä se mahdollistaa ratkaisujen kokeilun ennen niiden siirtämistä kehitystiimille. Vahva front end -osaaminen koettiin tärkeäksi, koska front end -työ voi olla hitaampaa ja vaatia enemmän kokeilua, jos sitä kehittää full stack tai back end -kehittäjä. Erityisesti suurissa projekteissa erillisten front end- ja back end -kehittäjien roolit voivat olla hyödyllisiä, jotta kaikki osa-alueet saavat tarvittavaa asiantuntemusta ja huomiota.

Kehitystiimin jäsenten roolit jakoivat mielipiteitä. Joidenkin mielestä ei ole väliä, onko kehittäjillä erikseen front end- ja back end -rooleja, vai toimivatko he full stack -kehittäjinä.

”Jos mietitään vuoden parin projektia, niin siellä on kehittäjiä, mun puolesta ei oo väliä onko erikseen frontti ja bäkkäri vai onko full stackki” – Suunnittelija 1

Toiset korostavat front end -osaamisen tärkeyttä, erityisesti käyttöliittymäsuunnittelun ja testauksen yhteydessä, ja ehdottavat erillisen front end -suunnittelijan roolia, joka voisi toteuttaa ratkaisuja ennen niiden siirtämistä varsinaiselle kehitystiimille.

”Sitten devi puolella ois tällanen niinku fronttidesigner jonka tehtävänä on oikeestaan toimia mun hiekkalaatikkona, että jos mulla on joku käyttöliittymäidea niin ennen kuin se siirretään varsinaiselle dev tiimille, niin me koeponnistetaan se ratkaisu et miten se vois toimia ja sitten kun meillä on ratkaisu niin sitten vasta tiketöidään se sinne varsinaisesti.” – Suunnittelija 3

Yksi haastateltava kuitenkin painottaa, että vahva front end-osaaminen on hyödyllistä, sillä full stack -kehittäjille front end -työskentely voi olla hitaampaa ja vaatia enemmän toistoja.

”Nyky UI-kehitystavoilla se vahva fronttiosaaminen on hyvä olla, toki on olemassa myös paljon full stack devaajia mutta sitten se frontin tekeminen on kuitenkin hitaampaa ja siinä on jonkin verran semmosta hakemista” – Kehittäjä 2

Toisen haastateltavan mielestä tiimissä pitäisi olla erilliset front end- ja back end -kehittäjät varmistaakseen, että kaikki osa-alueet saavat tarvittavaa huomiota ja asiantuntemusta.

”Pitäisi olla fronttidevaaja, bäkkäridevaaja, server side jamppa ois hyvä olla.” – Testaaja

4.2 Roolien yhteistyö projektinhallinnassa

Toinen teema syntyi haastattelutulosten perusteella. Se rajautuu kertomaan kehittäjien ja suunnittelijoiden välisen yhteistyön onnistumisista ja haasteista sekä ketterän kehityksen rooleista ja suunnittelutiimistä eri vaiheissa projektia. Yhteistyön sujuvuus riippuu erityisesti tiimien integroinnista, selkeästä työnjaosta ja jatkuvasta viestinnästä. Ketterät työskentelymallit tukevat tätä yhteistyötä erityisesti projektien alkuvaiheessa. Haastateltavat korostivat, että toimiva yhteistyö edellyttää suunnittelun ja kehityksen tiivistä yhteispeliä koko projektin ajan, mikä varmistaa projektin sujuvuuden ja laadun. Haasteita syntyy, jos viestintä on puutteellista tai suunnittelijat eivät ole jatkuvasti mukana projektin edetessä.

4.2.1 Suunnittelijoiden ja kehittäjien yhteistyö

Yhteistyö kuvattiin kuitenkin lähes poikkeuksetta toimivaksi, eikä suuria ongelmia nostettu esille. Monet haastateltavat korostivat, että heidän uransa aikana suunnittelutiimit ovat olleet hyvin integroituja kehitystiimeihin, ja ketterät mallit ovat auttaneet pitämään suunnittelutyön joustavana ja yhteistyökykyisenä. Usein myös projekti ja henkilöt keiden kanssa tekee töitä vaikuttavat yhteistyön onnistumiseen.

”Toimii ainakin tällä hetkellä tuossa nykyisessä projektissa todella hyvin, että se riippuu aina tietysti henkilöstä ja projektista” – Suunnittelija 3

Yhteistyö toimii parhaiten silloin, kun suunnittelutyö on aktiivinen osa kehitystiimiä ja suunnittelutyötä ei tehdä liikaa etupainotteisesti. Tämä lähestymistapa varmistaa, että suunnitteluratkaisut voidaan sopeuttaa nopeasti projektin edetessä ja että kehittäjien ja suunnittelijoiden välillä säilyy jatkuva vuoropuhelu.

”Mun työuran aikana on ollut aika hyvin toimivaa, että design on aika pitkälle osa niinku sitä devitiimiä ja mennään niinku näiden ketterien mallien mukaisesti, ettei suunnitella liikaa sinne etupeltoon” – Suunnittelija 2

Pienemmissä tiimeissä, joissa jokaisella oli selkeästi määritelty oma roolinsa, yhteistyö toimi hyvin, koska kaikki tiesivät tarkkaan omat tehtävänsä ja vastuualueensa. Tämä synkronointi auttoi varmistamaan, että tiimillä oli yhteinen ymmärrys projektin tavoitteista ja työskentelytavoista.

”Pieni tiimi, jokainen hoiti vähän niinku oman hommansa siinä ja se toimi tosi hyvin” – Konsultti

Haastatteluista kävi ilmi, että hyvä yhteistyö perustuu tehokkaaseen viestintään ja selkeään työnjakoon. Tiimien onnistunut yhteistyö vahvisti sitä, että suunnittelijat ja kehittäjät voivat toimia saumattomasti yhdessä, kunhan heillä on selkeät roolit ja he ylläpitävät jatkuvaa dialogia koko projektin ajan.

” Yhteistyö toimi hyvin, että meillä oli aika hyvä synkka että me tiedettiin mitä me haluttiin tehdä.” – Testaaja

”Tosi toimiva combo on ollut, että jos on ollut porukka full stack -kehittäjiä ja sitten niitä UX-tyyppejä yks tai kaks.” – Kehittäjä 1

Selvisi myös, että projektin luonne määrittelee merkittävästi yhteistyön tiiviyden ja laadun. Usein projektin alkupäässä yhteistyö kehittäjien ja suunnittelijoiden välillä on tiiviimpää, mutta loppupuolella se voi vähentyä tai jopa kadota kokonaan. Lisäksi pidempiaikaiset projektit tarjoavat paremmat edellytykset tehokkaalle yhteistyölle verrattuna lyhyempiin projekteihin.

”Alkupuolella se on niinku tiiviimpää se yhteistyö kehittäjien kanssa ja sitten ehkä kanssa, kun tulee niinku ensimmäiset testiversiot tai ensimmäiset tuotantoversiot niistä UX-näkökulmasta niin sitten on taas vähän aktiivisempaa. Julkisen puolen projekteista sitten niinku ollut monta vuotta osana sitä tiimiä niin kyllä se toimii paremmin” – Suunnittelija 1

Haastatteluissa korostettiin, että projektin alkuvaiheessa yhteistyö on erityisen tiivistä, kun suunnittelijat ja kehittäjät työskentelevät yhdessä ensimmäisten versioiden ja testiversioiden parissa. Tämä vaihe on kriittinen, sillä silloin varmistetaan, että suunnitelmat ja tekniset ratkaisut ovat linjassa ja tukevat toisiaan. Projekteissa, joissa tiimien jäsenet voivat työskennellä yhdessä useita vuosia, yhteistyö koettiin erityisen hyvin toimivaksi. Pitkäaikainen yhteistyö luo vakiintuneita käytäntöjä ja syventää tiimin jäsenten välistä ymmärrystä ja luottamusta, mikä parantaa työn sujuvuutta ja laatua. Näissä projekteissa jatkuva yhteistyö ja vuoropuhelu suunnittelijoiden ja kehittäjien välillä on ratkaisevaa onnistuneiden lopputulosten saavuttamiseksi.

4.2.2 Yhteistyön haasteet

Seuraavaksi tarkastellaan yhteistyön haasteita projekteissa, erityisesti kuinka projektin kesto ja luonne vaikuttavat yhteistyön laatuun. Vaikka yhteistyötä kehuttiin paljon, ilmeni myös ongelmia, jotka liittyivät suunnittelijan tai suunnittelijoiden puuttumiseen projektista joko osittain tai kokonaan, kuten kävi ilmi kohdassa 2.2.

Yhdessä haastatteluissa mainittiin, että projekteissa esiintyy toisinaan "rikkinäisen puhelimen" efektiä, jossa suunnittelijoiden päätökset ja tekemiset eivät välity selkeästi kehitystiimille. Tämä voi johtaa tilanteisiin, joissa kehittäjät huomaavat liian myöhään, että tiettyjä asioita ei olisi pitänyt tehdä tietyllä tavalla. Tämä ongelma korostuu erityisesti silloin, kun suunnittelijat eivät ole jatkuvasti mukana projektin edetessä.

”Tällä hetkellä toimitaan ne vähän semmoista niinku rikkinäisen puhelimen efektiä, että on jossain toisessa paikassa missä ite ei ollut mukana, niin on jotakin päätetty ja tehty ja sitten mä huomaan ehkä vielä myöhään, että oho, tuossa ei ois ehkä kannattanut tehdä noin” – Suunnittelija 1

Joissakin tapauksissa suunnittelutiimi tekee paljon työtä projektin alkuvaiheessa ja siirtyy sitten toiseen projektiin, jättäen kehitystiimin selvittämään suunnitteluratkaisuja yksin. Tämä voi aiheuttaa merkittäviä ongelmia, kun kehittäjät joutuvat tulkitsemaan suunnitelmia ilman suunnittelijoiden tukea, mikä johtaa epäselvyyksiin ja mahdollisiin virheisiin toteutuksessa.

”Aikasemmassa vaiheessa tehtiin tosi paljon sitäkin, jossa designtiimi teki paljon etukäteen ja sitten designtiimi saattoi häipyä toiseen projektiin ja sitten devitiimi on jäänyt keskenään ihmettelemään, että mitähän tälläkin on tarkoitettu.” – Suunnittelija 2

Vaikka tiivis ja jatkuva yhteistyö yleensä parantaa lopputuloksia ja käyttäjäystävällisyyttä, projektien aikana voi ilmetä merkittäviä ongelmia. Näitä ongelmia voivat olla suunnittelijoiden puuttuminen projektista tai tilanteet, jossa suunnittelijoiden päätökset eivät välity kehitystiimille selkeästi. Näin syntyy tilanteita, joissa kehittäjät huomaavat virheitä vasta liian myöhään. Lisäksi, jos suunnittelutiimi jättää projektin ennen sen päättymistä, kehittäjät voivat jäädä epäselvien suunnitteluratkaisujen kanssa ilman tarvittavaa tukea.

4.2.3 Roolit projektin eri vaiheissa

Tässä alakohdassa käsitellään suunnittelijoiden rooleja ketterän kehityksen eri vaiheissa. Haastattelujen perusteella on selvää, että jatkuva vuoropuhelu ja yhteistyö suunnittelijoiden ja kehittäjien välillä ovat keskeisiä onnistuneen lopputuloksen saavuttamiseksi. UI/UX-suunnittelija on tärkeä osa ketterää kehitystiimiä, mutta heidän tarpeellisuutensa vaihtelee projektin luonteen mukaan. Haastatteluissa mainittiin palvelumuotoilijan rooli UI/UX-suunnittelijan lisänä.

”On käyttöliittymäsuunnittelija tai UX-designer ja sitten on palvelumuotoilija erikseen vielä.” – Suunnittelija 2

Kirjallisuuskatsauksen alakohdasta 2.2.1 ei käy ilmi palvelumuotoilijan roolia käyttäjäkeskeisessä suunnittelussa integroituna ketterään kehitykseen, vaan Silva ym. [2018] puhuivat lähinnä käyttäjäkokemussuunnittelijan kolmesta eri roolista; vuorovaikutussuunnittelijasta, käyttäjäkokemussuunnittelijasta sekä käyttöliittymäkehittäjästä. Haastatteluissa korostettiin, että palvelumuotoilija ja UX-suunnittelija toimivat usein erillisinä rooleina, mutta heidän työnsä on tiiviisti yhteydessä toisiinsa. Palvelumuotoilijan työ on

kriittistä projektin alkuvaiheessa, kun taas UX-suunnittelijat ottavat vastuun myöhemmässä vaiheessa, varmistamalla, että palvelumuotoilijan luomat konseptit toteutetaan käytännössä ja ne palvelevat käyttäjiä parhaalla mahdollisella tavalla.

”Monesti projektin alussa on palvelumuotoilija, jolta projekti sitten siirtyy UI/UX designereille” – Suunnittelija 3

Tämä yhteistyömalli varmistaa, että palveluiden suunnittelu on kokonaisvaltaista ja käyttäjäkeskeistä, mikä on tärkeää onnistuneiden palveluinnovaatioiden kehittämisessä. Isoissa projekteissa suunnittelutiimi voi koostua useista suunnittelijoista ja tiimin koko vaihtelee projektin tarpeiden mukaan. Esimerkiksi yhdessä haastattelussa kerrottiin projektista, jossa oli 100 hengen tiimi ja suunnitteluporukka koostui 5–10 suunnittelijasta tilanteen mukaan.

”100 hengen tiimissä meillä oli semmonen suunnitteluporukka, siinä oli 5–10 designeria riippuen tilanteesta” – Suunnittelija 2

Toisessa projektissa palvelumuotoilu oli tehty omana projektinaan ennen kehitysprojektin alkamista, mikä varmisti, että kaikki käyttäjäkokemukseen ja palvelukonseptiin liittyvät seikat oli huolellisesti suunniteltu ja dokumentoitu etukäteen.

”Joissain projekteissa se palvelumuotoilu on tehty omana projektinaan” – Suunnittelija 2

Suuremmissa projekteissa palvelumuotoilija ja UX-suunnittelija työskentelevät tiiviissä yhteistyössä ja voivat joustavasti vaihtaa rooleja tarpeen mukaan. Tämä varmistaa, että palvelumuotoilun ja UX-suunnittelun välillä on saumaton integraatio ja että kaikki suunnittelutyön osa-alueet tulevat katetuiksi. Tämä joustavuus ja yhteistyö ovat avainasemassa suurten projektien onnistumisessa, sillä ne mahdollistavat monipuolisen osaamisen hyödyntämisen ja suunnittelutyön laadun varmistamisen koko projektin ajan.

”Esimerkiks tässä isommassa projektissa missä mä tällä hetkellä oon niin palvelumuotoilija on ollut koko ajan mukana ja mä oon ollut UI/UX-designerina niin oon saattanut tehdä palvelumutoilua ja palvelumuotoilija on vastaavasti tehnyt mun hommia eli UI/UX-suunnittelua.” – Suunnittelija 3

Jatkuva vuoropuhelu ja yhteistyö suunnittelijoiden ja kehittäjien välillä ovat ratkaisevia onnistuneen lopputuloksen saavuttamiseksi. On tärkeää varmistaa, että suunnittelijat ovat

käytettävissä tukemaan kehitystiimiä ja vastaamaan kysymyksiin, jotta voidaan välttää väärinkäsitykset ja varmistaa, että projektin kaikki osat toimivat saumattomasti yhteen. Haastatteluista kävi ilmi, että UI/UX-suunnittelija on tärkeä osa ketterää kehitystiimiä, mutta suunnittelijan tarpeellisuus riippuu täysin projektin luonteesta. Monissa projekteissa UX-suunnittelijoita on ollut mukana useampia, mutta kaikissa projekteissa heidän panoksensa ei ole ollut välttämätön. Palvelumuotoilija (engl. service designer) ja UX-suunnittelija toimivat usein erillisinä rooleina, mutta niiden tiivis yhteistyö on elintärkeää. Palvelumuotoilija huolehtii projektin alkuvaiheen suunnittelusta, kun taas UX-suunnittelija varmistaa, että palvelumuotoilun luomat konseptit toteutetaan käytännössä. Suuremmissa projekteissa suunnittelutiimi voi koostua useista henkilöistä ja rooleja voidaan joustavasti vaihtaa tarpeen mukaan, mikä mahdollistaa kattavan ja käyttäjäkeskeisen suunnittelun.

4.3 Suunnittelutyön sisältö

Kolmas teema käsittelee ja rajoittuu kertomaan suunnittelutyön sisällöstä. Haastatteluista ilmeni muutamia eri rooleja ja tehtäväalueita, joita suunnittelijat tekevät. Prototyyppejä ja rautalankamalleja luodaan usein Figma-työkalulla, mikä auttaa visualisoimaan käyttöliittymän toimintaa ennen varsinaista kehitystä. Käyttäjävirtojen suunnittelu ja ymmärtäminen ovat myös keskeisessä roolissa, sillä ne auttavat hahmottamaan, kuinka käyttäjät liikkuvat sovelluksessa ja mitkä toiminnot ovat heille olennaisia.

”Figma prototyyppeja, user flown tekemistä ja ymmärtämistä.” – Testaaja

Lisäksi suunnittelutyöhön kuuluu paljon käyttäjähaastatteluja ja käyttäjien seurantaa käytännössä, jotta voidaan nähdä, miten sovellusta oikeasti käytetään ja missä mahdolliset ongelmakohdat sijaitsevat. Tämä käytännön tieto on arvokasta, sillä se mahdollistaa suunnitteluratkaisujen hienosäädön ja parantaa sovelluksen käytettävyyttä.

”Hyvin paljon oon saanut tehdä käyttäjähaastatteluja ja sit oon päässy seuraa käyttäjiä, että miten ihmiset oikeasti käyttää sitä sovellusta ja missä on kipupisteitä.” – Konsultti

Suunnittelutyöhön liittyy myös palvelumuotoilu, joka aloitetaan usein projektin alkuvaiheessa. Palvelumuotoilussa keskitytään palvelun konseptin luomiseen käyttäjien kokemusten perusteella. Tämä vaihe on tärkeä, sillä se määrittää palvelun arvontuotannon peruspilarit ja varmistaa, että kehitys perustuu todellisiin käyttäjätarpeisiin ja odotuksiin.

”Joskus saa lähteä hyvinkin sieltä ylätasolta miettimään, että mitä palvelua me nyt ollaan tekemässä ja miksi ja mitä mikä, eli se niinku arvon tuotto on ja lähteä tekemään sitä palvelumuotoilua sieltä ylhäältä asti työpajoilla ja sitten loppukäyttäjiä haastattelemalla

ja tutkimalla ylipäätään sitä kenttää. Ymmärryksen luomisen jälkeen lähdetään luomaan figmassa sprinttiä paria edellä rautalankojen, kuvien ja määrittelyjen tekeminen." – Suunnittelija 2

Dokumentointi on olennainen osa suunnitteluprosessia, sillä se varmistaa, että kaikki tiimin jäsenet ymmärtävät suunnitteluratkaisut ja voivat palata niihin tarvittaessa. Dokumentointi auttaa myös kommunikoimaan suunnittelun tavoitteet ja perusteet kehitystiimille, mikä parantaa yhteistyötä ja vähentää väärinkäsityksiä. Työpajoilla on tarkoitus lisätä käyttäjäymmärrystä, sillä niissä pääsee työskentelemään loppukäyttäjien kanssa.

"Alkuvaiheessa tehty yhteissuunnittelu workshoppeja näiden tulevien käyttäjien kanssa, jonka jälkeen tehtiin konseptisuunnitteluvaihe, prototyyppejä ja dokumentaatiota." – Suunnittelija 3

Kaksi haastateltavaa kertoi tekevänsä kaikenlaista suunnittelutyötä johtuen aikaisemmasta taustasta tai projektin luonteesta ja pituudesta.

"Johtuen mun entisestä taustasta, kun oon ollut pienessä digistudiossa nii mä oon aikalailla semmoinen generalisti designer, että teen vähän kaikenlaista, mutta silleen niin kun pääpainoitus on siinä käyttöliittymän suunnittelussa ja sitten käytettävyystestauksessa." – Suunnittelija 1

"Nykyisessä projektissa joka on tämmönen useamman vuoden kestävä jossa tehdään ammattilaistyökalua erittäin isolle toimijalle, niin kaikkea mitä niin kun softaprojektin designiin ylipäätään liittyy niin on tullut tehtyä." – Suunnittelija 3

Kehittäjät kertoivat, etteivät ole varsinaisesti päässeet suunnittelutyöhön mukaan. Yksi haastateltava, joka toimii kehittäjänä, oli päässyt luomaan pieniä ja nopeita prototyyppejä, kun suunnittelijoita ei ollut projektissa saatavilla.

"Selaimessa oon inspectillä saattanut tehdä vaan jonku pikku proton tai sitten vaa piirtää joku hommeli ja siinä menee vaan yleensä se muutama minuutti." – Kehittäjä 4

Toisella kehittäjistä oli myös kokemusta projekteista, joissa front end -kehittäjät olivat päässeet suunnittelutyöhön mukaan.

"Toistaiseksi en ole päässyt, mutta kyllä aikaisemmin on fronttisuunnittelijat ja designerit ovat myös keskenään suunnitelleet." – Kehittäjä 5

Työalueisiin kuuluvat muun muassa käyttöliittymäsuunnittelu etukäteen rautalankamalleilla, kuvilla, prototyypeillä ja määrittelyillä. Palvelun arvontuottoa on myös mietitty projektin juuritasolla palvelumuotoilua hyödyntämällä. Käyttäjäymmärrystä on lisätty käyttäjävirran (engl. user flow) tekemisellä sekä haastatteluiden ja työpajojen (engl. workshop) järjestämisellä.

4.4 Suunnittelu sekä kehitys samanaikaisesti

Neljännessä teemassa käsitellään suunnittelu- ja kehitystyön toteutus samanaikaisesti. Aluksi käydään läpi haastateltavien omia kokemuksia, jonka jälkeen kerrotaan hyötyjä sekä haittoja suunnittelun ja kehityksen samanaikaisesta työskentelystä.

4.4.1 Haastateltavien kokemuksia

Joillakin haastateltavista oli kokemusta sekä suunnittelu- että kehitystyöstä, jota on jouduttu tekemään samanaikaisesti. Yksi haastateltavista mainitsi, että hänen kollegansa oli front end -kehittäjä, joka vastasi samaan aikaan sekä suunnittelu- että kehitystyöstä.

”On mulla kollega, joka on niinku fronttidevari, mikä tekee molempia asioita yhtä aikaa” – Suunnittelija 1

Muutamat haastateltavat kertoivat omasta kokemuksestaan, kuinka he olivat uransa alkuvaiheessa, esimerkiksi mainostoimistoissa, tehneet kokonaisia verkkosivuprojekteja konseptin suunnittelusta aina koodaukseen saakka. Tämä antoi heille laajan osaamisen ja ymmärryksen molemmista osa-alueista.

”Oon alottanu urani mainostoimistoissa ja siellä mä tein niinku webbisivuja alusta loppuun eli koko paketin konseptin suunnittelusta koodaukseen. Vielä viisi vuotta sitten tuli jonkin verran tehtyä käyttöliittymäkomponentteja, jonka takia tuli jonkin verran koodia värmättyä. ” – Suunnittelija 3

Joissain projekteissa, joissa ei ollut erillistä UX-suunnittelijaa, kehittäjät ottivat hoitaakseen käyttöliittymien suunnittelun. Tämä tarkoitti, että kehittäjät suunnittelivat kokonaisia käyttöliittymiä, mikä vaati heiltä suunnittelutaitoja teknisen osaamisen lisäksi.

”Semmosissa projekteissa missä ei oo erillistä UX-ihmistä niin kehittäjät on ollu just niitä henkilöitä, oon itsekin ollut suunnittelemassa kokonaisia käyttöliittymiä” – Kehittäjä 1

Haastateltavan eräässä projektissa, joissa haettiin uusia ratkaisuja eikä asiakkaalla ollut vielä tarkkaa käsitystä siitä, mitä haluttiin, kehittäjät joutuivat yhdessä hahmottelemaan ratkaisuja tekemällä prototyyppejä.

”Kyllä olen ollut. Se on ollut sellainen projekti missä ollaan oltu kokeilemassa tai hakemassa jotakin uutta ratkaisua josta asiakaskaan ei ole välttämättä oikein tiennyt vielä, että mitä haluaa ja siinä ollaan sitten hahmoteltu vaan tekemällä” – Kehittäjä 2

4.4.2 Hyödyt

Henkilö, joka toteuttaa suunnitelmia ja koodia samanaikaisesti voi johtaa tehokkuuden ja yhtenäisyyden paranemiseen, sillä henkilö pystyy välittömästi soveltamaan suunnitelmiaan koodin muodossa ilman, että tietoa tarvitsee siirtää toisen henkilön tai tiimin välityksellä. Tämä vähentää virhemahdollisuuksia ja nopeuttaa kehitysprosessia, koska henkilö on täysin perillä sekä suunnitelmien yksityiskohdista että koodin toteutuksesta.

”Fronttikoodin ymmärtämisestä tietylle tasolle tuo paljon hyötyä kommunikoinnissa sekä optimoinnissa” – Suunnittelija 2

”Se myös vaatii enemmän kommunikaatiotarvetta, jos on erillinen suunnittelija. Siinä tulee mahdollisesti palloteltua.” – Kehittäjä 1

Toinen kehittäjistä puhui kokemuksestaan positiivisesti, jossa hänen projektissaan oli mukana henkilö, joka tuotti suunnitelmia sekä front end -koodia samanaikaisesti.

”Kyllä siitä oli ehdottomasti hyötyä, kun se sai tulosta niin nopeasti aikaan niin pystyi kokeilemaan eri mahdollisuuksia” – Kehittäjä 2

4.4.3 Haitat

Toisaalta tässä toimintatavassa on myös merkittäviä haittoja. Ajan käytön haasteet ovat yksi keskeisimmistä ongelmista. Suunnittelun ja koodauksen yhdistäminen voi johtaa siihen, että kummankin tehtävän hoitaminen kärsii, kun aika ja resurssit jakautuvat epätasaisesti.

”Hänestä voi tulla pullonkaula projektille, jos hänet on allokoitu devaajaksi ja hän käyttää liikaa aikaa suunnitteluun, eri asia jos hänelle on allokoitu tarpeeksi aikaa molemmille” – Suunnittelija 2

Myös jos henkilö saattaa keskittyä liikaa koodin yksityiskohtiin, voi olla, että laajemmat suunnittelunäkökohdat jäävät huomiotta, tai päinvastoin.

”Aivot menee jännästi helposti siihen ajatusmaailmaan, että jos sulla on joku syvempi ymmärrys koodista, että miten se toteutetaan, niin sitten sitä lähtee ajattelemaan enemmän sen toteutettavuuden kannalta, mitä käytettävyyden kannalta.” – Suunnittelija 1

Lisäksi loppukäyttäjien unohtaminen on merkittävä riski. Kun yksi henkilö vastaa sekä suunnittelusta että toteutuksesta, voi olla haastavaa pitää mielessä loppukäyttäjien tarpeet ja näkökulmat.

”Haittoja tulee mieleen se, että semmonen insinöörikäyttöliittymä on ihan oikea termi, että se on kuitenkin ihan totta, että UX-suunnittelijat ovat taitavampia suunnittelemaan sellasia selkeämpiä käyttöliittymiä, jotka ovat intuitiivisia. Kehittäjien suunnitelmat soveltuu sellaseen yksinkertaiseen, esim. lomakkeiden tekoon” – Kehittäjä 1

Tämä voi johtaa tuotteisiin, jotka eivät täysin vastaa käyttäjien odotuksia tai tarpeita. Loppukäyttäjien unohtaminen voi myös heikentää käyttäjäkokemusta ja johtaa tyytymättömyyteen tuotteen käytössä.

Kuten kirjallisuuskatsauksen kohdassa 2.5 mainittiin, näitä haasteita koettiin enemmän kuin hyötyjä. Tämä viittaa siihen, että vaikka suunnitelmien ja koodin samanaikainen toteutus voi olla tehokasta tietyissä tilanteissa, on tärkeää tunnistaa ja hallita siihen liittyvät riskit ja haasteet, jotta lopputulos olisi mahdollisimman laadukas ja käyttäjäystävällinen.

4.5 Full stack -suunnittelu

Viides teema on rajattu käsittelemään full stack -suunnittelua ja full stack -suunnittelijan roolia. Seuraavaksi käydään läpi, että kuinka tunnettu full stack -suunnittelijan rooli oli haastateltavien keskuudessa ja mitä hyötyjä tai haittoja roolista olisi ketterässä kehityksessä.

4.5.1 Ennakkoluulot

Haastattelun aikana haastateltavalle määriteltiin full stack -suunnittelijan määritelmä suullisesti. Kukaan haastateltavista ei ollut kuullut full stack -suunnittelijan roolista aikaisemmin, vaikkakin joillain haastateltavista olikin ennakkoluuloja.

”Ennakkoluulona oli ennen haastattelua, että se ois semmonen generalisti” – Suunnittelija 1

Jotkut haastateltavat vertasivat roolia unicorn designeriin, korostaen sen harvinaisuutta ja monipuolisuutta.

”En, full stack -developereista oon, mutta en suunnittelijasta. Mutta se on varmaan just se yksisarvinen mihin viittasin aikaisemmin” – Suunnittelija 2

Yksi haastateltavista ajatteli ennen haastattelua, että full stack -suunnittelija vastaisi koko suunnitteluprosessin vaiheista alusta loppuun.

”En ole kuullut, mutta ennen haastattelua ajattelin, että se olisi designin työvaiheet kattava rooli” – Suunnittelija 3

Lisäksi kävi ilmi, että vaikka termi "full stack -suunnittelija" ei ollut tuttu, vastaavanlaista sisältöä saattaisi löytyä jonkin toisen tittelin alta. Yleisesti ottaen roolin käsitys jäi kuitenkin epäselväksi ja tarkentamista vaativaksi.

”Ei ehkä tolla termillä, että ehkä toi sisältö tungettuna jonku toisen tittelin alle” – Konsultti

4.5.2 Full stack -suunnittelijan hyödyt

Full stack -suunnittelijan rooli koettiin hyödylliseksi, koska se voi varmistaa, että front end toteutetaan kerralla oikein, mikä vähentää virheitä ja uudelleentyöstämisen tarvetta.

”Frontti tulisi kerralla oikein” – Suunnittelija 1

Rooli vähentää myös väärinkäsityksiä suunnittelun ja teknisen toteutuksen välillä, mikä parantaa projektin laatua. Syvällinen ymmärrys molemmista osa-alueista tuo uusia ajattelumalleja ja voi vähentää suunnitteludokumenttien ylläpidon tarvetta. Lisäksi mainittiin, että rikkinäisen puhelimen ilmiö vähenisi, sillä full stack -suunnittelija ymmärtäisi sekä suunnittelun että teknisen toteutuksen, mikä estäisi väärinkäsityksiä.

”Ei tuu rikkinäistä puhelinta, että niinku designer saattaa huomata vasta palaverissa, että ai niin, että tämmönenki on niinku mahdollista, ei tää käyny ees mielessäkään teknisessä mielessä” – Suunnittelija 2

Haastateltavat korostivat myös, että kokonaisvaltainen rooli, jossa on syvällinen ymmärrys sekä suunnittelusta että teknisestä toteutuksesta, voisi tuoda uusia ajattelumalleja ja parantaa projektin laatua.

”Kyllä joo tietyllä tavalla. Kokonaisvaltainen rooli, jolla on ymmärrys niin kun suunnittelutyöstä ja se sellanen tietynlainen ajattelumalli ylipäätään.” – Suunnittelija 3

Erään haastateltavan mukaan suunnitteludokumenttien ja prototyyppien ylläpidon tarve voisi vähentyä, koska ideat voitaisiin viedä suoraan käyttöliittymään.

”Suunnitteludokkareiden ja protojen ylläpito vähenisi suoraan ja sieltä saisi ne ajatukset suoraan käyttöliittymään” – Kehittäjä 1

Jos tiimissä olisi useampi full stack -suunnittelija, he voisivat sparrata toisiaan ja kehittää käyttöliittymää yhdessä. Yksi haastateltavista koki, että nykyisessä työelämässä tällaista osaamista tarvitaan roolista riippumatta.

”Jos tiimissä on enemmän kuin yks tuollaista henkilöä niin ne voisi keskenään sparrailla näitä ideoita ja rakentaa sitä UI:ta samalla.” – Kehittäjä 2

4.5.3 Full stack -suunnittelijan haitat

Yllä mainittujen hyötyjen lisäksi myös haittoja ilmeni haastatteluista useita. Full stack -suunnittelijan rooli voi aiheuttaa haasteita, sillä keskittyminen moneen eri tehtävään samanaikaisesti ja vastuun jakaminen ovat vaikeita.

”Ei pysty irtoamaan kaikkeen yhtä aikaa ja ottaa niinku koppia joka asiasta” – Suunnittelija 2

Yksi haastateltavista mainitsi, että projektin edetessä on todella vaikea suoriutua kaikista työtehtävistä kunnolla, sillä tehtävien määrä kasvaa projektin edetessä. Näin full stack -suunnittelijasta voisi tulla pullonkaula projektille, sillä ajankäytölliset haasteet nousevat esiin.

”Projektin alussa varmasti helppoa, kun ominaisuuksia on vähän, mutta projektin edetessä, kun ominaisuuksia tulee lisää niin näät vain kun häntä siellä perässä kasvaa ja tarkistat että kaikki toimii ja jos tämän lisäksi vielä devaisi itse ja pitäisi korjata kokoajan samalla kun niitä ominaisuuksia joku testaisi sekä suunnitella jo seuraavaa ominaisuutta,

niin se on siinä varmaan semmonen pullokaula varsinkin isommissa projekteissa” – Suunnittelija 2

Lisäksi, jos full stack -suunnittelija ei ole halukas tekemään yhteistyötä muiden kanssa ja toimii enemmän yksin, se voisi olla haitaksi tiimityölle. Tämä voisi johtaa siihen, että tällainen henkilö ei sopeudu hyvin tiimiin eikä jaa osaamistaan muiden kanssa.

”Voi olla myös haitaksi riippuu siitä, että miten sitä osaamista käytetään. Jos haluaa olla sankari designer, joka ei suostu tekemään yhteistyötä muiden kanssa niin sellasta en kyllä haluaisi.” – Suunnittelija 3

Myös roolin laaja-alaisuus saattoi olla ongelmallista, sillä se voisi olla perinteistä full stack -kehittäjän roolia vastaan, jossa keskitytään yhden ominaisuuden toteuttamiseen alusta loppuun. Tämä voisi aiheuttaa ristiriitoja ja tehokkuusongelmia projektin edetessä.

”Sotii vähän full stack -kehittäjän roolia vastaan, että siinä tehdään yleensä niinku kokonainen ominaisuus alusta loppuun käyttöliittymästä sinne ehkä ylläpitoon asti” – Kehittäjä 1

Full stack -suunnittelijan roolia pidettiin tuntemattomana, mutta sitä kuvailtiin generalistina, joka hallitsee monia eri osa-alueita. Roolin eduiksi nähtiin vähentyneet virheet, parempi suunnittelun ja teknisen toteutuksen yhteys sekä tiimityön tehostuminen. Haittoina mainittiin keskittymisen hajautuminen, haasteet tiimityössä ja ristiriidat perinteisten roolien kanssa. Full stack -suunnittelijan rooli koettiin hyödyllisimpänä pienissä projekteissa tai yrityksissä, joissa ei ole välttämättä useita eri osaajia.

4.6 Projektityypin vaikutus full stack -suunnittelulle

Uutena teemana nousi projektityypin vaikutus full stack -suunnittelijan tarpeellisuudelle. Projektityypillä tarkoitetaan tässä kontekstissa projektin kokoa, kestoa sekä kuinka monta suunnittelijaa projektissa on. Full stack -suunnittelija voisi tuoda arvokkaita oivalluksia ja ideoita, koska hänellä on laaja ymmärrys sekä suunnittelusta että teknisestä toteutuksesta. Tällaisissa projekteissa hänen monipuolinen osaamisensa voisi edistää uudenlaisten ratkaisujen kehittämistä ja toteuttamista.

”Jos pitäisi keksiä jotain innovatiivista, että haluttaisiin keksiä jotain uutta markkinarakoa, niin sittenhän tämmöisestä yksisarvisesta on niinku varmasti hyötyä, et hän pystyy oivaltamaan siellä jotakin, että mitä on mahdollista tehdä” – Suunnittelija 2

Toisaalta haastateltavat myös huomauttivat, että pitkissä ja monimutkaisissa projekteissa, joissa kehitystyö vie paljon aikaa, full stack -suunnittelijan työajan jakaminen eri osa-alueiden kesken voi olla haastavaa.

”Joo kyllä siitä ehdottomasti hyötyä olisi, mutta toisaalta ne projektit, jossa kehitystyö saattaa viedä aikaa niin se työajan jakaminen voi olla hankalaa eri osa-alueisiin ja ajankäyttö voi tulla pullonkaulaksi.” – Kehittäjä 2

Tämä voi johtaa siihen, että aikataulut kiristyvät ja ajan käyttö muodostuu pullonkaulaksi, mikä saattaa hidastaa projektin etenemistä. Yksi haastateltava suositteli, että isommissa projekteissa olisi hyvä olla erillinen tuotesuunnittelija, jotta nämä ajanhallintahaasteet vältettäisiin.

”Isommissa projekteissa on hyvä olla tuote designer, koska muuten se on pitemmän päälle hitaampaa rakennella sitä UI:ta ja siinä samalla tehdä designia.” – Kehittäjä 2

Lisäksi tuotiin esille, että erityisesti suuremmissa yrityksissä, olisi tärkeää, että full stack -suunnittelija itse ilmaisee, mihin tehtäviin hän haluaa keskittyä. Tämä vähentäisi riskiä, että jonkin osa-alueen laatu kärsii, jos yksi henkilö yrittää hoitaa kaikkia tehtäviä yksin.

”Jos ei puhuta startupeista, niin toivoisin, että tämä henkilö kertoisi mitä haluaa tehdä, koska siinä voi helposti jokin osa-alue kärsiä, jos yksi henkilö tekee kaiken” – Konsultti

Näin varmistettaisiin, että projektin eri osa-alueet saavat riittävästi huomiota ja asiantuntemusta. Haastatteluista muodostui käsitys, että nimenomaan pienissä projekteissa yhdestä roolista olisi hyötyä, sillä henkilö voisi hoitaa useampaa eri osa-aluetta kerralla eikä laatu kärsisi, sillä projektin mittakaavat eivät ole suuret.

”Mitä pienempi projekti on, nii sitä hyödyllisempi on olla yksi rooli” – Testaaja

Full stack -suunnittelijan rooli koettiin varsin hyödyllisenä innovatiivisissa projekteissa, joissa pyritään löytämään uusia markkinarakoja. Pienemmissä projekteissa sekä yrityksissä tämänkaltainen monipuolinen osaaminen voi tehdä työnteosta tehokkaampaa. Laajoissa sekä haastavammissa projekteissa työn jakaminen eri tehtävien välillä voi muodostua haasteeksi, mikä saattaa hidastaa edistymistä. Suuremmissa yrityksissä on tärkeää, että suunnittelija itse määrittelee, mihin tehtäviin hän keskittyy, jotta työn laatu ei kärsi.

5 Johtopäätökset ja pohdinta

Tässä tutkielmassa selvitettiin, kuinka tarpeelliseksi full stack -suunnittelijan rooli koetaan ketterässä ohjelmistokehityksessä työntekijän näkökulmasta. Tässä luvussa kerrotaan tutkimuksen keskeiset tulokset ja verrataan niitä aikaisempiin tutkimuksiin. Lisäksi pohditaan tämän tutkielman onnistumista sekä luodaan katsaus jatkotutkimusmahdollisuuksiin.

5.1 Tutkimustulokset

Kirjallisuuskatsauksessa muodostettiin käsitys, ettei full stack -suunnittelijalla ole vielä vakiintunutta roolia ketterässä ohjelmistokehityksessä, eikä sen perusteella voida päätellä onko roolille tarvetta tai mitkä sen hyödyt ja haasteet ovat. Haastatteluiden perusteella saadun käsityksen perusteella voidaan vahvistaa tieto siitä, ettei rooli ole vakiintunut termi ketterässä ohjelmistokehityksessä. Kukaan haastateltavista ei ollut kuullut roolista aikaisemmin, mutta osa aavisteli mitä sillä voitaisiin tarkoittaa. Abid ym., [2023], Botha, [2018], Kulchinskiy [2021] ja Ng, [2024] puhuivat yksisarvissuunnittelijasta (engl. unicorn designer), sillä on vaikea löytää suunnittelijaa, joka osaa sekä ohjelmoida että suunnitella. Yksi haastateltavista viittasikin unicorn designeriin puhuttaessa full stack -suunnittelijasta.

Tutkimuskysymykseen vastatessa voidaan sanoa, että full stack -suunnittelijan rooli voi olla erittäin hyödyllinen pienissä tiimeissä, projekteissa tai yrityksissä. Näissä tilanteissa yksi henkilö, joka pystyy hoitamaan sekä käyttöliittymän (UI/UX) suunnittelun että taustajärjestelmien teknisen toteutuksen, voi merkittävästi nopeuttaa kehitysprosessia ja tuoda joustavuutta tiimin toimintaan. Pienemmissä projekteissa tämä moniosaajuus voi johtaa sujuvampaan yhteistyöhön ja yhtenäisempään lopputulokseen. Kuitenkin suuremmissa projekteissa full stack -suunnittelijan rooli voi koitua ongelmaksi. Ajankäytön haasteet korostuvat, kun yhden henkilön on hallittava monia erilaisia osa-alueita, mikä voi johtaa siihen, että osa-alueiden syvällinen ymmärrys ja toteutus kärsivät. Koska on epätodennäköistä, että yksi henkilö olisi kaikessa yhtä taitava, voi lopputulos jäädä keskinkertaiseksi.

Haastateltavat kokivat, että full stack -suunnittelijasta olisi hyötyä projektille. Kirjallisuuskatsauksessa Botha [2018], Canziba [2018] ja Kulchinskiy [2021] kertoivat muun muassa, kuinka full stack -suunnittelijalla on mahdollisuus ymmärtää eri alueet ja projektin kokonaiskuva sekä projektissa on mahdollisuus säästää rahaa ja aikaa. Haastateltavien mielestä full stack -suunnittelijalla voitaisiin saada front end -koodi oikein nopeammin, eikä kehittäjien sekä suunnittelijoiden välistä kommunikaatiota tarvitse huomioida, sillä

yksi henkilö tekisi ne molemmat. Heidän näkemyksensä mukaan full stack -suunnittelijan rooli on kokonaisvaltainen, ja siihen sisältyy laaja ymmärrys ja ajattelutapa projekteista, mikä vastaa kirjallisuuskatsauksessa esitettyjä näkemyksiä.

Puhuttaessa full stack -suunnittelijan haitoista ovat haastattelut ja kirjallisuuskatsaus linjassa toistensa kanssa. Bothan [2018], Canziban [2018] ja Kulchinskiyn [2021] katsauksissa nostettiin esille full stack -suunnittelijan ajanhallinnan vaikeus. Haastateltavat henkilöt olivat samaa mieltä ja kokivat, että full stack -suunnittelijan olisi vaikea olla tekemässä useaa eri asiaa samanaikaisesti sekä projektin laajetessa saattaisi full stack -suunnittelijasta tulla pullon kaula projektille. Full stack -suunnittelija koettiinkin hyödylliseksi vain silloin, kun projektin laajuus on pieni tai että full stack -suunnittelija työskentelisi startup- tai pienyrityksissä.

Kirjallisuuskatsauksessa kerrottiin, kuinka haasteita ilmenee käyttäjäkeskeisessä suunnittelussa integroituna ketterään kehitykseen. Haasteita olivat muun muassa riittämätön viestintä sekä valtataistelu käyttäjäkokemussuunnittelijoiden ja kehittäjien välillä [Argumanis ym., 2020; Curcio ym., 2019]. Haastateltavien henkilöiden mukaan yhteistyön onnistumiseen vaikuttaa hyvin paljon projektin luonne. Isommissa projekteissa, jotka kestävät vuoden tai enemmän yhteistyö koettiin lähes poikkeuksetta onnistuneeksi, sillä kehittäjien ja suunnittelijoiden välille on löytynyt hyvä yhteys. Haastateltavat kertoivat, että parhaiten yhteistyö on onnistunut silloin, kun suunnittelutyö on osana kehitystiimiä sekä etupainotteista suunnittelutyötä ei tehdä liikaa. Silva ym. [2011] ehdottivat, että pienen suunnittelutyön tekeminen etukäteen, tiiviiseen yhteistyöhön keskittyminen sekä yhden sprintin edellä työskentely kehittäjistä auttavat varmistamaan, että ketterän kehityksen sekä käyttäjäkeskeisen suunnittelun integroimisen välille tulee mahdollisimman vähän haasteita.

Haastateltavat kertoivat, kuinka he pitävät tuoteomistajan (PO) sekä käyttäjäkokemussuunnittelijan välisen yhteistyön tärkeänä osana projektin onnistumisen kannalta. Kuusinen [2015] havaitsi kolme erilaista yhteistyötyyppiä käyttäjäkokemussuunnittelijoiden ja ketterän kehityksen muiden roolien väliltä, joista toiseksi parhaimpana yhteistyötyyppinä hän kertoi olevan tuoteomistajan sekä käyttäjäkokemussuunnittelijoiden läheinen yhteistyö. Kuusisen mukaan tuoteomistajan tehtävänä on pitää kehitysprosessihallinnassa koko kehitysprosessin ajan, jolloin kokemattomammat tiimit voisivat hyötyä paremmin tästä yhteistyötavasta, kuin esimerkiksi kehittäjien ja käyttäjäkokemussuunnittelijoiden välisestä yhteistyöstä. Tuoteomistajan ja käyttäjäkokemussuunnittelijan välinen yhteistyö

toimii myös silloin, jos projektin laajuus on epäselvä tai ei ole tiedossa. Kuusinen kuitenkin myös muistuttaa, että tämä yhteistyötyyppi toimii vain silloin, kun tuoteomistaja ymmärtää sekä arvostaa käyttäjäkokemustyötä.

Näiden roolien lisäksi palvelumuotoilijan rooli nousi esille haastatteluissa, mitä kirjallisuuskatsauksessa ei otettu esille. Palvelumuotoilulla tarkoitetaan suunnitteluperusteista menetelmää palveluinnovaatioihin, jossa painopiste on ihmisten kokemusten ymmärtämisessä ja tämän tiedon hyödyntämisessä palveluiden parantamisessa [Sangiorgi ym., 2017]. Haastatteluissa kävi ilmi, että palvelumuotoilijan rooli on merkittävä erityisesti projektien alkuvaiheessa, jolloin määritellään palvelukonseptia. Usein palvelumuotoilija aloittaa projektin tutkimalla ja muodostamalla palvelukonseptin, joka pohjautuu käyttäjien kokemusten syvälliseen ymmärtämiseen. Tämä varmistaa, että palvelu vastaa käyttäjien todellisia tarpeita ja odotuksia. Kun palvelukonsepti on määritelty, varsinainen suunnittelutyö siirtyy UX-suunnittelijoille, jotka keskittyvät käyttöliittymän ja käyttäjäkokemuksen yksityiskohtaiseen suunnitteluun. Tämä vaiheistus mahdollistaa sen, että suunnitteluprosessi on johdonmukainen ja perustuu selkeään ymmärrykseen käyttäjäkokemuksesta. [Stickdorn ym., 2018]

Tutkimuksen tulokset osoittavat, ettei full stack -suunnittelijan rooli ole vielä vakiintunut termi ketterässä ohjelmistokehityksessä. Rooli kuitenkin koettiin hyödylliseksi erityisesti pienissä tiimeissä sekä startup-yrityksissä monipuolisen osaamisen johdosta, joka mahdollistaa tehokkaammat resurssien käyttömahdollisuudet. Kuitenkin suurissa projekteissa full stack -suunnittelijalla voi olla ajankäytöllisiä haasteita suoriutua annetuista työtehtävistä ja rooli koetaan olevan ristiriidassa perinteisten roolien, kuten full stack -kehittäjän tai käyttäjäkokemussuunnittelijan kanssa.

5.2 Tämän tutkimuksen pohdinta

Tämä tutkielma eteni lähes kokonaan suunnitelmien mukaisesti ja tarpeeksi sujuvasti. Alussa ongelmia oli lopullisen tutkimusaiheen rajaamisessa sillä en ollut varma, kenen näkökulmasta halusin tutkia aihetta. Aluksi lähdin miettimään aihetta yrityksen näkökulmasta, mutta hyvin pian vaihdoin työntekijän näkökulmaan, sillä se tuntui itselleni luontevammalta näkökulmalta. Itse tutkielman kirjoittaminen alkoi kirjallisuuskatsauksesta, jota ennen olin huomannut, ettei full stack -suunnittelijasta ole julkaistu vertaisarvioituja artikkeleita juuri ollenkaan. Ketteristä menetelmistä sekä käyttäjäkeskeisestä suunnittelusta oli kuitenkin tieteellistä tekstiä paljon, joiden perusteella pystyin muodostamaan nykytilanteen käyttäjäkeskeisen suunnittelun tilasta ketterässä kehityksessä. Tiedonkeruuprosessin ajan pystyin muodostamaan laajan näkemyksen käyttäjäkeskeisestä suunnitte-

lusta integroituna ketterään kehitykseen, mitä rooleja ketterässä käyttäjäkokemussuunnittelussa on sekä mitä full stack -suunnittelijalla voitaisiin tarkoittaa ja mitä hyötyjä tai haittoja roolista on.

Teemahaastattelurunko ja kysymykset saatiin luotua kirjallisuuskatsauksen perusteella. Kysymyksiä viilailtiin useampaan kertaan ja lopullisesta versiosta poistettiin muutama kysymys, sillä niissä oli toistoa. Lopullinen versio saatiin valmiiksi ennen haastatteluiden alkamista, joten kaikilta haastateltavilta saatiin kysyttyä samat kysymykset. Varsinaisten haastateltavien etsimistä ei tarvinnut tehdä juuri ollenkaan. Keskustelin gradun aiheestani Solitalla työskentelevän henkilön kanssa ja vajaan puolen tunnin keskustelun jälkeen kysyin, suostuisiko hän sekä osa hänen kollegoistaan osallistumaan tämän tutkielman haastatteluihin, johon hän vastasi myöntävästi. Kun haastatteluiden aika lähestyi, laiton tapaamalleni henkilölle sähköpostia, johon hän palasi noin viikon kuluessa. Hän oli löytänyt hänet mukaan lukien kuusi haastateltavaa tutkielmaan, joista viisi osallistui haastatteluihin. Jälkikäteen ajateltuna haastateltavat henkilöt Solitalta vastasivat hyvin tarpeitani, sillä tarvitsin haastateltavaksi henkilöitä, joilla on useamman vuoden kokemusta käyttäjäkeskeisestä suunnittelusta ketterässä ohjelmistokehityksessä. Kaksi muuta haastateltavaa tunsin etukäteen yliopiston kursseilta ja lähestyin heitä Telegram viestisovelluksessa kysyäkseni osallistumista haastatteluihin, joihin molemmat vastasivat myöntävästi. Kurssikavereiden valinta haastatteluun oli tarkoituksen mukaista, vaikkakin valintakriteerit eivät täysin täyttyneet käyttäjäkeskeisen suunnittelun toteutuksesta ketterissä menetelmissä. Koin kuitenkin, että kursseilta saatu oppi oli arvokasta tietoa tälle tutkielmalle.

Haastattelut itsessään menivät hyvin. Ennen ensimmäistä haastattelua, suoritin testipuhelun käyttäen Microsoft Teams -sovellusta todetakseni, että mikrofoni, kamera sekä esitys toimivat oikein. Kuitenkin ensimmäisen esityksen loppupuolella törmäsin ongelmiin, jossa en voinut jakaa kuvaa full stack -suunnittelijan määritelmästä puhelun chat-kenttään. Tämän jälkeen kokeilin jakaa näyttöä, mutta sekin epäonnistui. Molemmat jaot epäonnistuivat siksi, koska MacBookille ei ollut annettu oikeuksia jakaa sisältöä. Tiedostin ennestään, että MacBookissa on tarkat suojaukset ja monet toiminnot vaativat lupien antamisen sovelluksille, jotta niiden kaikki toiminnot olisivat saatavissa. En kuitenkaan tajunnut tarkistaa tätä aikaisemmin. Sovimme haastateltavan kanssa, että kerron suullisesti hänelle full stack -suunnittelijan määritelmän. Loput haastatteluista etenivät kuitenkin suunnitellusti. Yhteneväisyyden vuoksi kerroin myös muille haastateltaville suullisesti full stack -suunnittelijan määritelmän.

Aineistoa kertyi 2 tuntia ja 27 minuuttia. Ajansäästämisen takia haastatteluja ei litteroitu kokonaan, vaan jokaisesta haastattelusta etsittiin tärkeimmät kohokohdat, jotka litteroitiin

teemoittelun luomiseksi. Analysoinnin tuloksena syntyi useita teemoja, joita pystyi vertaamaan kirjallisuuskatsauksessa löydettyihin tuloksiin. Alkuun ongelmaksi nousi liian laajat teemat. Lopullisen teemoittelun kaksi ensimmäistä teemaa olivat aluksi yhdessä, jolloin yhden teeman alla oli liikaa tekstiä ja se teki lukemisesta hankalaa. Jälkeenpäin ajateltuna alateemoja olisi voinut olla vielä enemmän, niin että jokaisen pääteeman alla olisi ollut ainakin kaksi alateemaa, jotta se olisi yhä selkeämpää lukijalle.

Tutkielman tulosten yleistettävyyden suurimpana ongelmana on haastateltavien pieni määrä. Lisäksi viisi haastateltavaa seitsemästä työskenteli samassa yrityksessä. Haastateltavien taustat olivat kuitenkin erilaiset, vaikka yhteisenä tekijänä voidaan pitää työskentely ketterässä ohjelmistokehitys ympäristössä, johon on integroitu käyttäjäkeskeinen suunnittelu. Haastatteluissa olisi silti voitu kysyä yleisellä tasolla, että mitä työtä on tehnyt aikaisemmin tai tarkemmin, että missä rooleissa on työskennellyt aikaisemmin ketterässä kehityksessä. Jos jokainen haastateltava olisi ollut eri yrityksestä, vastauksissa olisi voinut olla enemmän vaihtelevuutta. Tutkielman tuloksia voidaan pitää joka tapauksessa luotettavina, sillä haastattelun tulokset pystyttiin liittämään kirjallisuuskatsaukseen ja haastatteluissa toistuivat samat teemat. Kaiken kaikkiaan tutkielman tekeminen oli mielekästä, sillä aihe oli minulle läheinen ja se opetti minulle mitä full stack -suunnittelijalla tarkoitetaan, miten rooli vaikuttaa ketterässä ohjelmistokehityksessä sekä miten käyttäjäkeskeinensuunnittelu on integroitu ketteriin menetelmiin.

5.3 Jatkotutkimusmahdollisuudet

Full stack -suunnittelijan roolia on tutkittu niukasti sekä roolista on muutenkin todella vähän tieteellistä tekstiä olemassa. Tässä tutkimuksessa keskityttiin löytämään hyötyjä ja haasteita full stack -kehittäjän roolista ketterässä ohjelmistokehityksessä.

Jatkotutkimusmahdollisuutena olisi esimerkiksi tutkia kahta eri tiimiä, jotka tekisivät samanlaista projektia. Kriteereinä projekteille olisivat tiimijako, laajuus, aikataulu sekä resurssit. Ensimmäisessä tiimissä olisi full stack -suunnittelijoita sekä back end -kehittäjiä ja toisessa tiimissä olisi suunnittelijat ja kehittäjät erikseen. Molemmille tiimeille annettaisiin yhtä laaja projekti, jossa on samat ominaisuudet, käyttöliittymä sekä määräaika, jolloin projektin pitää olla valmis. Myös resurssit ovat täysin samat tiimien välillä. Resursseja voi olla muun muassa työvälineet sekä ohjelmistot. Tarkoituksena olisi seurata, kuinka projektit eroaisivat ajankäytöllisesti, laadullisesti ja lopputulokseltaan toisistaan. Mittareita seurantaan voivat olla esimerkiksi koodin laatu, käyttäjäkokemus sekä kuinka nopeaa tietyt tehtävät saatiin suoritettua.

Full stack -suunnittelijan roolin hyödyllisyyttä voitaisiin mitata myös jossain oikeassa toimintaympäristössä, jossa suoritettaisiin projektin seurantaa eri kokoisissa yrityksissä.

Yksi startup, toinen keskisuuri ja kolmas suuri yritys, joissa tiimit voivat olla erikokoisia. Projektille annettaisiin mittareiksi lopullisen tuotteen käyttäjäkokemus, eri roolien yhteistyön sujuvuus ja integraatio sekä käyttöliittymän valmiusaste ennalta määrätyssä ajassa.

6 Yhteenveto

Ketterä kehitys, johon on integroitu käyttäjäkokemussuunnittelu, on ajankohtainen ja trendikäs aihe ohjelmistokehityksessä, sillä elämme kokemusvetoisessa maailmassa [Silva ym., 2018]. Tämän tutkimuksen tavoitteena oli määrittää full stack -suunnittelijan rooli sekä selvittää, onko full stack -suunnittelijasta hyötyä suunnittelussa sekä kehityksessä näiden kahden menetelmän muodostamissa projekteissa työntekijän näkökulmasta. Vastausten etsimiseksi haastateltiin seitsemää eri henkilöä, jotka työskentelevät tai olivat työskennelleet projekteissa, joissa käyttäjäkeskeinen suunnittelu oli integroitu ketterään menetelmään. Haastattelutuloksia verrattiin kirjallisuuskatsauksessa löydettyihin vastauksiin ja näin voitiin muodostaa lopulliset tutkimustulokset.

Haastatteluiden tulokset olivat pääosin yhdenmukaisia kirjallisuuskatsauksessa esitettyjen tutkimustulosten kanssa, vaikka joitakin poikkeamia havaittiin. Keskeinen tulos on, että full stack -suunnittelijan rooli koetaan hyödylliseksi vain, jos projekti tai tiimi on tarpeeksi pieni. Näin ollen yhden henkilön on pakko osata tehdä useampaa asiaa samanaikaisesti. Kuitenkin jos projektin koko kasvaa, full stack -suunnittelijan tehtävät alkavat laajenemaan liikaa, jolloin alkaa ilmetä muun muassa ajankäytöllisiä haasteita. Kirjallisuuskatsauksissa nousi esille, että suunnittelijoiden sekä kehittäjien välinen kommunikaatio on huono ja heillä voi olla usein käynnissä valtataistelu. Haastatteluista kuitenkin muodostui käsitys, ettei näin kuitenkaan ole. Pääasiassa kehittäjät ja suunnittelijat tulevat toimeen, sillä käyttäjäkokemussuunnittelun integroiminen on tehty kunnolla sekä tuoteomistaja ja palvelumuotoilu ovat auttaneet suunnittelijoita työssään.

Full stack -suunnittelijaa ei ole tutkittu vielä tarpeeksi hyvin, eikä roolista ei ole tehty virallista määritelmää. Rooliin liittyvää taustaa kuitenkin on tutkittu paljon, kuten käyttäjäkeskeistäsuunnittelua, siinä olevia rooleja ja ketteriä menetelmiä, joiden avulla voidaan muodostaa käsitys siitä, mitä full stack -suunnittelijalla voidaan tarkoittaa. Tässä tutkielmassa tutkittiin full stack -suunnittelijan roolin tarpeellisuutta ketterässä ohjelmistokehityksessä työntekijän näkökulmasta. Jos aihetta tutkittaisiin esimerkiksi asiakkaan, yrityksen johdon tai loppukäyttäjän näkökulmasta, lopputulema voisi olla erilainen.

7 Lähdeluettelo

1stWebDesigner. (2017). The Definition of a Full Stack Designer. Viitattu 8. huhtikuuta 2024 osoitteesta *https://1stwebdesigner.com/full-stack-designer/*

Abid, M., & Bazaz, S. (2023). The rise of the Full Stack Designer. Viitattu 8. huhtikuuta 2024 osoitteesta *https://www.linkedin.com/pulse/rise-full-stack-designer-grayhatpk-yegif/*

Argumanis, D., Moquillaza, A., & Paz, F. (2020). Challenges in integrating SCRUM and the user-centered design framework: a systematic review. In Human-Computer Interaction: 6th Iberoamerican Workshop, HCI-Collab 2020, Arequipa, Peru, September 16–18, 2020, Proceedings 6 (pp. 52–62). Springer International Publishing.

Atrash, D. (2022). Full-stack User Experience Design: what it is and why you need it. Viitattu 8. huhtikuuta 2024 osoitteesta *https://bootcamp.uxdesign.cc/full-stack-user-experience-design-what-it-is-and-why-you-need-it-68c1a11aba7b*

Beck, K., Beedle, M., van Bennekum, A., et al. (2001). Agile manifesto. Viitattu 15. maaliskuuta 2024 osoitteesta *http://agilemanifesto.org*

Botha, C. (2018). The Unicorn Designer - Full Stack Designer. Viitattu 8. huhtikuuta 2024 osoitteesta *https://www.linkedin.com/pulse/unicorn-designer-full-stack-cindy-serfontein/*

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

Canziba, E. (2018). Hands-on UX design for developers: Design, prototype, and implement compelling user experiences from scratch (1st ed.). Packt.

Chung, K. (2021). What does it mean to be a “full stack” designer? Is it worth the effort? Viitattu 8. huhtikuuta 2024 osoitteesta *https://medium.com/ux-school/what-does-it-mean-to-be-a-full-stack-designer-is-it-worth-the-effort-575b07e3a4ea*

Cockton, G., Lárusdóttir, M., Gregory, P., & Cajander, Å. (2016). *Integrating user-centred design in agile development* (pp. 1–46). Springer International Publishing.

Curcio, K., Santana, R., Reinehr, S., & Malucelli, A. (2019). Usability in agile software development: A tertiary study. Computer Standards & Interfaces, 64, 61–77. *https://doi.org/10.1016/j.csi.2018.12.003*

Da Silva, T. S., Silveira, M. S., Maurer, F., & Silveira, F. F. (2018). The evolution of agile UXD. Information and Software Technology, 102, 1–5. *https://doi.org/10.1016/j.infsof.2018.04.008*

Dingsør, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. Journal of Systems and Software, 85(6), 1213–1221. *https://doi.org/10.1016/j.jss.2012.02.033*

Ferreira, J., Sharp, H., & Robinson, H. (2010). Values and assumptions shaping agile development and user experience design in practice. In Agile Processes in Software Engineering and Extreme Programming: 11th International Conference, XP 2010, Trondheim, Norway, June 1–4, 2010. Proceedings 11 (pp. 178–183). Springer Berlin Heidelberg.

Hirsjärvi, S., Remes, P., & Sajavaara, P. (2007). Tutki ja kirjoita (13. painos). Helsinki: Tammi.

Hooda, S. (Toim.). (2023). Agile Software Development: Trends, Challenges and Applications. Hoboken, New Jersey: John Wiley & Sons.

Hron, M., & Obwegeser, N. (2022). Why and how is Scrum being adapted in practice: A systematic review. *Journal of Systems and Software*, *183*, 111110.

Hussain, Z., Slany, W., & Holzinger, A. (2009). Current state of agile user-centered design: A survey. In *HCI and Usability for e-Inclusion: 5th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society, USAB 2009, Linz, Austria, November 9-10, 2009 Proceedings 5* (pp. 416-427). Springer Berlin Heidelberg.

Iiu, T. (2017). What is a Full Stack Designer? Will You Be One? Viitattu 8. huhtikuuta 2024 osoitteesta *https://medium.muz.li/what-is-a-full-stack-designer-in-2017-will-you-be-one-7933a7145fb7*

Knight, W., Knight, W., & Corrigan, L. (2019). UX for Developers. Northampton: Apress.

Kulchinskiy, A. (2021). What the hell are Full-stack Designers? Viitattu 8. huhtikuuta 2024 osoitteesta *https://www.linkedin.com/pulse/what-hell-full-stack-designers-arcadiy-kulchinskiy/*

Kuusinen, K. (2015). Task allocation between UX specialists and developers in agile software development projects. In Human-Computer Interaction–INTERACT 2015: 15th IFIP TC 13 International Conference, Bamberg, Germany, September 14–18, 2015, Proceedings, Part III 15 (pp. 27–44). Springer International Publishing.

Kuusinen, K., Mikkonen, T., & Pakarinen, S. (2012). Agile user experience development in a large software organization: Good expertise but limited impact. In *Human-Centered Software Engineering: 4th International Conference, HCSE 2012, Toulouse, France, October 29–31, 2012. Proceedings 4* (pp. 94–111). Springer Berlin Heidelberg.

Larocca, M. J. (2023). How to Become a Full Stack Designer in 2023. Viitattu 8. huhtikuuta 2024 osoitteesta *https://selftaughttxg.com/2023/01-23/how-to-become-a-full-stack-designer-in-2023/*

Nadikattu, S. R. (2016). Integrating User Experience (UX) Development with Agile Software Development Practices.: A Multiple Case Study Involving Organizations DevelopingInteractive Healthcare Technology (IHT) Applications.

Ng, M. (2024). The rise of the ‘full-stack’ designer. Viitattu 8. huhtikuuta 2024 osoitteesta *https://uxdesign.cc/the-rise-of-the-full-stack-designer-042adaeace39*

Persson, J. S., Bruun, A., Lárusdóttir, M. K., & Nielsen, P. A. (2022). Agile software development and UX design: A case study of integration by mutual adjustment. Information and Software Technology, 152, 107059.

Petersen, K., Wohlin, C., & Baca, D. (2009). The waterfall model in large-scale development. In *Product-Focused Software Process Improvement: 10th International Conference, PROFES 2009, Oulu, Finland, June 15–17, 2009. Proceedings 10* (pp. 386–400). Springer Berlin Heidelberg.

Saaranen-Kauppinen, A., & Puusniekka, A. (2006). KvaliMOTV - Menetelmäopetuksen tietovaranto. Tampere: Yhteiskuntatieteellinen tietoarkisto. Viitattu 29. huhtikuuta 2024 osoitteesta *http://www.fsd.uta.fi/menetelmaopetus/*

Salah, D., Paige, R. F., & Cairns, P. (2014). A systematic literature review for agile development processes and user centred design integration. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering* (pp. 1–10).

Sangiorgi, D. & Prendiville, A. (2017). Designing for Service: Key Issues and New Directions. Bloomsbury.

Schmitz, K., Mahapatra, R., & Nerur, S. (2019). User Engagement in the Era of Hybrid Agile Methodology. *IEEE Software*, *36*(4), 32–40. *https://doi.org/10.1109/MS.2018.290100623*

Schwaber, K., & Sutherland, J. (2020). The 2020 Scrum Guide. *https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Finnish.pdf*

da Silva, T. S., Martin, A., Maurer, F. & Silveira, M. (2011). User-centered design and agile methods: a systematic review, Anonymous, in: Proceedings of the Agile Conference, IEEE, pp. 77–86.

da Silva, T. S., Silveira, M.S., de O. Melo, C., Parzianello, L.C. (2013). Understanding the UX Designer's Role within Agile Teams, in: Marcus, A. (ed.), Design, User Experience, and Usability: Design Philosophy, Methods, and Tools, Berlin, Heidelberg, LNCS 8012 - Part I, Springer Berlin Heidelberg, pp. 599–609.

SimplifiedUX. (2023). Full-Stack Designers: What You Need To Know. Viitattu 8. huhtikuuta 2024 osoitteesta *https://simplifiedux.com/full-stack-designers-what-you-need-to-know/*

Solita. (2024). Impact that lasts. Viitattu 7. kesäkuuta 2024 osoitteesta *https://www.solita.fi/*

Sommerville, I. (2010). Software Engineering, 9"'ed. *England: Education Limited*.
Stickdorn, M., Hormess, M. E., Lawrence, A., & Schneider, J. (Eds.). 2018. This is Service Design Doing: Applying Service Design Thinking in the Real World. O'Reilly Media, Inc.

Stickdorn, M., Hormess, M. E., Lawrence, A., & Schneider, J. (Toim.). (2018). This is Service Design Doing: Applying Service Design Thinking in the Real World. O'Reilly Media, Inc.

Suomen Standardisoimisliitto SFS ry. (2021). *Ihmisen ja järjestelmän vuorovaikutuksen ergonomia. Osa 210: Vuorovaikutteisten järjestelmien käyttäjäkeskeinen suunnittelu.* SFS-EN ISO 9241-210:2019, 5–23.

Sutherland, J. (2015). The Scrum Papers: Nuts, Bolts, and Origins of an Agile Process. Viitattu 8. huhtikuuta 2024 osoitteesta *https://www.researchgate.net/publication/242437392_The_Scrum_Papers_Nuts_Bolts_and_Origins_of_an_Agile_Process*

Sy, D. (2007). Adapting usability investigations for agile user-centered design. Journal of Usability Studies, 2, 112–132.

Teka, D., Dittrich, Y., & Kifle, M. (2018). Adapting Lightweight User-Centered Design with the Scrum-Based Development Process. *2018 IEEE/ACM Symposium on Software Engineering in Africa (SEiA)*, 35–42. *https://doi.org/10.1145/3195528.3195530*

Thesing, T., Feldmann, C., & Burchardt, M. (2021). Agile versus waterfall project management: decision model for selecting the appropriate approach to a project. *Procedia Computer Science*, *181*, 746–756.

Tiainen, T. (2014). Haastattelu tietojenkäsittelytieteen tutkimuksessa. Informaatiotieteiden yksikön raportteja 25/2014, Tampereen yliopisto.

VersionOne. (2020). 14th Annual State of Agile Report.

Viitanen, M. (2018). *Full-stack -suunnittelijan roolin hyödyt ja haasteet ketterässä ohjelmistokehityksessä*. Diplomityö. Tampereen teknillinen yliopisto.

Liite 1: Haastattelukutsu

Otsikko: Tutkimus full stack -suunnittelijan roolin hyödyistä ja haasteista ketterässä ohjelmistokehityksessä – Määritelmä sekä esitiedot haastatteluun

Hei,

Nimeni on Topi Kalermo ja opiskelen viimeistä vuotta Tampereen yliopistossa tietojenkäsittelytiedettä. Pyydän sinua haastatteluun, joka on osana pro gradu -tutkielmaani. Osallistumalla tähän haastatteluun autat minua tutkimaan full stack -suunnittelijan roolin merkitystä ketterässä ohjelmistokehityksessä. Haastatteluun voi osallistua, vaikka ei olisi roolista aikaisemmin kuullut.

Vapaita aikoja haastatteluun on tällä viikolla 8.5. keskiviikkona ja seuraavalla viikolla maanantaista 13.5. eteenpäin. Haastatteluun voi osallistua klo 11–20 välisenä aikana. Kerro, mikä aika sopii sinulle, niin lähetän linkin haastatteluun (Microsoft Teams) jos päällekkäisyyksiä ei ole.

Haastattelun tallennussuostumuslomake, esitietolomake sekä linkki varsinaiseen haastatteluun lähetetään paria päivää ennen sovittua haastatteluaikaa.

Ystävällisin terveisin,
Topi Kalermo

Liite 2: Esitietolomake

Tutkimus full stack -suunnittelijan hyödyistä ja haasteista - Haastattelun esitietolomake

Ennen varsinaista haastattelua, täytäthän esitietolomakkeen. Tiedot pysyvät anonyymina tämän tutkimuksen loppuun asti, jonka jälkeen ne poistetaan.
Tähän lomakkeeseen vastaaminen vie arviolta aikaa noin minuutin.

1. Nimikirjaimesi, esimerkiksi N. N. (käytetään vain vastausten erotteluun). *
2. Syntymävuotesi *
3. Korkein suorittamasi tutkinto *
    - Peruskoulu
    - Ammattikoulu tai jokin muu ammatillinen koulutus
    - Lukio
    - Alempi korkeakoulututkinto (Kandidaatti, AMK)
    - Ylempi korkeakoulututkinto (Maisteri, YAMK)
    - Tohtori tai lisensiaatti
    - Muu
4. Kuinka suuressa yrityksessä työskentelet tällä hetkellä? *
    - Alle 10 työntekijää
    - 10–49 työntekijää
    - 50–249 työntekijää
    - 250 työntekijää tai enemmän
    - Opiskelija
5. Kuinka monta henkilöä tiimissäsi työskentelee itsesi mukaan lukien?
   Jos olet mukana useassa eri projektissa, erota luvut pilkuilla. Esim. 5,4. Jos et ole mukana projektissa, vastaa 0.

6. Kuinka monen vuoden kokemus sinulla on seuraavista?

- Ketterät menetelmät (esim. Scrum, Kanban)
    - Ei kokemusta
    - Alle vuosi
    - 1–3 vuotta
    - 4–7 vuotta
    - Yli 7 vuotta
- Käyttäjäkokemussuunnittelu (UX)
    - Ei kokemusta
    - Alle vuosi
    - 1–3 vuotta

    - 4–7 vuotta
    - Yli 7 vuotta
- Front-end kehitys
    - Ei kokemusta
    - Alle vuosi
    - 1–3 vuotta
    - 4–7 vuotta
    - Yli 7 vuotta
- Back-end kehitys
    - Ei kokemusta
    - Alle vuosi
    - 1–3 vuotta
    - 4–7 vuotta
    - Yli 7 vuotta

7. Tämänhetkinen roolisi

Voit valita usean vaihtoehdon.

- UI/UX-suunnittelija tai vastaava
- Ohjelmistokehittäjä tai vastaava
- Opiskelija
- Muu

Liite 3: Suostumuslomake haastatteluun

Otsikko: Suostumus haastatteluun ja lupa tallentaa se

Huomioithan, että mahdolliset taustalla olevat tapahtumat ja sieltä kuuluvat äänet voivat myös tallentua. Vain minä tulen katsomaan tallenteet analyysin tekemistä varten.

Kaikki materiaali, josta sinut voi tunnistaa, tallennetaan salasanalla suojaten. Nauhoitukset tuhotaan tutkielman jälkeen. Lisäksi tulokset analysoidaan niin, ettei sinua voi niistä tunnistaa. Audio- tai videotallenteita sekä henkilötietojasi ei luovuteta eteenpäin.

Haastatteluun on hyvä varata aikaa noin 30 minuuttia. Voit halutessasi lopettaa haastattelun milloin tahansa.

Vastaan mielelläni, jos sinulla on jotain kysyttävää.

1. Annan luvan tallentaa

- ääntä (puheeni ja mahdolliset taustaäänet)
- videota (kasvoni ja mahdolliset taustalla näkyvät asiat)

2. Olen saanut tietoa haastattelusta ja mahdollisuuden kysyä siihen liittyvistä asioista. Osallistun haastatteluun vapaaehtoisesti.

- Anna suostumuksesi yllä kuvattuihin ehtoihin kirjoittamalla nimesi (etunimi sukunimi)

Liite 4: Haastattelurunko

Otsikko: Tutkimus full stack -suunnittelijan roolin hyödyistä ja haasteista ketterässä ohjelmistokehityksessä – Työntekijän näkökulmasta

ALOITUS

Hei, nimeni on Topi Kalermo ja kiitos, kun pääsit osallistumaan haastatteluun. Tätä haastattelua käytetään Tampereen yliopiston tietojenkäsittelyn pro gradu -tutkielmaan ja haastattelun tarkoituksena on selvittää mitä hyötyjä ja haasteita full stack -suunnittelijan roolista on ketterässä ohjelmisto kehityksessä. Pyydänkin sinua tässä vaiheessa laittamaan puhelimen äänettömälle, jos se sinulla on vieressä.

Tämän haastattelun arvioitu kesto on noin 30 minuuttia, voimme kuitenkin venyttää 45 minuuttiin jos juttu luistaa ja sinulla on siihen aikaa. Tallennan tämän haastattelun vain itselleni, jotta voin kirjoittaa vastaukset ylös ja analysoida niitä myöhemmässä vaiheessa tätä tutkielmaa. Tallenne poistetaan sen jälkeen, kun analysointi on valmis, kuitenkin viimeistään kesäkuun loppuun mennessä. Varmistin suostumuksesi haastatteluun tallennussuostumuslomakkeella, mutta varmistan sen vielä kohta uudestaan.

HAASTATTELUN ESITTELY JA TARKOITUS

Kerron vielä lyhyesti, että mitä tämä haastattelu pitää sisällään. Eli olen lähtenyt tutkimaan full stack -suunnittelijan roolia ja sen merkitystä ketterässä ohjelmistokehityksessä. Tämän haastattelun tarkoituksena on avata mahdollisia omia kokemuksia eri roolien vaikutuksista ketterästä ohjelmistokehityksestä sekä mietteitä full stack -suunnittelijan roolista työntekijän tai opiskelijan näkökulmasta, sekä selvittää tämän pohjalta roolin hyödyt sekä haitat. Roolin määrityksen tulen kertomaan sinulle haastattelun aikana.

Voit lopettaa haastattelun milloin tahansa ihan mistä tahansa syystä, eikä sinun tarvitse selittää syytä, jos et halua. Samoin, jos joku yksittäinen kysymys tuntuu liian vaikealta etkä halua vastata siihen, voimme siirtyä seuraavaan kysymykseen.

Onko sinulla jotain kysyttävää vielä ennen haastattelua?

Olisin laittamassa nyt tallennusta päälle, sopiiko se sinulle?

HAASTATTELURUNKO

Nykytilanne

1. Miten projekteissa, joissa olet mukana on hoidettu käyttöliittymien suunnittelu ja toteutus?

a. Millä tavalla projektinne roolit on jaoteltu?

b. Miten yhteistyö toimii eri roolien välillä?

c. Toivoisitko enemmän yhteistyötä vai työnjakoa roolien välillä?

2. Minkälaista suunnittelutyötä olet tehnyt viimeaikaisissa projekteissa? (Jos kehittäjä -> Oletko kehittäjänä ollut mukana antamassa näkökulmia käyttöliittymän suunnitteluun?)

a. Mitä se on käytännössä pitänyt sisällään?

3. Oletko työskennellyt tiimissä, jossa sinä tai jokin muu henkilö olisi osallistunut käyttöliittymän suunnitteluun sekä toteutukseen?

a. Jos on, niin mistä tämä henkilö vastasi?

b. Koetko, että tästä henkilöstä oli hyötyä tai haittaa tiimille projektissa?

Haastateltavan toiveet

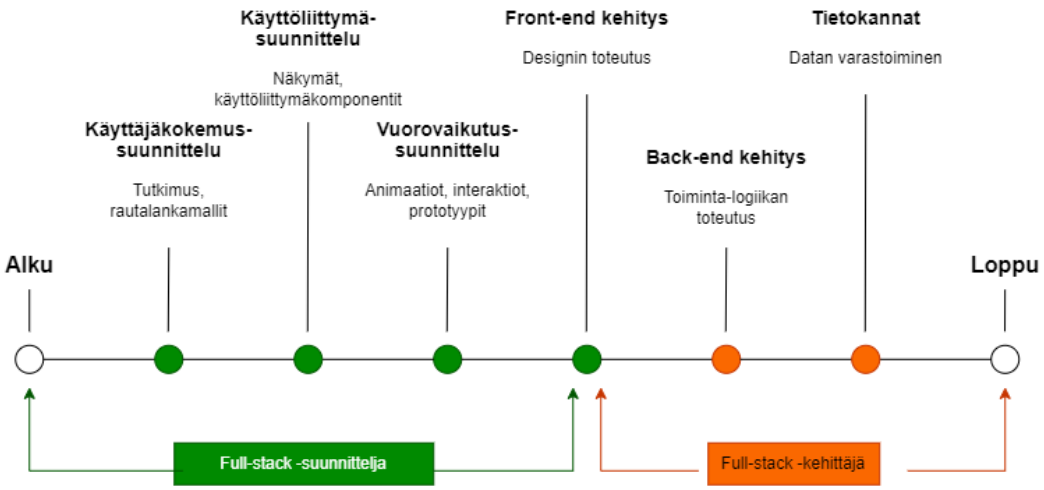
4. Jos saisit itse päättää miten projektit tulisi tehdä, niin miten tekisit ne?

a. Mitä rooleja?

b. Miten laajoja tehtäväalueita rooleilla olisi hyvä olla?

5. Millaisissa tilanteissa olisi hyvä olla erilliset henkilöt käyttöliittymien suunnitteluun ja toteutukseen?

Full stack -suunnittelija – KUVA CHATTIIN - Full stack -suunnittelija on henkilö, joka hallitsee käytettävyys-, käyttöliittymä- ja vuorovaikutussuunnittelun sekä front end -kehittämisen. Eli käytännössä full stack -suunnittelija osaa tarvittaessa suunnitella sekä toteuttaa itsenäisesti koko tuotesuunnitteluprosessin alusta loppuun aina käyttäjätutkimuksesta käyttöliittymän toteutukseen.



6. Olitko kuullut termistä aikaisemmin?

a. Jos olit kuullut termistä full stack -suunnittelija aikaisemmin, niin muuttiko se käsitystä siitä mitä full stack -suunnittelijalla tarkoitetaan?

i. Jos muutti, niin miten?

7. Kuvitellaan tilanne, että tiimissäsi on henkilö, joka osaa suunnitella käyttäjäkokemuksen, käyttöliittymän sekä vuorovaikutukset projektiinne ja tämän lisäksi vielä koodata käyttöliittymän vastaamaan suunnitelmia. Kokisitko, että tällaisesta henkilöstä olisi hyötyä juuri sinun tiimissäsi?

a. Miksi?

Haastattelu on ohi, kiitokset! Jos haluat lukea graduni kun se on valmis, niin voin lähettää sen samaan sähköpostiin johon lähetin haastattelukutsun.