Estimoinnin hyödyllisyys ohjelmistoprojekteissa

Toni Judin

Maisterintutkielma
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen osasto

Helsinki, 15. huhtikuuta 2020

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen osasto

Tekijä — Författare — Author
Toni Judin

Työn nimi — Arbetets titel — Title

Estimoinnin hyödyllisyys ohjelmistoprojekteissa

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Maisterintutkielma	15. huhtikuuta 2020	48 sivua + 1 liitesivu

Tiivistelmä — Referat — Abstract

Estimointia eli työmääräarviota pidetään yleisesti olennaisena osana onnistunutta ketterää ohjelmistoprojektia. Agile-yhteisössä ja Helsingin yliopistossa tämän opinnäytetyön aikoihin suhteellista estimointia eli ominaisuuksien työmääräarvioiden vertailua toisiinsa pidettiin ”parhaana” estimointikäytänteenä. Internetissä on alettu viime aikoina keskustella suhteellisen estimointitavan ja yleisesti estimoinnin hyödyllisyydestä ohjelmistoprojekteissa avainsanalla ”NoEstimates”. NoEstimates-liike keskustelee siitä, onko estimointi aina välttämätöntä ja miten estimointi tulisi toteuttaa.

Tämän opinnäytetyön tarkoituksena on löytää vastauksia kysymyksiin: onko estimointi aina välttämätöntä ohjelmistoprojekteissa ja onko muita varteenotettavia estimointimenetelmiä kuin suhteellinen ja absoluuttinen estimointi? Kirjallisuudesta ja NoEstimates-liikkeen blogeista ja keskusteluista pyrittiin löytämään estimointiin liittyviä kokemuksia, teorioita ja ideoita, joista pyrittiin saamaan lisätietoa haastattelututkimuksen avulla. Tutkimuksena tehtiin teemahaastattelu viidelle ohjelmistoalan ammattilaiselle. Tutkimuksella pyrittiin mm. kartoittamaan yrityksissä käytettäviä estimointimenetelmiä ja keräämään estimointikokemuksia ohjelmistoalan ammattilaisilta.

Haastattelututkimuksen johtopäätökseksi saatiin, että estimointi on lähes aina oleellinen osa onnistunutta ohjelmistoprojektia. Suhteellista estimointimenetelmää käytetään harvoin ohjelmistoyrityksissä, vaikka menetelmää pidetään kirjallisuudessa tämän hetken ”parhaana” estimointikäytänteenä. Tutkimuksen perusteella yleisin käytössä oleva estimointitapa yrityksissä on absoluuttinen estimointi. Haastattelututkimuksessa keskustelua käytiin myös NoEstimates-liikkeen keskusteluissa esiintyneestä mutta vähemmän tunnetusta estimointimenetelmästä, jota tässä opinnäytetyössä kutsustaan ominaisuuksien lukumäärällä estimoimiseksi. Suhteellisen ja absoluuttisen estimointimenetelmän lisäksi, ominaisuuksien lukumäärällä estimointi vaikuttaa lupaavalta estimointimenetelmältä.

ACM Computing Classification System (CCS):
Software and its engineering → Software creation and management → Software development process management → Risk management
Software and its engineering → Software creation and management → Software development process management → Software development methods → Agile software development

Avainsanat — Nyckelord — Keywords
Agile, estimointi, suhteellinen estimointi, käyttäjätarinapisteytys (Story Point), #NoEstimates

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

1 Johdanto

Estimointia ja suunnittelua (planning) pidetään olennaisena osana ohjelmistoprojektin onnistumista (Cohn, 2006, s. 3). Estimaatit projektin kestosta ja kustannuksista määräävät monesti esimerkiksi ohjelmistoprojektin aloittamisen kannattavuuden. Iteratiivisissa ohjelmistoprojekteissa estimoinnilla ei kuitenkaan pelkästään määritellä projektin kokonaiskestoa ja kustannuksia vaan pyritään löytämään optimaalinen ratkaisu, mitä ominaisuuksia voidaan toteuttaa iteraatioiden aikana. Hyvän suunnittelun ja estimoinnin ajatellaan vähentävän ohjelmistoprojektin riskejä ja epävarmuutta, ohjaavan päätöksentekoa, rakentavan luottamusta ja välittävän tietoa.

Estimointia pidetään kuitenkin erittäin haastavana ja epätarkkana erityisesti projektin alussa (Cohn, 2006, s. 4). Epäonnistunutta estimointia pidetään yhtenä merkittävänä syynä ohjelmistoprojektien epäonnistumiselle (Verner et al., 2008).

Ohjelmistoprojekteissa nykyisin yleisesti käyttössä olevat estimointimenetelmät voidaa jakaa kahteen kategoriaan: suhteellisiin (esim. käyttäjätarinapisteytys (Story Point)) ja absoluuttisiin estimointimenetelmiin (esim. henkilötyöpäivä). Suhteellisella estimoinnilla tarkoitetaan, että ominaisuuksia ei estimoida niiden toteutukseen kuluvan ajan perusteella (absoluuttinen estimointi), vaan ominaisuuden työläyden perusteella suhteessa muihin ominaisuuksiin. Suhteellista estimointimenetelmää pidettiin opinnäytetyön kirjoituksen aikaan ns. ”parhaana” estimointikäytänteenä Helsingin yliopiston tietojenkäsittelytieteen osastolla ohjelmistotuotantokurssilla ja monissa ketterään ohjelmistokehitykseen liittyvissä internetkeskusteluissa.

Viime vuosina internetissä on alettu keskustella esimerkiksi käyttäjätarinapisteille vaihtoehtoisista estimointitavoista ohjelmistoprojekteissa. Internetiin on syntynyt niin sanottu NoEstimates-liike, jonka kannattajat kritisoivat blogikirjoituksissaan nykyisiä vallitsevia estimointitapoja avainsanalla ”NoEstimates”. Eräs NoEstimates-liikkeen pioneereista on Ron Jeffries, joka julkaisi vuonna 2013 artikkelin otsikolla ”The NoEstimates Movement” (Jeffries, 2013). Estimointia pidetään vaikeana, jopa mahdottomana. Estimointi on vaikeaa, koska projektien vaatimukset ovat lähes aina epämääräisiä. Vaikka vaatimukset olisivatkin selkeitä, on lähes mahdotonta tietää, kuinka kauan

jonkin asian tekemisessä kestää, koska kukaan ei ole tehnyt sitä ennen.

Vaikka liikkeen nimi onkin provosoivan oloinen, NoEstimates-liikkeen tarkoituksena ei kuitenkaan ole tuomita kaikkea estimointia (Jeffries, 2013). Eräs keskusteluissa esitetty mielipide estimoinnista on, että kaikki olisi paremmin, jos estimointia pystyttäisiin kokonaan välttämään. Liikkeessä kuitenkin tiedostetaan, ettei estimoinnista kokonaan luopuminen ole realistista. Tarkoitus on saada ohjelmistokehittäjät ajattelemaan, milloin estimointi on hyödyllistä ja millä tavalla estimointi tulisi toteuttaa. Eräänä tavoitteena onkin saada muutettua ohjelmistoprojektin johdon, ohjelmistokehittäjien ja asiakkaiden suhtautumista estimointiin.

NoEstimates-keskusteluissa estimointia ei pidetä ”kaikissa muodoissa” täysin mahdottomana (Jeffries, 2013). On mahdollista tuottaa riittävän tarkkoja estimaatteja ja estimaattien antamaa tietoa tulisi käyttää harkiten. Usein estimoinnin tuottamaa tietoa käytetään väärin – estimaatteja käytetään monesti esim. pakotteena saada ohjelmistokehittäjät työskentelemään nopeammin. Tällaisen pakottamisen kerrotaan lisäävän kehittäjien tyytymättömyyttä ja huonontavan tuotettavan ohjelmiston laatua. Liikkeen tarkoituksena onkin löytää parhaita mahdollisia tapoja välittää ohjelmistoprojektin johdolle ja asiakkaalle niiden tarvitsemaa tietoa, ja saada ne pohtimaan tiedon käyttötarkoitusta.

Tässä opinnäytetyössä pyritään selvittämään ohjelmistoprojekteissa käytettävän ns. suhteellisen (Cohn, 2006, s. 36) estimointimenetelmän ongelmia. Suhteellisella estimointimenetelmällä tarkoitetaan estimaatin muodostamista vertailemalla eri työvaiheita toisiinsa, ja käyttämällä mittarina ajan sijasta ominaisuuden toteuttamiseen tarvittavaa työmäärää (effort). Työmäärä ei suoraan kuvaa työvaiheen tekemiseen kuluvaa aikaa. Käyttäjätarinoita estimoidessa suhteellista estimointimenetelmää kutsutaan käyttäjätarinapisteytykseksi (Story Point).

Opinnäytetyössä keskitytään tarkastelemaan suhteellista estimointimenetelmää ohjelmistoprojekteissa, jotka käyttävät Scrum-projektinhallintakehystä (ScrumGuides.org, 2018), tai sen kaltaista ketterää prosessimallia. Opinnäytetyössä pyritään selvittämään millaisia ajatuksia ja parannusehdotuksia NoEstimates-liikkellä on ohjelmistoprojektien estimointiin liittyen. Tarkoituksena on selvittää millaisia estimointikäytänteitä NoEstimates-liikkeen kes-

kustelijat suosivat ja näiden perusteella arvoida, onko suhteellinen estimointitapa ”paras” käytänne ohjelmistoprojektin estimoinnille. Lisäksi pyritään selvittämään onko estimointi ylipäätään välttämätöntä ohjelmistoprojektin onnistumisen kannalta.

1.1 Tutkielman rakenne

Opinnäytetyön toisessa luvussa esitetään estimaatin määritelmä ja mitä hyötyjä estimoinnista on, sekä millaisia ongelmia estimointiin liittyy. Lisäksi selvitetään millaista estimaattia pidetään onnistuneena ja miten estimointi on onnistuneet aiemmin ohjelmistoprojekteissa. Kolmannessa luvussa selvitetään, miten estimointia käytetään Scrum-projektinhallintakehyksessä. Neljännessä luvussa esitellään NoEstimates-liike ja liikkeen keskusteluista löytyneitä ajatuksia estimoinnin toteuttamiseksi ja estimointiongelmien ratkaisemiseksi. Luvussa keskitytään tarkastelemaan liikkeen ideoita erityisesti Scrum-projektinhallintakehyksen kanssa käytettävien estimointimenetelmien näkökulmasta. Viidennessä luvussa esitellään viidelle ohjelmistoalan ammattilaiselle tehty haastattelututkimus ja kuudennessa luvussa tutkimuksen tulokset. Opinnäytetyön seitsemännessä luvussa esitellään tutkimuksen pohjalta tehtyjä pohdintoja. Kahdeksannessa luvussa kerrotaan tutkimuksen perusteella tehdyistä johtopäätöksistä.

2 Mikä on estimaatti?

Eräs määritelmä estimoinnille löytyy Merriam-Webster sanakirjasta. Sanakirja antaa estimoinnille määritelmän: ”to give or form a general idea about the value, size or cost of something” (Merriam-Webster, 2019). Määritelmän mukaan verbi ”estimointi” tarkoittaa idean muodostamista jonkin asian arvosta, koosta tai kustannuksesta. Substantiivi estimaatti tarkoittaa arviota jonkin asian arvosta koosta tai kustannuksesta.

Estimaatti kuvataan arviona tehtävään tarvittavista resursseista (Duarte, 2016, s. 25). Estimaatin kuvauksissa estimointi ei perustu faktoihin. Estimoinnin laatimisessa käytetään hyväksi tehtäväkuvausta, aikaisempaa kokemusta ja muuta saatavilla olevaa tietoa muodostamaan järkevältä kuulostava luku. Monesti tämä luku on jokin rahamäärä tai aikamääre. Ihmiset pitävät lukua

usein faktatietona, vaikka kyseessä on vain tapa ilmaista, että asiasta ei ole kovin tarkkaa tietoa. Mikäli tehtävään käytettävät resurssit ja aika olisivat ennalta tiedossa, kyseessä ei olisi estimaatti vaan faktojen esittäminen. Estimaatteja ajatellaan monesti sitoutumisena saada jokin asia valmiiksi estimoiduilla resursseilla (Moskalenko, 2017).

Ohjelmistoprojekteissa estimoinnin eräs tarkoitus on määritellä, että onko projektia mahdollista hallita niin, että projektin tavoitteet on mahdollista saavuttaa (McConnell, 2006). Projektin hallitsemisella tarkoitetaan erilaisten ohjelmistoprojektin muuttujien muokkaamista. Muuttujia voivat olla esimerkiksi: toteutettavat ominaisuudet, aikataulu ja tiimin koko. Estimaattien ei tarvitse olla täysin tarkkoja ollakseen hyödyllisiä. Estimaattien tarkoitus on selvittää, kasvaako tavoitteiden ja käytettävissä olevien resurssien kuilu liian isoksi.

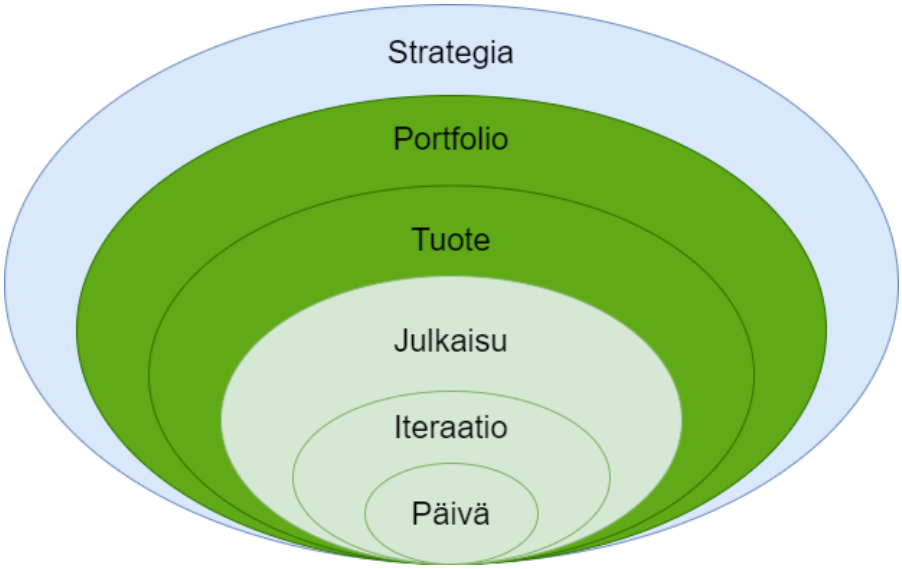
Joskus estimaattia pidetään ennusteen (forecast) synonyyminä (Duarte, 2016, s. 30). Estimaatti saatetaan määritellä ihmisyksilön tekemäksi arvioksi, kun taas ennusteella viitataan eksaktin mallin tuottamaan tulokseen. Ennustemallin syötteinä toimivat projektien toteutunut historiadata. Joskus ennusteita kutsutaan estimointimalleiksi ja estimaatteja asiantuntijaestimoinneiksi (Jørgensen, 2002). Ennusteita on pyritty hyödyntämään ohjelmistotuotantoprojekteissa ennustamaan esimerkiksi projektin valmistumisajankohtaa (Colucci, 2016). Ihmisten käyttäytyminen tuo kuitenkin erittäin suuren epävarmuustekijän ohjelmistoprojekteissa ja niiden ottaminen huomioon ennusteissa on erittäin haasteellista (Duarte, 2016, s. 31). Estimaatin ja ennusteen määritelmä on kuitenkin häilyvä. Tässä opinnäytetyössä puhutaan pääasiassa estimaatista ellei lähdemateriaali anna aihetta puhua ennusteesta.

2.1 Estimoinnin tasot

Estimoinnille voidaan määritellä eri tasoja riippuen siitä, mitä ohjelmistoyrityksen kerrossuunnittelun kerrosta ne koskevat. Ohjelmistoyrityksen kerrossuunnittelu koostuu kuudesta kerroksesta (kuva 1). Kerrossuunnittelun kolmea ulointa kerrosta koskevat estimaatit ovat korkeimman tason estimaatteja, kun taas kolmea sisintä kerrosta koskevat estimaatit ovat matalimman

tason estimaatteja. Tässä opinnäytetyössä käsitellään suurimmaksi osaksi kolmea matalinta estimoinnin tasoa: julkaisua, iteraatiota ja päivää.

Monesti yrityksissä korkeamman tason estimaatteja tuottavat eri rooleissa olevat ihmiset kuin matalan tason estimaatteja. Matalan tason estimaatteja tuottavat yleensä jonkin tietyn ohjelmistoprojektin ohjelmistokehittäjät. Korkeimman tason estimaatteja voivat tuottaa esimerkiksi yrityksen johtohenkilöstö tai heidän palkkaamat konsultit. Joissakin yrityksissä samat ihmiset voivat tuottaa korkean ja matalan tason estimaatteja.

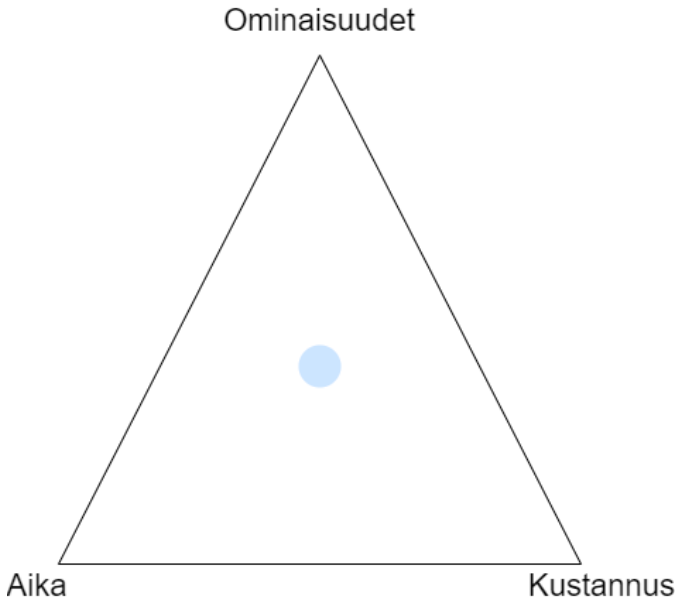


Kuva 1: Ohjelmistoyrityksen kerrossuunnittelu (Roach, 2011).

2.2 Miksi estimoidaan?

Estimointi on eräs tärkeimmistä elementeistä ohjelmistoprojektin onnistumisen kannalta. Estimoinnilla pyritään ennustamaan projektiin tarvittavia resursseja ja resurssitarpeen perusteella päättelemään kannattaako projekti toteuttaa. Epätarkka estimointi voi johtaa esimerkiksi laadun heikkenemiseen tai toteutettavien ominaisuuksien karsimiseen kokonaan (tästä lisää luvussa 2.6). Epätarkka estimointi voi johtaa myös projektin keskeytykseen mikäli esimerkiksi projektin aikana käy ilmi, että projektin aikaisemmassa vaiheessa tekemät estimaatit ylittivät projektin sietokyvyn.

Projektin vastuuhenkilöilä on vastuu projektin kolmesta elementistä: mitä (ominaisuudet, performance), milloin (aika, time) ja mitä maksaa (kustannus, cost) (Dalcher, 2014, s. 32). Projektin aikana projektin vastuuhenkilöt yrittävät pitää nämä kolme elementtiä tasapainossa. Estimaatit auttavat projektin vastuuhenkilöitä tekemään päätöksiä näiden elementtien tasapainottelussa. Projektin kolmen elementin tasapainoa kuvataan esimerkiksi elementtien valintakolmiolla (kuva 2).



Kuva 2: Ominaisuuksien, ajan ja kustannuksen valintakolmio. Pisteen liikuttaminen kolmion kärkeä kohti kuvaa elementtiin panostamista muiden elementtien kustannuksella (Dalcher, 2014).

Ihmisten stressitaso laskee, kun ihmiset saavat tietää, kuinka kauan jokin asia kestää, jotta ihmiset voivat suunnitella muita asioita estimoidun asian ympärille (Duarte, 2016, s. 46). Estimaatit tuottavat ihmisille turvallisuudentunnetta, koska ne tuovat luottamusta siihen, että projekti on hallinnassa, riippumatta estimaattien todenperäisyydestä.

Ohjelmistoprojekteissa estimaateilla pyritään voittamaan tarjouskilpailuja (Duarte, 2016, s. 48). Tarjouskilpailujen voittaminen on tärkeätä erityisesti isoille ohjelmistoprojekteilla, koska kyseessä on isot rahamäärät, statusarvo ja maine. Isoissa projekteissa estimointiin kiinnitetään erityisen paljon

huomioita, joten estimointiin käytetään erittäin paljon aikaa ja rahaa.

2.3 Millainen on hyvä estimaatti?

Kirjallisuudessa yleisesti käytetyn määritelmän mukaan estimaatti on käytännössä onnistunut silloin, kun 75 % tapauksista lopputulos on 25 % sisällä estimaatista (Conte, Dunsmore, & Shen, 1986). Määritelmän mukaan siis projektin estimointi on onnistunut, kun 75 % projektin työvaiheiden lopputuloksista on 25 % sisällä työvaiheiden estimaateista. Jokaisen työvaiheen lopputuloksen ei siis tarvitse osua estimaatin 25 % virhemarginaalin sisälle.

Edellä mainittu määritelmä on vain yksi näkemys hyvästä estimaatista ilman tieteellistä pohjaa, joten siihen tulee suhtautua varauksella. Määritelmää voi olla hankala soveltaa kaikkiin mahdollisiin projekteihin. Jokaisen projektin tulisikin itse määritellä, mikä on riittävä estimointitarkkuus.

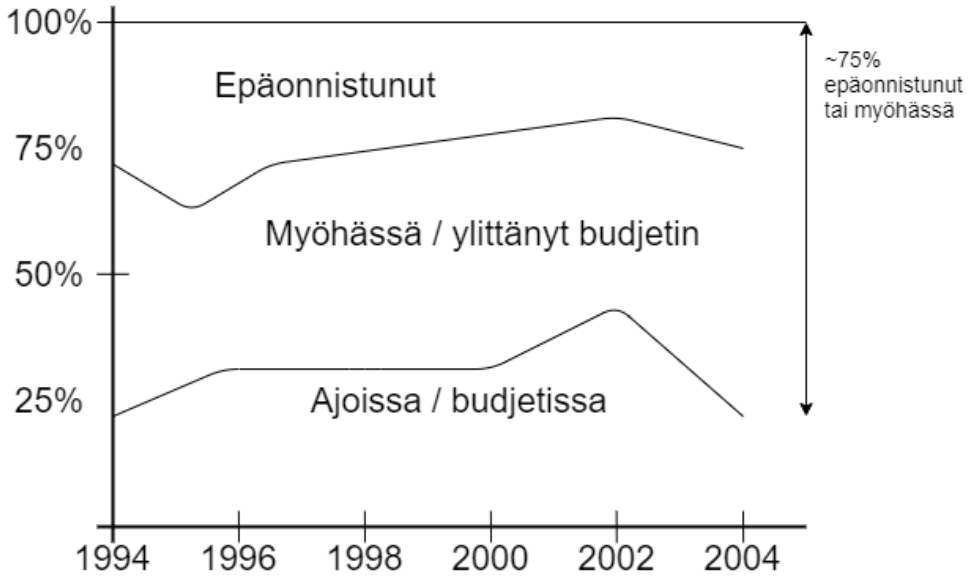
2.4 Miten estimointi toimii ohjelmistoprojekteissa?

Eräs merkittävimmistä ja laajimmista tutkimuksista kuvaamaan ohjelmistoprojektien tilaa on The Standish Groupin julkaisema CHAOS-katsaus (katsaukseen viitataan myös NoEstimates-keskustelussa) (The Standish Group, 1995). CHAOS-katsauksen tuloksia on kritisoitu epämääräsistä tutkimusmenetelmistä ja niukasta tutkimusmenetelmän kuvaamisesta, joten tutkimustuloksiin tulee suhtautua varauksella (Eveleens & Verhoef, 2010).

CHAOS-katsaus luokittelee ohjelmistoprojektit kolmeen eri tyyppiin (The Standish Group, 1995). Onnistunut projekti on projekti, joka on pysynyt aikataulussa ja budjetissa sekä kaikki suunnitellut ominaisuudet on saatu toteutettua. Haasteellinen projekti on valmistunut mutta projekti ei ole pysynyt estimoidussa aikataulussa, projekti on ylittänyt estimoidut kustannukset tai projektissa on jätetty toteuttamatta suunniteltuja ominaisuuksia. On kuitenkin epäselvää, ovatko toteuttamatta jätetyt ominaisuudet projektin onnistumisen kannalta kriittisiä ominaisuuksia. Ominaisuuksien toteuttamatta jättäminen ei itsessään tarkoita epäonnistunutta projektia, vaan voi olla projektille jopa eduksi (tästä lisää luvussa 2.6.1). Epäonnistunut projekti on jossakin projektin kehityksen vaiheessa keskeytetty projekti.

Vuosien 1994-2004 julkaistujen CHAOS-katsausten tulosten (kuva 3)

mukaan noin 25 % ohjelmistoprojekteista on epäonnistuneita projekteja ja noin 50 % projekteista on myöhästynyt tai ylittänyt budjetin (McConnell, 2006). Onnistuneita projekteja on noin 25 %, eli katsausten mukaan noin 75 % ohjelmistoprojekteista on epäonnistunut, myöhässä aikataulusta, ylittänyt budjetin tai projekteissa on jouduttu karsimaan toteutettavia ominaisuuksia (Duarte, 2016, s. 12).



Kuva 3: Ohjelmistoprojektien onnistumisprosentti CHAOS-katsauksen mukaan vuosilta 1994-2004. Vasco Duarten tekemä adaptaatio Steve McConnelin kirjassa esitetyistä tuloksista (Duarte, 2016).

Ohjelmistoprojektin onnistumismahdollisuuksia voidaan parantaa käyttämällä vesiputousmallin sijasta ketteriä projektinhallintamenetelmiä (Duarte, 2016, s. 12). Ketteriä menetelmiä käyttävistä projekteista 42 % on vuoden 2018 CHAOS-katsauksen (kuva 4) mukaisia onnistuneita projekteja (Johnson, 2018). Vesiputousmallia käyttävistä projekteista 11 % on onnistuneita projekteja. Vuoden 2018 katsauksen aineisto on kerätty yli 50 000 ohjelmistoprojektista.

Ohjelmistoprojektin laajuus vaikuttaa merkittävästi projektin onnistumistodennäköisyyteen. Uusi CHAOS-katsaus-malli erottelee ohjelmistoprojektit pieniin, keskisuuriin ja suuriin projekteihin (Johnson, 2018). Kuvasta näh-

dään, että pienistä ketteriä menetelmiä käyttävistä projekteista 59 % on onnistuneita projekteja (kuva 4). Isoista ketteristä projekteista 18 % on onnistuneita projekteja. Katsauksen mukaan projektin koko vaikuttaa siis olennaisesti projektin onnistumismahdollisuuksiin; pienillä projekteilla on paremmat mahdollisuudet onnistua kuin isoilla. Tätä tulosta tukee ulkopuolinen tutkimus, jossa tutkittiin projektin koon vaikutusta estimointitarkkuuteen (Usman et al., 2018). Tutkimuksen mukaan ohjelmistoprojektin koko vaikuttaa suoraan estimaattiin seuraavasti: mitä pienempi projekti, sitä helpommin projektissa yliestimoidaan eli estimaatit ovat isompia kuin toteutunut tulos. Vastaavasti taas mitä suurempi projekti, sitä helpommin se aliestimoidaan, eli estimaatti on pienempi kuin toteutunut tulos. Tutkimuksessa on ollut mukana 60 projektia, jotka on jaettu pieniin, keskisuuriin, suuriin ja erittäin suuriin projekteihin.

CHAOS-katsauksen taulukosta nähdään myös, että muita projektinhallintamenetelmiä käyttävistä pienistä projekteista 56 % on onnistuneita projekteja ja suurista projekteista 9 % (Johnson, 2018). Ketteriä projektinhallintamenetelmiä käyttävällä projektilla on siis muita menetelmiä käyttävää projektia paremmat mahdollisuudet onnistumiseen projektin koosta riippumatta.

Katsauksen perusteella ohjelmistoprojektien onnistumistodennäköisyys on parantunut ketterien projektinhallintamenetelmien ansiosta merkittävästi. Ketterät projektinhallintamenetelmät ovat tuoneet suhteelliset estimointimenetelmät (esim. käyttäjätarinapisteet) ohjelmistoprojekteihin. NoEstimates-keskustelussa pyritään kuitenkin parantamaan edelleen projektien onnistumistodennäköisyyksiä muuttamalla suhtautumista estimaatteihin ja vähentämään estimointiin käytettävää aikaa. Tästä lisää tutkielman myöhemmässä vaiheessa.

Katsauksen perusteella pienillä projekteilla on paremmat onnistumismahdollisuudet kuin isoilla projekteilla. Tämä voidaan tulkita myös siten, että pieneen projektiin tarvittavat resurssit osataan estimoida isoa projektia tarkemmin.

Projektien onnistumistodennäköisyys on noussut vuosien 1994-2004 25 % tasosta (kuva 3) vuoden 2013-2017 noin 40 % tasoon (kuva 4), kun käytössä on ketterät projektinhallintamenetelmät. Nousun syynä on ketterien projektinhallintamenetelmien yleistyminen ja mahdollisesti näiden mukana

Koko	Menetelmä	Onnistuneet	Haasteelliset	Epäonnistuneet
Kaikki projektit	Agile	42%	50%	8%
	Muut	26%	53%	21%
Isot projektit	Agile	18%	66%	16%
	Muut	9%	56%	35%
Keskikokoiset projektit	Agile	31%	59%	10%
	Muut	19%	61%	20%
Pienet projektit	Agile	59%	37%	4%
	Muut	56%	34%	10%

Kuva 4: Vuoden 2018 CHAOS-katsauksen jaottelu ohjelmistoprojektien onnistumisesta koon ja käytettävän projektinhallintamenetelmän mukaan (Johnson, 2018).

tuomat uudet estimointikäytänteet (kuten käyttäjätarinapisteet). Onnistumistodennäköisyydessä on kuitenkin vielä paljon parannettavaa, mihin entistä paremmat estimointimenetelmät saattavat tuoda parannusta.

2.5 Estimointiin liittyviä ongelmia

Estimointien epäonnistumiselle esitetään NoEstimates-keskustelussa neljä keskeistä ongelmaa: yllätyksellinen kustannus (Accidental Complication), muutokset projektissa, Hofstadterin laki ja Parkinsonin laki, (Duarte, 2016). Nämä ongelmat perustuvat lähinnä yksittäisten henkilöiden kokemuksiin eikä niitä ole tieteellisesti tutkittu kovin tarkasti. Näihin ongelmiin tuleekin suhtautua varauksella. Näiden lisäksi on joukko muita ongelmia, jotka vaikuttavat estimoinnin luotettavuuteen ja estimoinnin tarkkuuteen.

2.5.1 Yllätyksellinen kustannus

J.B. Rainsberger esitti vuonna 2013 konseptin yllätykselliselle kustannukselle ohjelmistoprojekteissa (Rainsberger, 2013). Yllätyksellisen kustannuksen teorian mukaan jokaisella ohjelmistoprojektin ominaisuudella on kaksi kustannuselementtiä: Välttämätön kustannus (Essential Complication) ja yllätyksellinen kustannus (Accidental Complication).

Välttämättömällä kustannuksella tarkoitetaan jonkin ominaisuuden toteuttamiseen käytettäviä välttämättömiä resursseja.

Yllätyksellisellä kustannuksella tarkoitetaan kaikkea muuta kuin itse ominaisuuden toteuttamiseen käytettävää työtä. Muihin kustannuksiin sisältyy esimerkiksi byrokraattiset työt uutta testiympäristöä varten tai olemassa olevien ominaisuuksien muokkaaminen, jotta uusi ominaisuus saadaan toimimaan vanhojen ominaisuuksien kanssa. Ominaisuuden kokonaiskustannus on siis välttämättömän kustannuksen ja yllätykellisen kustannuksen summa.

Yllätyksellisen kustannuksen ongelmat tulevat esille, kun ominaisuudet estimoidaan vain välttämättömän kustannuksen perusteella (Duarte, 2016). Kaksi ominaisuutta voivat olla välttämättömältä kustannukseltaan samankaltaisia, mutta yllätykselliseltä kustannukselta hyvin erilaisia. Mitkään estimointimenetelmät (kuten suhteellinen estimointi) eivät toimi luotettavasti, ellei yllätyksellisiä kustannuksia huomioida.

2.5.2 Muutokset projektissa

Yllätykselliset muutokset projektissa aiheuttavat estimoinnin epätarkkuutta (Popli & Chauhan, 2014). Syitä yllätykselliseen muutokseen projektissa voi olla esimerkiksi kommunikointiongelmat asiakkaan ja kehittäjien välillä. Asiakas voi olla epävarma projektin sisällöstä, mikä aiheuttaa jatkuvasti muutoksia projektiin projektin aikana.

Scrum-projektihallintakehyksen kanssa yleisesti käytettävä käyttäjätarinoiden pisteytys hankaloittaa muutoksiin sopeutumista (Duarte, 2012). Mikäli käyttäjätarinoita tulee jatkuvasti lisää projektin aikana, uudet käyttäjätarinat täytyy pisteyttää ennen kuin esimerkiksi projektin valmistumisaikataulu voidaan päivittää sisältämään uudet käyttäjätarinat. Käyttäjätarinoiden pisteytys kuluttaa kehitystiimin aikaa, joka voitaisiin hyödyntää

ominaisuuksien kehittämiseen.

2.5.3 Hofstadterin laki

Hofstadterin laki sanoo, että jonkin asian tekemiseen menee aina oletettua kauemmin, vaikka Hofstadterin laki otettaisiin huomioon (Hofstadter, 1979, s 160). Alun perin Douglas Hofstadter viittasi lailla shakkia pelaaviin tietokoneohjelmiin. Hofstadter tarkoitti laillia, että shakkia pelaavat tietokoneohjelmat ovat aina 10 vuotta parhaita ihmispelaajia jäljessä riippumatta siitä, kuinka kauan aikaa niiden kehittämiseen käytettiin. Ohjelmistokehityksessä lakiin viitataan, kun halutaan perustella estimoinnin hankaluutta (Duarte, 2016, s 15).

2.5.4 Parkinsonin laki

Parkinsonin laki on peräisin vuonna 1955 kirjoitetusta artikkelista, jossa alun perin pyritään esittämään valtion hallintobyrokratian kasvua ajan kuluessa (Parkinson, 1955). Ohjelmistokehityksessä on alettu viitata Parkinsonin lain artikkelista lainattuun osaan: ”Work expands so as to fill the time available for its completion”, eli työn kesto täyttää työlle annetun aikamääreen (Duarte, 2016, s 15). Tätä lainausta on alettu kutsua ohjelmistokehityksessä Parkinsonin laiksi. Parkinsonin lakia on käytetään perusteluna estimoinnin epävarmuudelle. Eräs tulkinta laillie on, että mikäli jokin projektin osavaihe myöhästyy estimoidusta aikataulusta, mikään muu projektin osavaihe ei valmistu estimointia aikasemmin, jolloin yhden osavaiheen myöhästymistä ei saada ikinä kiinni. Lopulta osavaiheiden myöhästymset kasaantuvat ja projektin kokonaisaika myöhästyy alkuperäisestä estimaatista. Mikäli myöhästymisiä koetetaan kompensoida pakottamalla muita työvaiheita valmistumaan nopeammin kuin alkuperäinen suunnitelma, estimaatit voivat osoittautua liian lyhyiksi (McConnell, 2006). Mikäli osavaiheet estimoidaan liian lyhyiksi, osavaiheiden toteutuksen huolellisuus kärsii, jolloin toteutuksessa syntyy enemmän virheitä. Virheitä joudutaan korjaamaan jälkeenpäin, mikä lisää työmäärää entisestään.

2.5.5 Muita ongelmia

Estimoinnissa monesti hyödynnetään kokemusta edellisistä projekteista (Popli & Chauhan, 2014; Keaveney & Conboy, 2006). Edellisten projektien tarkastelussa on kuitenkin huomioitava projekteissa sattuneet virheet, jotka ovat vaikuttaneet estimoinnin epätarkkuuteen. Mikäli edellisten projektien tarkastelu on jäänyt puutteelliseksi, projekteissa sattuneista virheistä ei ole opittu, joten virheet siirtyvät uuteen projektiin, jossa on käytetty vanhoja projekteja hyväksi estimoinnissa.

Poliittiset voimat, kuten projektin tai yrityksen johto, voivat painostaa kehitystiimiä pysymään määrätyssä aikataulussa tai kustannuksissa (Popli & Chauhan, 2014; Keaveney & Conboy, 2006). Tällainen painostus voi johtaa epärealistisiin estimaatteihin, jos estimaatteja ei tehdä rationaalisesti. Painostuksen alla tehdyillä estimaateilla tiimi yrittää miellyttää johtoporrasta tai asiakasta, eikä näin ollen projektia tarkastella objektiivisesti, mikä johtaa epätarkkaan estimointiin.

Monesti arvioita ei tehdä rationaalisesti. Tiimien käyttäytymisnormit vaikuttavat siihen, millaisia arvioita tehdään (Duarte, 2012). Esimerkiksi käyttäjätarinoita estimoitaessa yhden tiimiläisen sosiaalinen status voi vaikuttaa muiden tiimin jäsenten mielipiteisiin käyttäjätarinan koosta. Muut tiimiläiset luottavat liikaa korkeammalla statuksella olevan tiimiläisen arvoihin, jolloin he hyväksyvät korkeamman statuksen omaavan tiimiläisen arviot herkästi, eikä objektiivista keskustelua synny, jolloin estimaatissa voi jäädä näkökantoja huomioimatta ja estimointitarkkuus heikkenee.

Estimaatteja pidetään monesti sitoutumisena saada jokin asia valmiiksi estimoiduilla resursseilla (Moskalenko, 2017). Tästä syystä estimaatteja käytetään suunnitelman luomiseen. Kaikki myöhemmät muutokset estimaateissa muuttavat myös estimaattien pohjalta tehtyä suunnitelmaa.

Lyhyissä, muutaman iteraation, ohjelmistoprojekteissa estimointitarkkuus on projektin alussa heikko, mutta tarkentuu projektin aikana nopeasti (Mahnič, 2011). Erään tutkimuksen mukaan muutaman kymmenen iteraation projekteissa estimointitarkkuus ei kuitenkaan parane projektin edetessä (Cao, 2008). Iteraatioiden velositeetin estimointitarkkuus saattaa vaihdella erittäin paljon projektin aikana. Ketterien projektinhallintamenetelmien

käyttäjätarinoita hyödyntävät estimointimenetelmät ovat tarkempia kuin perinteiset vesiputousmallin menetelmät, mutta projektit saattavat sisältää enemmän epävarmuutta kuin ketterien menetelmien kirjallisuus antaa ymmärtää. Suuren estimoinnin epävarmuuden arvioidaan johtuvan esimerkiksi resurssien puutteesta, muutoksista projektissa, puutteellisesta refaktoroinnissa, yksikkötestien puutteesta, kommunikaatio-ongelmista ja häiriöistä. Ketterien projektinhallintamenetelmien jatkuva mukautuminen ja nopea palaute ei poista estimoinnin epävarmuutta.

Ohjelmointivirheiden korjausten estimointi on haastavampaa kuin uusien ominaisuuksien valmistumisen estimointi (Cao, 2008). Erään ohjelmistoprojektin ohjelmointivirheiden korjausten keskimääräinen estimointivirhe oli 28 %, kun taas uusien ominaisuuksien valmistumisen keskimääräinen estimointivirhe oli 19 %.

Ohjelmistokehittäjien keskuudessa estimointia saatetaan pitää Scrumin käytänteistä vähiten hyödyllisenä ajankäyttönä. Opiskelijoilla tehdyssä kyselytutkimuksessa estimointia pidettiin vähiten tärkeänä Scrumin käytänteenä (Mahnič, 2011). Kyselytutkimuksessa opiskelijat arvoivat 12 Scrumin käytänteen tärkeyttä asteikoilla 1-5, jossa 1 on vähiten tärkeä ja 5 on erittäin tärkeä. Tutkimuksessa vähiten tärkeänä pidettiin tarkkaa velositeetin estimointia keskiarvolla 3,54 ja toiseksi vähiten tärkeänä tarkkaa tarinapisteiden estimointia keskiarvolla 3,56. Luvut ovat selkeästi keskitason yläpuolella, mutta monet opiskelijat pitivät estimointia ja suunnittelua tuottamattomana hallinnollisena työnä. Monet opiskelijat eivät myöskään luottaneet tekemiinsä estimaatteihin. Tärkeimpänä Scrumin käytänteenä pidettiin tiimityötä ja tiimin sisäistä kommunikaatiota keskiarvolla 4,82. Kyselytutkimus suoritettiin 51 opiskelijalla.

2.6 Estimoinnin epäonnistuminen

Mikäli projektin vastuuhenkilöt huomaavat, että projektin kolme elementtiä eivät pysy estimoiduissa rajoissa, aletaan elementtien valintakolmion (kuva 2) keskipistettä siirtää, eli luovutaan muiden elementtien estimoidusta tavoitteesta, jotta jokin toinen elementti saadaan projektissa toteutumaan (Dalcher, 2014, s. 32). Tämä keskipisteen siirtäminen on osa vaatimustenhal-

lintaa. Lisäksi projektin aikana voidaan alkaa vaatia enemmän ominaisuuksia kuin mihin alun perin on sitouduttu ja ne kasvattavat projektin kehitysaikaa ja kustannuksia, jolloin vaatimustenhallinta on perusteltua.

Myös projektin suuri monimutkaisuus lisää estimoinnin epävarmuutta ja epäonnistumisen riskiä (Fitsilis et al., 2015). Monimutkaisuuden määritelmästä ei ole selkeää yksimielisyyttä. Monimutkaisuudella voidaan tarkoittaa järjestelmän monimutkaisuutta tai järjestelmän käyttäjän kokemaa monimutkaisuutta järjestelmää käytettäessä. Järjestelmän monimutkaisuuden mittarina on käytetty esimerkiksi koodirivien tai toiminnallisuuspisteiden (Function Point) lukumäärää.

Vaatimustenhallinnalla tarkoitetaan toimintaa, jossa kartoitetaan asianosaisten odotukset ja vaatimukset projektin suhteen, jotta projektille osataan ohjata oikea määrä resursseja (Fageha & Aibinu, 2013). Vaatiumstenhallinnalla pyritään siis kartoittamaan projektin onnistumisen kannalta välttämätön työ. Vaatimustenhallinnalla voidaan viitata projektin alussa tehtävään vaatimusmäärittelyyn tai projektin aikana tehtävään projektin ohjaamiseen. Projektin aikana tehtävällä vaatimustenhallinnalla pyritään ohjaamaan epäonnistumisriskissä olevaa projektia onnistumiseen kartoittamalla tarkemmin vain välttämättömimpiä työvaiheita ja ominaisuuksia (Duarte, 2016, s. 69). Ohjelmistoprojektin alussa ei välttämättä vielä tiedetä kovinkaan tarkasti mitkä toteutettavat ominaisuudet ovat välttämättömiä. Vaatimustenhallintaan tulisi kiinnittää huomiota mikäli estimaattien perusteella projektin tavoitteet näyttävät epäonnistuvan.

3 Ketterä ohjelmistokehitys

Tässä opinnäytetyössä tarkastellaan estimointia pääasiassa ketterien ohjelmistokehitysmenetelmien näkökulmasta. Ketterät ohjelmistokehitysmenetelmät ovat nykyisin vallitsevia ohjelmistokehitysmenetelmiä organisaatioissa (Version One Inc, 2019). Ketterä ohjelmistokehitys on kattotermi ohjelmistokehitysmetodeille ja projektinhallintakehyksille, jotka noudattavat ketterän ohjelmistokehityksen julistuksen (Manifesto for Agile Software Development) arvoja ja periaatteita (Agile Alliance, 2019). Ketterän kehityksen arvoihin kuuluu asettaa yksilöt ja kanssakäymiset menetelmien ja työkalujen edelle,

toimiva ohjelmisto kattavan dokumentaation edelle, asiakasyhteistyö sopimusneuvottelujen edelle ja muutoksiin mukautumisen suunnitelmassa pysymisen edelle (Agile Finland, 2001; Cohn, 2006, s 21). Eräs ketterän ohjelmistokehityksen periaate on iteratiivisuus. Iteratiivisuudella tarkoitetaan ohjelmistoprojektin kehittämistä lyhyissä, parin viikon mittasissa, kehitysjaksoissa (Cohn, 2006, s 21). Iteraation alussa ohjelmistotiimi päättää iteraation aikana kehitettävistä ominaisuuksista ja iteraation jälkeen valmiit ominaisuudet esitellään asiakkaalle. Ohjelmistotiimin vastuulla on luoda estimaatti projektille projektin alussa ja päivittää estimaattia jokaisen iteraation jälkeen. Iteratiivisuutta hyödynnetään ketterän ohjelmistoprojektin estimaattien laatimisessa, joten siihen palataan opinnäytetyön myöhemmässä vaiheessa.

3.1 Scrum-projektinhallintakehys

Tässä opinnäytetyössä keskitytään tarkastelemaan Scrumia, koska Scrum, tai sen muunnelmat, ovat erittäin yleisesti käytettyjä projektinhallintakehyksiä. Ketteristä projektihallintamentelmistä Scrum oli vuonna 2019 suosituin projektinhallintakehys 54 % osuudella (Version One Inc, 2019).

Scrumin määritelmänä on olla iteratiivinen, läpinäkyvä ja sopeutuva projektinhallintakehys (ScrumGuides.org, 2018). Tässä opinnäytetyössä tarkastellaan pääasiassa Scrumin tuotteen kehitysjonon (Product Backlog) estimointia. Tuotteen kehitysjonolla tarkoitetaan priorisoitua listaa Scrum-projektin kehitettävistä ominaisuuksista. Scrum-projektinhallintakehyksen opas ei erikseen määrittele Scrumin kanssa käytettävän tuotteen kehitysjonon sisällön muotoa (ScrumGuides.org, 2018). Opas määrittelee ainoastaan, että jokaisella kehitysjonoon kirjatulla vaatimuksella tulee olla kuvaus, järjestys, estimaatti ja arvo. Arvoa (Business value, Value) ei määritellä erikseen Scrumin oppaassa ja terminä sitä pidetään epämääräisenä, mutta erään määritelmän mukaan arvo on vaatimuksen ominaisuus, joka antaa hyötyä tuotteen omistajan (Product Owner) organisaatiolle esimerkiksi kasvattamalla tehokkuutta, asiakaskuntaa, asiakastyytyväisyyttä tai tuottamalla uusia ominaisuuksia, joista asiakkaat ovat valmiita maksamaan (Verwijs, 2014). Arvo voidaan jakaa esimerkiksi liiketalouden arvoksi, markkina-arvoksi, tehokkuusarvoksi, asiakasarvoksi tai tulevaisuusarvoksi. Arvoa voidaan siis

tuottaa esimerkiksi uudella ominaisuudella, laajentamalla alustatukea, kehittämällä virheiden seulontaa, parantamalla tuotteen käytettävyyttä tai vähentämällä teknistä velkaa.

Yleisesti käytetty tapa kirjata vaatimukset kehitysjonoon ovat käyttäjätarinat, joihin myös tässä opinnäytetyössä keskitytään (Cohn, 2004, s 173). Scrumin opas ei myöskään määrittele vaatimusten estimointitapaa, mutta tässä opinnäytetyössä keskitytään käyttäjätarinapisteytyksen käyttöön vaatimusten estimoinnissa.

3.2 Estimointi ja suunnittelu Scrum-projektinhallintakehyksessä

Ketterän ohjelmistokehityksen termistöllä on useita eri tulkintoja, joten on syytä selventää, mitä eri termit tarkoittavat tässä opinnäytetyössä. Tässä luvussa esitellään ja tarkennetaan Scrumin estimoinnissa ja suunnittelussa käytettävää termistöä.

3.2.1 Käyttäjätarinat

Idea käyttäjätarinoista on alun perin luotu osaksi Extreme Programming -ohjelmistokehitysmetodologian vaatimustenhallintayprosessia (Beck, 2004). Käyttäjätarinoita on kuitenkin alettu käyttää myös monien ketterien projektinhallintakehysten, kuten Scrumin, kanssa (Cohn, 2004, s 165).

Käyttäjätarinoilla (User Stories) tarkoitetaan asiakkaan esittämää vaatimusta, joka tuottaa arvoa ohjelmiston käyttäjälle tai hankkijalle (Cohn, 2004, s 4). Käyttäjätarinat muodostuvat kolmesta osasta: lyhyestä kirjallisesta kuvauksesta, joka toimii apuvälineenä suunnittelussa ja muistutuksena käyttäjätarinan sisällöstä, keskustelusta, jossa pyritään kartoittamaan käyttäjätarinan yksityiskohdat sekä testeistä, jotka toimivat yksityiskohtaisempana kirjallisena kuvauksena ja määritelmänä milloin käyttäjätarina on valmis (Cohn, 2004, s 4). Käyttäjätarinoiden kuvaukset kirjoitetaan yleensä paperilapulle ja ne ovat yleensä muotoa: "[Käyttäjän rooli] voi tehdä järjestelmällä jotakin", esimerkiksi: "Käyttäjä voi lisätä ansioluettelon sivulle."Käyttäjätarinat kirjoitetaan aina asiakkaan kielellä niin, että asiakas ymmärtää käyttäjätarinoiden sisällön. Teknistä kieltä tulee välttää käyttäjätarinoissa ellei asiakas

tai projektin luonne sitä erikseen vaadi.

Käyttäjätarinoita ei tule sekoittaa esimerkiksi käyttötapauksiin (Use Case), käyttöskenaarioihin (Scenario) tai IEEE 830 -standardin mukaisiin vaatimusmäärittelyihin (IEEE 830 software requirements specification) (Cohn, 2004, s 133). IEEE 830 -standardin mukaisissa vaatimusmäärittelyissä kuvataan mitä järjestelmän tulisi tehdä, kun taas käyttäjätarinat kuvaavat mitä käyttäjä haluaa tehdä. Vaatimusmäärittelyjen suositellaan olevan muodossa: "Järjestelmä [verbi]..."Käyttäjätarinat taas ovat yleensä muodossa: "Käyttäjä haluaa [verbi]..."Käyttötapauksissa kuvataan erittäin yksityiskohtaisesti vaihe vaiheelta käyttäjän ja järjestelmän toimintaa. Käyttöskenaariot taas ovat yksityiskohtauksia kuvauksia käyttäjän käyttökokemuksesta järjestelmän kanssa. Käyttötapaukset ja käyttöskenaariot ovat käyttäjätarinoita raskaampia keinoja kuvata käyttäjän vaatimuksia. Tässä opinnäytetyössä käsitellään lähes ainoastaan käyttäjätarinoita.

3.2.2 Käyttäjätarinapisteet

Eräs tapa estimoida käyttäjätarinoiden työläys on käyttää käyttäjätarinapisteitä (Story Point) (Cohn, 2016, 2004, s 87). Käyttäjätarinapisteytys on suhteellinen estimointimenetelmä - Yhden käyttäjätarinan käyttäjätarinapisteytys on suhteutettu muiden käyttäjätarinoiden käyttäjätarinapisteisiin. Käyttäjätarinapisteiden asteikkona käytetään usein Fibonaccin lukujonon kaltaista arvoasteikkoa esimerkiksi: 1, 2, 3, 5, 8, 13, 20, 40, 80 (Cohn, 2004, s 93). Fibonaccin kaltaisessa asteikossa kolmen pisteen käyttäjätarina ajatellaan noin 50 % työläämmäksi kahden pisteen tarinaan verrattuna ja kolme kertaa työläämmäksi yhden pisteen tarinaan verrattuna. Lineaarisesti kasvavaan (1,2,3,4,5,6,7...) asteikkoon verrattuna Fibonaccin lukujonon kaltainen asteikko kuvaa paremmin estimoinnin epävarmuutta estimoitavan ominaisuuden laajuuden kasvaessa.

Työläyden lisäksi käyttäjätarinapisteiden mittarina on aiemmin käytetty myös esimerkiksi ideaaliaikaa (ideal time) (Buglione & Abran, 2007; Cohn, 2004, s 87). Nykyään käyttäjätarinapisteitä tulisi ajatellaan ainoastaan työläyttä kuvaavana suhteellisena estimointitapana (Cohn, 2016).

Käyttäjätarinapisteytystä ei tule sekoittaa esimerkiksi iteraatiopistey-

tykseen (Sprint Point), toiminnallisuuspisteytykseen (Function Point) tai käyttötapauspisteytykseen (Use Case Point) (Popli & Chauhan, 2013; Borade & Khalkar, 2013). Näitä ei käsitellä tässä opinnäytetyössä.

3.2.3 Velositeetti

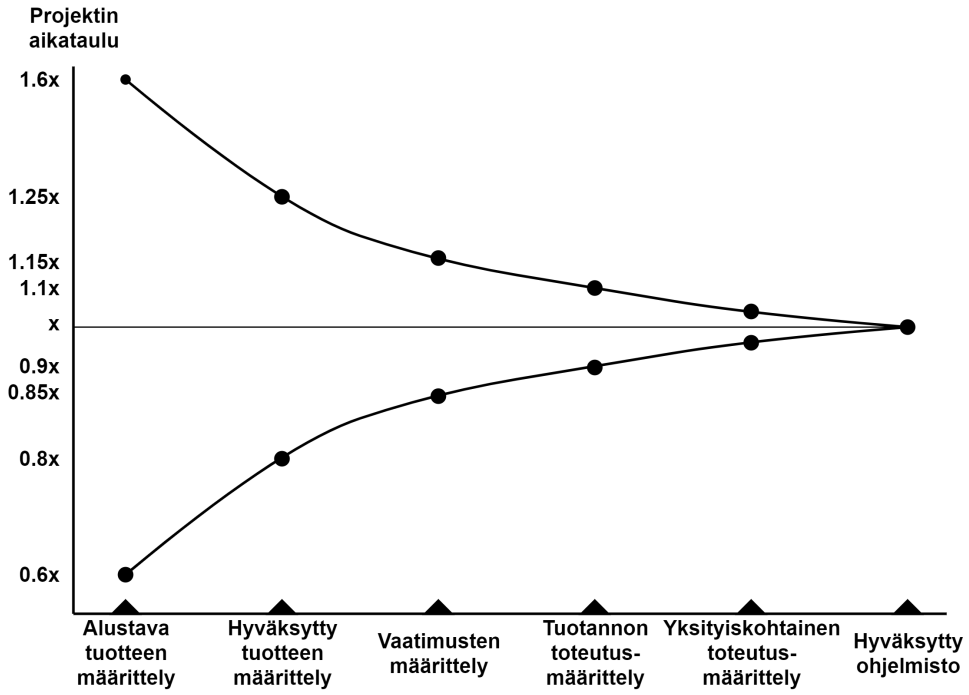
Velositeetti on mittari, joka mittaa ohjelmistotiimin edistystahtia (Cohn, 2006, s 38). Velositeetin avulla estimoidaan, kuinka monta käyttäjätarinaa ohjelmistotiimi saa valmiin määritelmän (Definition of Done) mukaisesti valmiiksi tulevan sprintin aikana. Velositeetti muodostetaan summaamalla iteraation valmiiksi saatujen käyttäjätarinoiden käyttäjätarinapisteet yhteen. Velositeetti päivitetään yleensä jokaisen iteraation jälkeen, jolloin kehitystiimi voi arvioida, mitkä käyttäjätarinat saadaan valmiiksi seuraavan iteraation aikana.

Velositeettia tulisi kuitenkin ajatella lukuvälinä, joka tarkentuu projektin edetessä (Cohn, 2006, s 177). Projektin alussa tiimin velositeetti on hyvin epätarkka, koska tietoa esimerkiksi projektin luonteesta tai kehitystiimin suorituskyvystä ei ole vielä riittävästi saatavilla. Projektin edetessä velositeetin kannalta olennainen tieto tarkentuu, joten myös velositeetin lukuväli kaventuu eli velositeetin epävarmuus pienenee. Velositeetin lukuvälin kaventumisen mallia kutsutaan epävarmuusmalliksi (kuva 5) (Cone of Uncertainty) (Cohn, 2006, s 177).

Velositeettia voidaan hyödyntää projektin valimstumisaikataulun estimoinnissa jakamalla projektin kehitysjonon jäljelle jääneiden käyttäjätarinoiden käyttäjätarinapisteiden summa velositeetilla ja kertomalla jakolaskun osamäärä iteraation pituudella.

3.2.4 Suunnittelupokeri

Suunnittelupokeri (Planning Poker, Scrum Poker) on eräs Scrumin, ja myös muiden ketterien ohjelmistokehitysmetodien, kanssa käytettävä ryhmäestimointimenetelmä. Suunnittelupokerin sanotaan tuottavan nopeasti tarkkoja estimaatteja perustuen asiantuntijoiden keskusteluun (Cohn, 2006, p 56). Suunnittelupokeriin osallistuvat kaikki kehitystiimin jäsenet. Monesti asiakkaan edustaja myös osallistuu suunnittelupokeriin tarkkailijan ja kysymyksiin



Kuva 5: Estimoinnin epävarmuusmalli. Projektin alussa estimaattien yläraja on 60 % suurempi ja alaraja 40 % pienempi kuin laskettu arvio. Estimaattien välit kuitenkin kaventuvat projektin edetessä (Cohn, 2006, s 179).

vastaajan roolissa.

Suunnittelupokerissa jokaiselle estimointiin osallistuvalle annetaan korttipakka, jonka kortteihin on painettu estimointiasteikon luvut (Cohn, 2006, s 56). Jokainen kehitystiimin jäsen valitsee käsiteltävälle käyttäjätarinalle kortin korttipakasta kortin, joka kuvaa hänen mielestään parhaiten käyttäjätarinan vaativuutta. Kortit paljastetaan muille tiimin jäsenille vasta, kun kaikki ovat valinneet sopivan estimaatin. Mikäli estimaatit eroavat toisistaan, pienimmän ja suurimman estimaatin valinneet henkilöt keskustelevat rakentavasti toistensa kanssa, miten ovat päätyneet kyseiseen estimaattiin. Tämän jälkeen suoritetaan uusi arvointikierros. Kierroksia jatketaan niin kauan, kunnes jokainen tiimin jäsen valitsee saman luvun korttipakasta. Estimointia jatketaan, kunnes jokainen oleellinen käyttäjätarina on käyty läpi.

Suunnittelupokeri tuottaa tarkempia estimaatteja kuin vapaa ryhmäkeskustelu sellaisten käyttäjätarinoiden estimoinnissa, joista kehitystiimillä

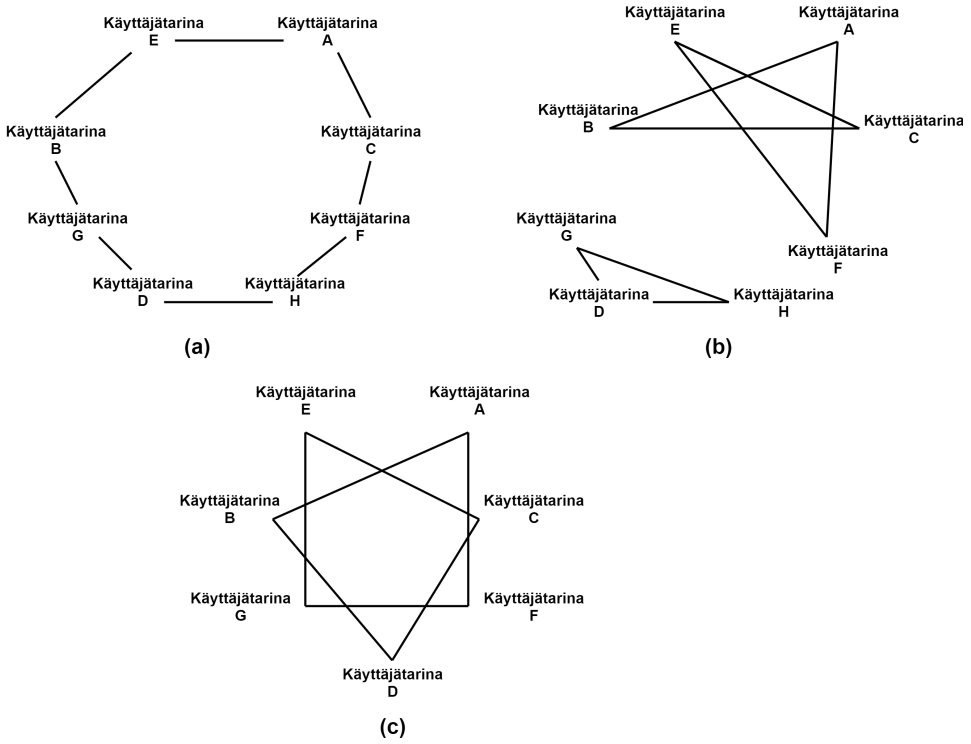
on aikaisempaa kokemusta (Haugen, 2006). Kehitystiimille tuntemattomien käyttäjätarinoiden estimoinnissa tehdään kuitenkin enemmän virhearvioita kuin vapaassa ryhmäkeskustelussa riippumatta käyttäjätarinan koosta. Suunnittelupokeri saattaa tuottaa suurempia eroja isoimpien ja pienimpien estimaattien välille kuin vapaa keskustelu. Suuremmat erot estimaattien välillä saattavat johtaa myös suurempaan estimointivirheeseen.

Nuorella kehitystiimillä suunnittelupokerin ryhmäkeskustelu tuottaa yli-optimistisia estimaatteja verrattuna tilastollisesti yhdistettyyn usean asiantuntijan itsenäisesti tekemään estimaattiin (Mahnič & Hovelja, 2012). Asiantuntijoiden tekemään tilastollisesti yhdistettyyn estimaattiin verrattuna suunnittelupokerin ryhmäkeskustelu tuottaa hieman realistisempia estimaatteja.

3.2.5 Triangulaatio

Käyttäjätarinapistepohjaista estimointimenetelmää kutsutaan myös suhteelliseksi estimointitavaksi. Suhteellinen estimointi tarkoittaa, että esimerkiksi käyttäjätarinoita estimoidessa hyödynnetään muiden käyttäjätarinoiden estimaatteja vertailemalla estimoitavia käyttäjätarinoita toisiinsa (Cohn, 2004, s. 90). Tätä vertailua kutsutaan triangulaatioksi (Triangulate).

Yksinkertaisimmillaan triangulaatiossa estimoitavaa käyttäjätarinaa vertaillaan kahteen käyttäjätarinaan (Miranda et al., 2009). Tällainen yksinkertainen vertailu kuitenkin luo epäjohdonmukaisen triangulaation, mikä lisää estimoinnin epätarkkuutta. Triangulaatiosta saadaan johdonmukaisempi tekemällä useampia vertailuja. Hyvän triangulaation tulisi olla tasapainoinen ja yhtenäinen. Tasapainoisuudella tarkoitetaan, että jokaiselle käyttäjätarinalle tehdään yhtä monta vertailua. Tällä varmistetaan, että mitään yhtä käyttäjätarinaa ei painoteta muita enemmän. Yhtenäisyydellä tarkoitetaan, että jokaista käyttäjätarinaa vertaillaan suorasti tai epäsuorasti kaikkiin muihin käyttäjätarinoihin (kuva 6). Epäyhtenäinen triangulaatiomalli johtaa pirstaloitumiseen, mikä tarkoittaa, että jokaisella osajoukolla olisi oma estimointitarkkuus. Tällainen epäyhtenäinen estimointitarkkuus lisää estimoinnin epätarkkuutta.



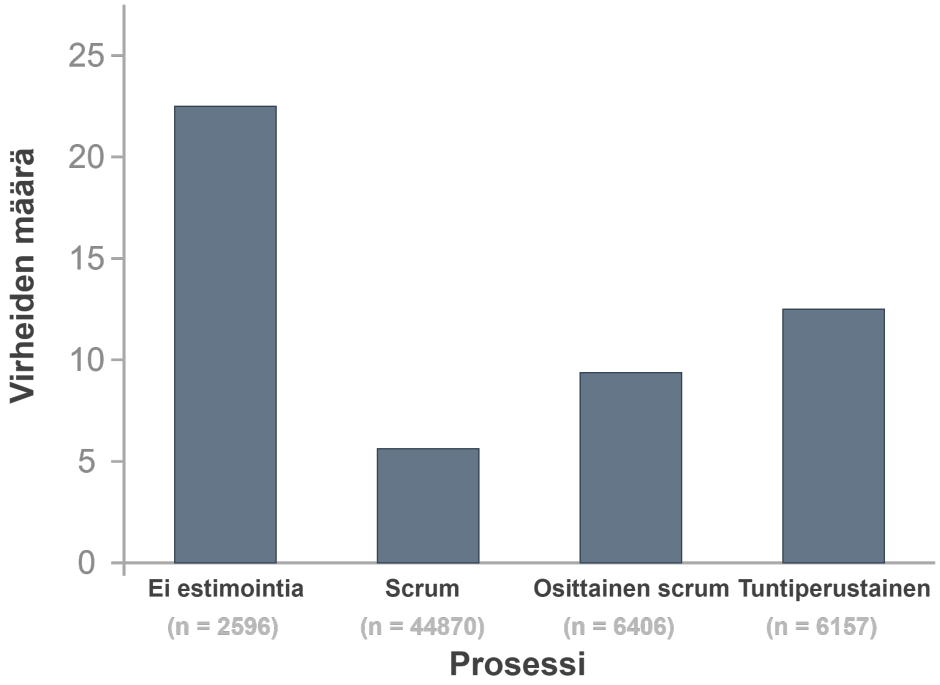
Kuva 6: Mallit a ja c ovat yhtenäisiä triangulaatiomalleja. Malli b on epäyhtenäinen triangulaatiomalli (Miranda et al., 2009).

4 NoEstimates-liike

NoEstimates-liikkeeksi kutsutaan tässä tutkielmassa ihmisjoukkoa, joka jakaa estimointinäkemyksiään internetissä NoEstimates-avainsanalla. Liikkeen jäsenet pitävät esitelmiä ja kirjoittavat blogeja ja kirjoja estimointinäkemyksistään. Vaikka liikkeen nimestä voisi päätellä, liike ei tuomitse kaikkea estimointia (Duarte, 2016). Liikkeen tarkoitus on edistää ajattelua, että estimointi ei aina ole välttämätöntä ja saada ihmiset ajattelemaan, milloin estimointi on järkevää. Lisäksi liike pitää yllä keskustelua vaihtoehtoisista tavoista tehdä estimaatteja (Karhatsu, 2015). NoEstimates-keskusteluun tulisi suhtautua rakentavana kehittämiskeskusteluna, eikä jonkin yhden tietyn ajattelutavan levittäjänä.

On kuitenkin selvää, että estimointi on välttämätön osa laatuun pyrkivää ohjelmistoprojektia (CA Technologies, 2017). Estimointi toimii mm.

osana virheiden seulontaa, projektin ohjaamista ja työrutiinin ylläpitämistä. Estimaatit saattavat painostaa kehittäjiä miettimään ja aikatauluttamaan tekemisiään, mikä auttaa työrutiinin ylläpidossa ja johtaa tehokkaampaan työskentelyyn. Virheiden, jotka saattavat johtua esim. väärinymmärretyistä ominaisuuskuvauksista, seulonnassa estimointi ja estimaattien päivittäminen saattaa auttaa ylläpitämään keskustelua toteutettavien ominaisuuksien sisällöstä ja toteutustavasta, mikä edistää virheiden löytymistä. Täysin ilman estimointia olevissa projekteissa virheiden määrä kasvaa moninkertaiseksi (kuva 7).



Kuva 7: Ohjelmointivirheiden määrä projektissa kasvaa huomattavasti, kun estimointi jätetään kokonaan tekemättä (CA Technologies, 2017). Pystyakselilla virheiden määrä keskimäärin julkaisun yhteydessä ja n on tarkasteltavien projektien lukumäärä jokaisessa kategoriassa.

Estimoinnin ajatellaan olevan monesti yrityksessä tapa, joka ei tuota arvoa (Karhatsu, 2015). Arvoa tuottamattomat tavat tuottavat hukkaa (waste). Hukalla tarkoitetaan Lean-ohjelmistokehityksessä keskeneräistä tai turhaa työtä (Poppendieck, 2003). Keskeneräisellä työllä tarkoitetaan usein ominai-

suuksia, jotka eivät ole asiakkaan käytettävissä, tai ne toimivat puutteellisesti. Niin kauan kuin ominaisuutta ei ole viety tuotantoympäristöön, se ei tuota arvoa asiakkaalle ja on näin ollen hukkaa. Hukaksi luokitellaan myös kaikki sellaiset ominaisuudet, jotka eivät vastaa asiakkaan tarpeita. Turhaksi työksi voidaan luokitella esimerkiksi, jos ohjelmistokehittäjä joutuu jatkuvasti keskeytetyksi tai vaihtamaan työtehtäviä usean eri projektin välillä (Task Switching). Ohjelmistokehittäjä joutuu tällöin käyttämään aikaa keskeytetyn projektin pariin palaamiseen ja siihen keskittymiseen. Ohjelmistokehittäjän jokainen keskeytys ja projektin pariin palaamiseen kuluva aika määritellään hukaksi, koska se ei tuota arvoa asiakkaalle.

NoEstimates-keskustelun eräänä keskeisenä teemana on käyttäjätarinoiden estimointi, ja hyvän käyttäjätarinan kuvaamiseen käytettävä INVEST-sääntö. INVEST-säännöllä (Independent, Negotiable, Valuable, Estimable, Small, Testable) tarkoitetaan, että hyvän käyttäjätarinan tulee olla itsenäinen, joustava, asiakkaalle arvoa tuottava, estimoitavissa oleva, pieni ja testattavissa (Wake, 2003). Säännön alisäännöistä NoEstimates-keskustelussa keskitytään sääntöön ”estimoitavissa oleva”, joka korvataan säännöllä "välttämätön"(essential) (Duarte, 2016, s. 71). Käyttäjätarinan välttämättömyydellä tarkoitetaan, että jokaisen käyttäjätarinan tulee olla välttämätön osa projektia. Säännöllä pyritään siihen, että projektissa tulee tehdä ainoastaan välttämättömiä ja arvoa tuottavia asioita, jotka johtavat onnistuneeseen projektiin.

Kuten luvussa 2.2 todettiin, estimoinnilla pyritään vastaamaan mm. seuraaviin kysymyksiin:

- Mikä on paras projekti toteutettavaksi kustannusten ja hyödyn kannalta?
- Kuinka suuret ovat kehityskustannukset?
- Milloin saamme tuotteen?
- Kuinka pitkäksi ajaksi olemme sitoutuneet?
- Ovatko käyttäjätarinat tarpeeksi pieniä?
- Kuinka paljon voimme toteuttaa iteraation aikana?

- Miten saamme parhaan hinta-hyötysuhteen?

Näihin kysymyksiin NoEstimates-liike pyrkii löytämään nykyistä parempia vastaustapoja (Arlen, 2019). Seuraavaksi esitellään NoEstimates-liikkeen ideoita, miten vastata näihin kysymyksiin estimointia paremmin.

4.1 Nelivaiheinen NoEstimates-menetelmä

Yksinkertaisimillaan NoEstimates on nelivaiheinen menetelmä, jolla korvataan projektin estimointivaiheet (Overeem, 2015).

1. Valitse tärkein työtehtävä ensimmäiseksi. Tärkeimmäksi työtehtäväksi määritellään tehtävä, joka tuottaa eniten "arvoa". Priorisoi työtehtävät, mutta ole avoin muutoksille.

2. Pilko työtehtävä riskineutraaleiksi paloiksi. Riskineutraaleilla paloilla tarkoitetaan, että palan toteutuksen myöhästyminen ei vaaranna projektin muita tavoitteita.

3. Toteuta palat yksi kerrallaan ja noudata kehityksessä Definition of Done -periaatetta.

4. Ota opiksi edellisistä iteraatioista ja toista kohtia 1-3, kunnes tuote täyttää laatuvaatimukset.

Nelivaiheisen menetelmän kuvauksesta ei kuitenkaan tarkasti selviä, mitä ”riskineutraaleilla paloilla” tarkoitetaan. Riskineutraalilla palalla tarkoitetaan mahdollisesti luvussa neljä esitetyn INVEST-säännön kohtaa ”itsenäinen”. Itsenäisyydellä tarkoitetaan, että käyttäjätarina ei saa olla riippuvainen muista käyttäjätarinoista eli yhden käyttäjätarinan toteutusaikataulu tai toteuttamatta jättäminen ei saa vaikuttaa muiden käyttäjätarinoiden toteutukseen.

4.2 Käyttäjätarinoiden pilkkominen

Käyttäjätarinoiden pilkkomisella (Story Splitting) tarkoitetaan isojen käyttäjätarinoiden pilkkomista pienempiin, jotta käyttäjätarinoista saadaan esimerkiksi eroteltua arvokkaat osat käyttäjätarinoiden vähemmän arvokkaista osista (Wake, 2005). Pilkkominen auttaa kehittäjiä työvaiheiden priorisoinnissa,

koska arvokkaimmat työvaiheet voidaan toteuttaa ensin. Lisäksi pilkkominen auttaa löytämään ominaisuuksista turhia työvaiheita, joiden poisjättäminen on helpompi ja halvempi vaihtoehto kuin niiden toteuttaminen (Karhatsu, 2015).

Ketterässä kehityksessä isoja käyttäjätarinoita tulisi välttää ja liian isot käyttäjätarinat tulisi pilkkoa pienemmiksi (Duarte, 2016, s. 69). Scrumissa isoksi käyttäjätarinaksi luokitellaan käyttäjätarinat, jotka oletettu kesto on yli puoli iteraatiota ja tällaiset käyttäjätarinat tulisi pilkkoa pienempiin tarinoihin. NoEstimates-keskustelussa ehdotetaan, että käyttäjätarinoita tulisi mahduttaa kahden viikon iteraatiojaksoon 6-12. Tämän lisäksi erikokoisten käyttäjätarinoiden tulisi jakautua tasaisesti pitkin projektia, jotta valmistuneiden käyttäjätarinoiden käyttö estimoinnissa olisi luotettavampaa.

4.3 Estimointi käyttäjätarinoiden lukumäärällä

NoEstimates-keskustelussa ehdotetaan, että pisteiden sijaan estimointiin käytettäisiin käyttäjätarinoiden lukumäärää (Duarte, 2012). Käyttäjätarinoiden lukumäärällä estimoitaessa toteutettavat käyttäjätarinat pilkotaan niin pieniin käyttäjätarinoihin, että yksi kehittäjä saa käyttäjätarinan valmiiksi yhden iteraation aikana. Estimoinnin perustana käytetään oletusta, että jokainen käyttäjätarina on vakiokokoinen ja yksi kehittäjä saa keskimäärin valmiiksi yhden käyttäjätarinan iteraation aikana. Suuria ja matalan prioriteetin käyttäjätarinoita voidaan estimoida lisäämällä käyttäjätarinaan arvio, kuinka monta iteraatiota käyttäjätarinan valmiiksi saaminen kestää. Keskustelussa perustellaan, että käyttäjätarinapisteet eivät anna mitään lisäarvoa projektille. Eräässä tutkimuksessa estimoitiin projektin valmistumisaikataulua käyttäjätarinapisteillä sekä muuttamalla käyttäjätarinapisteet vakiokokoisiksi käyttäjätarinoiksi (Duarte, 2017). Vakiokokoisilla käyttäjätarinoilla luotu estimaatti oli lähes identtinen käyttäjätarinapisteiden avulla luotuun estimaattiin nähden (ero noin 8 %). Kyseessä ei kuitenkaan ole tieteellinen tutkimus, joten siihen tulee suhtautua varauksella.

Suoraan käyttäjätarinoilla estimoidessa Scrum-tiimin ei tarvitse käyttää aikaa käyttäjätarinoiden pisteyttämiseen ja projektinhallinnasta saadaan yksinkertaisempi. Käyttäjätarinoiden lukumäärän käyttö estimaatteina no-

peuttaa uuden estimoinnin tekemistä, mikäli projekti muuttuu. Isosta käyttäjätarinasta voidaan estimoida käyttäjätarinaan sisältyvien ominaisuuksien lukumäärä. Ison käyttäjätarinan pilkkomisen vaikutus saadaan näkyviin esim. projektin valmistumisaikataulussa ilman, että kehittäjien tarvitsee pisteyttää pilkkomisesta syntyneitä käyttäjätarinoita.

Toteutetut ja testatut tarinat (running-tested-stories, running-tested-features) ovat ohjelmistoprojektissa käyttäjätarinoita, jotka on viety tuotantoon ja ovat asiakkaan käytettävissä, eli käyttäjätarinat on tehty valmiin määritelmän mukaisesti (Jeffries, 2004). Toteutetut ja testatut tarinat ovat mittari, joka kuvaa ohjelmistoprojektin todellista edistymistä. Kaikki käyttäjätarinoiden eteen tehty työ on merkityksetöntä niin kauan kunnes käyttäjätarina ei ole asiakkaan käytettävissä (Duarte, 2016, s. 68). NoEstimates-keskustelussa toteutettujen ja testattujen tarinoiden lukumäärää pidetään tärkeimpänä edistyksen mittarina eikä esimerkiksi tehtyjä työtunteja tai käyttäjätarinapisteitä.

Erään esimerkkiprojektin datan perusteella käyttäjätarinoiden lukumäärällä estimoitavan projektin viiden iteraation jälkeen tehty valmistumisajan ennusteen virhe oli 4 % todelliseen valmistumisaikaan verrattuna, kun taas käyttäjätarinapisteillä estimoituna projektin virhe oli 13 % (ConfEngine, 2015). Käyttäjätarinoiden lukumäärällä estimoitaessa projekti valmistui aikaisemmin kuin mitä estimaattien perusteella tehty ennuste ennusti, kun taas käyttäjätarinapisteillä ennustaessa projekti oli myöhässä ennusteesta. Kyseessä ei kuitenkaan ole tieteellinen tutkimus ja otannassa on vain yksi projekti, joten tulokseen tulee suhtautua varauksella.

4.4 Minimalistinen projektin aloitus

Eräs NoEstimates-liikkeen ehdotus on, että ulkoistettuja Ohjelmistoprojekteja voitaisiin aloittaa minimalistisena, halpana ja lyhyellä irtisanoutumisajalla (Heusser, 2013). Pienillä vaatimusmäärillä saadaan minimoitua asiakkaan riskit jolloin asiakkaat saadaan lähtemään helpommin mukaan projektiin ilman tarjousvaiheen laajoja estimaatteja. Alun minimalistisilla vaatimuksilla saadaan tehtyä minimalistinen toimiva tuote, jonka kehittämisestä kertynyttä kokemusta ja tietoa voitaisiin käyttää hyväksi estimaatin tekemiseen

projektin laajentuessa. Jokaisen kehitysiteraation jälkeen asiakas voi päättää, haluaako hän jatkaa projektia. Tämänkaltaisessa toiminnassa ohjelmistoyritykselle riskinä on asiakkaan nopea irtisanoutumisaika projektista. Irtisanoutumisen jälkeen asiakkaalle jää toimiva osa ohjelmistoa, jolloin asiakkaan projektiin käyttämät resurssit eivät mene hukkaan.

Tällaisessa projektimuodossa asiakas voisi palkata kehitystiimin esimerkiksi kuukaudeksi tekemään jokin pieni mutta käytettävä ominaisuus ohjelmistosta (Karhatsu, 2015). Tämän kehitysvaiheen jälkeen asiakas päättäisi, että jatketaanko sopimusta. Tällaisen projektiajattelun tarkoitus on hakea tietoa projektin luonteesta ennusteiden tekemiseksi. Tiimi ja yrityksen omistajat voisivat käyttää projektista saatua tietoa ja kokemusta tulevaisuudessa tarkempien ennusteiden tekemiseen, vaikka asiakas ei jatkaisikaan projektin rahoittamista kuukauden kehitysvaiheen jälkeen.

5 Tutkimusmenetelmä

Empiirinen tutkimus on kokonaisuus, joka sisältää monia eri vaiheita, jotka ovat riippuvuussuhteessa toisiinsa (Hirsijärvi & Hurme, 2000, s. 14). Jokainen tutkimus on erilainen, eikä kaikkia tutkimuksia voida kuvata täysin tarkasti. Jokaisesta empiirisestä tutkimuksesta löytyvät kuitenkin vaiheet, jossa määritellään alustava tutkimusongelma, perehdytään aiheeseen ja täsmennetään ongelmaa, kerätään aineisto ja analysoidaan se, ja tehdään raportti ja johtopäätökset tutkimustuloksista.

Tutkimusmenetelmäksi valittiin puolistrukturoitu haastattelumenetelmä, jota kutsutaan tässä tarkemmin teemahaastatteluksi. Puolistrukturoidulle haastattelumenetelmälle ei ole yhtä oikeaa määritelmää, mutta menetelmälle ominaista on, että jokin haastattelun näkökohta on lyöty lukkoon, mutta ei kaikkia (Hirsijärvi & Hurme, 2000, s. 47). Teemahaastattelulle on ominaista ettei se edellytä tiettyä kokeellisesti aikaansaatua yhteistä kokemuksta vaan siinä oletetaan, että kaikkia yksilön kokemuksia, ajatuksia, uskomuksia ja tunteita voidaan tutkia. Kysymykset, kysymysten muotoilu ja kysymysten järjestys eivät ole tiukasti rajattuja teemahaastattelussa.

Teemahaastattelu valittiin tutkimusmenetelmäksi, koska haluttiin kuulla ohjelmistoalan ammattilaisten estimointikokemuksista ja käytännön käy-

tänteistä ohjelmistoyrityksissä. Tutkimuksessa haluttiin olla myös avoimia mahdollisille uusille ideoille, joista haastateltavat voisivat kertoa haastattelussa.

Haastatteluja varten kehitettiin kysymyspohja (liite A), jota haastattelija sovelsi sen mukaan, miten haastattelutilanne eteni. Haastattelutilanne pyrittiin pitämään mahdollisimman lähellä huoletonta keskustelutilannetta, joten kysymyspohjaa ei noudatettu erityisen tiukasti. Tyypillinen haastattelutilanne eteni suunnilleen seuraavalla tavalla: Haastattelun alussa haastateltavat valitsivat mielessään yhden projektin, josta he kertoivat estimointiin liittyviä kokemuksia. Haastattelussa pyrittiin löytämään vastauksia esimerkkiprojektista ainakin seuraaviin kysymyksiin: miten estimoitiin, kuinka paljon aikaa käytettiin estimointiin, mihin estimaatteja käytettiin, kuinka hyödylliseltä estimointi tuntui ja miten tyypillinen projektissa käytetty estimointitapa on haastateltavan muihin projekteihin verrattuna. Haastattelukysymysten tarkoitus oli luoda keskustelua haastateltavan estimointikokemuksista ja mielipiteistä estimointiin liittyen. Haastattelutilanteessa haastateltavat saivat myös kertoa vapaammin estimointikokemuksistaan yleisesti kaikkien projektien näkökulmasta.

Tässä haastattelututkimuksessa haastateltavat jaettiin kahteen kategoriaan: johtajaksi ja kehittäjäksi. Johtajaksi määriteltiin ohjelmistoalan ammattilainen, joka tekee ohjelmistoprojektissa päätöksiä estimaattien pohjalta. Kehittäjäksi määriteltiin ohjelmistoalan ammattilainen, joka pelkästään luo estimaatteja projektin johdolle. Johtaja on voinut toimia myös aikaisemmin kehittäjän roolissa, mutta kehittäjä ei ole toiminut johtajana. Haastateltaviksi valittiin viisi ohjelmistoalan ammattilaista sen mukaan, mitä kategoriaa he edustivat (taulukko 1). Tarkoituksena oli saada haastateltavaksi kaksi kehittäjää ja kolme johtajaa, mikä toteutui. Haastateltavat ovat olleet osallisina ohjelmistoprojekteissa, joissa on käytetty ketteriä projektinhallintamenetelmiä mutta eivät välttämättä Scrumia. Haastateltavista kahdella oli kokemusta pelkästään kehittäjänä toimimisesta ohjelmistoprojekteissa ja kolmella sekä kehittäjän että johtajan tehtävistä. Haastateltavien työkokemukset vaihtelivat vuodesta kymmeniin vuosiin. Jokainen haastateltava edusti haastattelussa eri yritystä.

Taulukko 1: Haastateltavien taustatiedot

	Titteli yrityksessä	Kokemus	Kategoria
H1	Ohjelmistokehittäjä	1 vuosi	Kehittäjä
H2	Ohjelmistokehittäjä	5 vuotta	Kehittäjä
H3	DevOps-insinööri	2 vuotta	Johtaja
H4	Operatiivinen johtaja	30 vuotta	Johtaja
H5	Prosessisuunnittelija	10 vuotta	Johtaja

5.1 Tutkimuskysymykset

Ideat tässä opinnäytetyössä esiteltyihin tutkimuskysymyksiin syntyivät keskusteluista kanssaopiskelijoiden kanssa ja kirjallisuutta selaamalla. Monet kanssaopiskelijoista ja ohjelmistokehittäjät pitävät estimointia turhana työnä. Tämä saattaa johtua siitä, että ohjelmistokehittäjät osallistuvat usein estimaattien tekemiseen, mutta he osallistuvat harvoin estimaattien perusteella tehtävien päätösten tekemiseen. TK1 pyrkii selventämään estimoinnin tarkoitusta ohjelmistokehittäjille. Ohjelmistoyrityksissä on epätietoisuutta siitä, miten estimointi kannattaa toteuttaa ohjelmistoprojekteissa. TK2 pyrkiikin selventämään käsitystä siitä, mikä on suositeltava tapa toteuttaa estimointi.

1. Onko estimointi välttämätöntä ohjelmistoprojektin onnistumiselle?
2. Onko suhteellinen estimointitapa ”paras” estimointikäytäntö (Best Practice) ohjelmistoprojekteissa?

6 Tulokset

Tähän lukuun on koottu teemahaastattelun tulokset. Tulokset on jäsennelty haastatteluissa korostettujen teemojen mukaan.

6.1 Haastateltavien kokemukset eri estimointitavoista

Haastateltavat H1, H2, ja H5 käyttivät haastattelun aikana projekteissa henkilötyöpäiviä (HTP), H3 käyttäjätarinapisteiden (KP) tapaista estimointia

ja H4 luvussa 4.3 kuvatunkaltaista ominaisuuksien lukumäärällä estimointia (OL).

Taulukko 2: Haastateltavien estimointitausta

	Nykyisin käyttämä tapa	Muut tavat
H1	HTP	
H2	HTP	
H3	KP	
H4	OL	KP FPA HTP
H5	HTP	KP

Henkilötyöpäivä (HTP), Käyttäjätarinapisteytys (KP), Ominaisuuksien lukumäärä (OL), Toiminnallisuuspisteanalyysi (FPA)

H1:n projekteissa estimoitiin projektin ominaisuuksien työvaiheita työtunteina. Ominaisuuksien työvaiheiden estimaatteja käytettiin hyväksi ominaisuuksien estimoinnissa, joiden estimointiin käytettiin henkilötyöpäiviä.

H2:lla oli kokemusta projekteista, joissa käytettiin henkilötyöpäiviä. Henkilötyöpäivillä estimoitiin projektin ominaisuuksien toteuttamiseen tarvittava työpanos. Ominaisuuksien estimointiin käytettiin asteikkoa 10-100 henkilötyöpäivää.

H3:lla oli kokemusta käyttäyttäjätarinapisteytyksestä, jossa käyttäjätarinapiste oli projektin alussa sitoutettu työn vaativuuteen. Myöhemmin projektin edetessä käyttäjätarinapiste määriteltiin yhdeksi työpäiväksi. Projektin aikana käyttäjätarinapisteiden suhteellinen asteikko vaihdettiin absoluuttiseen asteikkoon, jotta estimaateista saataisiin yhtenäisempiä tiimien välisten estimaattien kanssa. Estimaattien asteikkona käytettiin Fibonaccin lukujonon lukuja 1, 2, 3, 5, 8 ja 13.

H4 suosi haastattelun aikana menetelmää, jossa estimaatteina käytetään ominaisuuksien lukumäärää (OL). OL-menetelmän toimivuutta perusteltiin sillä, että jokainen kehitystiimi löytää pitkällä aikavälillä rytmin, jolla se saa aikaan suunnilleen vakiomäärän ominaisuuksia per iteraatio erityisesti, jos ympäristö, tiimi, teknologiat ja niiden soveltamisalueet pysyvät suunnilleen samanlaisina. Jos ympäristössä, tiimissä, teknologioissa tai niiden sovelta-

misalueissa tapahtuu merkittäviä muutoksia, niin kestää jonkin aikaa, että tiimi löytää uudelleen rytminsä. Mikäli ympäristötekijät muuttuvat paljon, käyttäjätarinapisteytys saattaa olla toimivampi menetelmä ainakin alkuun. OL-menetelmässä tiimin kunnianhimo tehdä asioita ja tiimiin kohdistuva paine pitää huolen, että tiimi pysyy riittävän suorituskykyisenä koko iteraation ajan.

Joissakin OL-mentelmän muodoissa käyttäjätarinat pyritään pakottamaan vakiokokoisiksi. Käyttäjätarinoiden pakottamista vakiokokoisiksi H4 piti vanhakantaisena eikä pitänyt sitä tarpeellisena. Kehitystiimi päättää itse siitä, kuinka monta ominaisuutta he saavat valmiiksi iteraation aikana. Kehitystiimi pyrkii pitämään lupauksen tavoiteesta jakamalla ominaisuudet riittävän pieniin ominaisuuksiin. Ominaisuuksien tulee olla kooltaan sellaisia, että useampi ominaisuus mahtuu yhteen iteraatioon. Isojen ominaisuuksien pilkkoutumista estimoitaessa estimoinnista saatujen ominaisuuksien lukumäärä tulee kertoa ns. laajennuskertoimella (Expansion Factor). Laajennuskerroin saadaan tarkastelemalla projektihistoriasta, kuinka monta prosenttia tiimin estimointivirhe on ollut keskimäärin aikaisemmin isoja ominaisuuksia pilkottaessa kehityskokoisiin ominaisuuksiin. Hyvän laajennuskertoimen tulisi olla suuruusluokkaa 1,x. Tiimin pilkkomis- tai estimointikäytänteisiin tulisi kiinnittää erityistä huomiota, mikäli laajennuskerroin on yli kaksi.

Ominaisuuksien lukumäärällä estimoiminen edellyttää vakaata projektiympäristöä. Vakaalla ympäristöllä tarkoitetaan, että projektiin osallistuvat tiimit ovat kokeneita, tiimien suorituskyky on ennakkoon tiedossa ja projektissa ei ole liikaa yllätyksiä. Kokemattomilla tiimeillä tiimien suorituskyky täytyy selvittää ja yllätykset täytyy minimoida. Suorituskyvyn selvittämiseen ja yllätyksien minimoimiseen voidaan käyttää vaihtelevanpituisia iteraatioita. Kokemattomalla tiimillä ja uusia asioita tehdessä iteraatioiden pituudet tulisi pitää lyhyinä, jotta saadaan nopeammin selville tiimin suorituskyky, ja jotta tiimin toimintaa voidaan tarkastella tarkemmin. Iteraation pituutta voidaan kasvattaa sen jälkeen, kun tiimin suorituskyvystä ja projektin luonteesta ollaan saatu tarkempi kuva ja tiimi on löytänyt hyvän rytmin saada ominaisuuksia valmiiksi. Hyvänä suorituskykynä pidetään, jos tiimi pystyy toteuttamaan iteraation aikana vähintään 70 % lupaamistaan ominaisuuksista. Mikäli tiimi ei pysy lupauksessaan, täytyy tiimin toimintaa tarkastella

retrospektiivillä.

H4:llä oli kokemusta OL-menetelmän lisäksi mm. toiminnallisuuspisteanalyysista (Function Point Analysis, FPA), jota H4 oli oppinut työpaikan kurssilla. Toiminnallisuuspisteanalyysia ei pidetty toimivana menetelmänä, koska eri tiimin tuottamat estimaatit eivät olleet vertailukelpoisia, ja lähes saman asian estimoinnissa oli eri tiimien välillä erittäin suuria eroavaisuuksia. Eroavaisuuksia ei voitu selittää tiimien kokemuksen puuttella.

H4:llä oli lisäksi kokemusta absoluuttisesta tuntimääräestimoinnista, jossa jokaiselle ominaisuudelle estimoitiin valmistumiseen tarvittava tuntimäärä. Ominaisuudet jaettiin tiimeille kehitettäväksi tiimien arvioiman kapasiteetin perusteella. Tällaisessa menetelmässä estimaatit pitivät melko hyvin paikkansa, mutta menetelmää pidettiin työläänä. Estimointiin kuluvaa aikaa jouduttiin korvaamaan tekemällä pidempiä työpäiviä.

Suhteellisesta estimointitavasta H4:llä oli kokemusta käyttötapauksien (Use Case) kanssa. Käyttötapaukset estimoitiin asteikolla, jonka arvot olivat: helppo, keskivaikea ja vaikea. Jokainen projektiin osallistunut henkilö antoi estimaatin jokaisesta käyttötapauksesta, jonka jälkeen jokainen henkilö muutti asteikon arvot työpäiviksi. Estimoinnin ja arvojen muuttamisen jälkeen pidettiin keskustelutilaisuus, joissa vertailtiin henkilöiden tekemiä estimaatteja toisiinsa. Keskustelun pohjalta tehtiin lopulliset estimaatit käyttötapauksille. Tätäkin tapaa H4 piti työläänä.

H5:n yrityksessä käytetään suurimmaksi osaksi henkilötyötunteja projektien estimointiin. Jokainen ominaisuuden työvaihe estimoidaan henkilötyötunnin tarkkuudella ja yli yhden henkilötyöpäivän työvaiheet pilkotaan pienempiin työvaiheisiin. Estimaatteja päivitetään sitä mukaan, kun saadaan lisää tietoa estimoitavasta työtehtävästä. Kehittäjien kokemus huomioitiin estimaateissa siten, että kokemattomien kehittäjien estimaattien ei oletettu olevan yhtä tarkkoja kuin kokeneiden kehittäjien.

H5:n yrityksessä oli muissa projekteissa henkilötyötuntien lisäksi käytössä käyttäjätarinapisteet. Henkilötyötunnit olivat kuitenkin huomattavasti yleisemmässä käytössä kuin käyttäjätarinapisteet. Käyttäjätarinapisteisiin verrattuna H5 piti henkilötyötunteja huomattavasti hyödyllisempänä menetelmänä. Yrityksessä työntekijät pitivät kirjaa tehdyistä työtunneista, minkä ansiosta henkilötyötunnit ovat estimoinnissa ymmärrettävämpi ja helpommin

seurattava menetelmä kuin käyttäjätarinapisteet.

H5 piti molempien estimointimenetelmien ongelmana kokemattomien kehittäjien ja uusien kehitystiimin jäsenten tekemiä estimaatteja, jotka voivat olla erittäin epätarkkoja, koska tiimin jäsenten velositeettia ei ole vielä saatu selville riittävän tarkasti. Estimointitarkkuuden uskotaan kuitenkin paranevan ajan myötä.

6.2 Suhtautuminen estimointiin

H1 pitää estimointia pääasiallisena keinona ilmaisemaan, kuinka kauan jonkin asian tekemiseen menee aikaa. H1:n suhtautuminen estimointiin ei ole ajan myötä muuttunut.

H2 jakaa estimaatit kahteen luokkaan: työpanoksen estimointiin ja valmistumisajankohdan estimointiin. Valmistumisajankohdan estimointia pidettiin huomattavasti epätarkempana ja hyödyttömämpänä kuin työpanoksen estimointia. Valmistumisajankohdan estimointia H2 pitää täysin hyödyttömänä johtuen valmistumisajankohdan estimoinnin epävarmuudesta. H2:n mielestä estimaattien käyttämistä jonkin ajankohdan ennustamiseen tulisi välttää, eikä tällaisten ennusteiden pohjalta tulisi tehdä mitään päätöksiä.

H3:n suhtautuminen estimointiin on ajan myötä muuttunut positiivisempaan suuntaan. Aiemmin haastateltava piti estimaatteja melko hyödyttömänä ajankäyttönä. Estimoinnista on kuitenkin ajan myötä tullut hyödyllinen apuväline työn jäsentämiseen.

H4 suhtautuu nykyään aikaisemmin käyttämiinsä estimointimenetelmiin (FPA, KP, HTP) turhana ajankäyttönä. Näillä menetelmillä tehdyt estimaatit eivät tuo projektille mitään lisäarvoa. Nykyään H4 pitää OL-menetelmää hyödyllisenä estimointimenetelmänä eikä suosittele projekteja täysin ilman estimointia.

H5 on suhtautunut estimointiin aina positiivisesti ja pitää estimaatteja erittäin tärkeinä ja hyödyllisinä. H5:n kokemusten perusteella pienemmissä organisaatioissa estimointi ei ole erityisen tärkeää mutta organisaation koon kasvaessa estimoinnin tärkeys korostuu, johtuen kommunikointitarpeesta usean eri henkilön välillä.

6.3 Estimoinnin hyödyllisyys

Jokaiselta haastateltavalta pyydettiin arviota estimoinnin hyödyllisyydestä asteikolla 1-5, jossa 1 on ei ollenkaan hyödyllinen ja 5 erittäin hyödyllinen. Tulokset eivät kuitenkaan ole täysin vertailukelpoisia. H1 ja H3 arvioivat yleisesti projektinaikaisen estimoinnin hyödyllisyyttä. H2 arvioi estimoinnin hyödyllisyyttä sovelluskehittäjän näkökulmasta. H4 arvioi estimointimenetelmien (SP, FPA, HTP) hyödyllisyyttä. H5 antoi arvion estimoinnin tärkeydestä omassa organisaatiossaan.

Taulukko 3: Haastateltavien arvio estimoinnin hyödyllisyydestä

	Estimoinnin hyödyllisyys 1-5
H1	3
H2	2
H3	3,5
H4	0
H5	4

H1 oli hiukan epävarma onko estimaattien luontiin käytettävä aika hyödyllistä ajankäyttöä.

H2 ei pitänyt estimaatteja sovelluskehittäjän näkökulmasta erityisen hyödyllisinä, mutta piti estimaatteja hyödyllisinä esimerkiksi tilanteissa joissa estimaatit ovat ristiriidassa projektiin varattujen resurssien kanssa.

H3:n mielestä estimoinnin hyödyllisyys riippuu projektin koosta, pienissä projekteissa estimointia pidettiin vähemmän hyödyllisenä kuin isoissa.

H4 kertoi, että ominaisuuksien estimointi FP- tai HTP-menetelmällä on vaivalloista, eikä sen kerrottu tuovan mitään lisäarvoa projektille. KP-menetelmän hyödyksi kerrottiin menetelmän kannustavan keskusteluun tiimin jäsenten kesken siitä, miksi käyttäjätarinapistearviot poikkeavat toisistaan. Käyttäjätarinapisteiden ei tarvitse olla vertailukelpoisia muiden tiimien kesken ja tällaista ajattelua H4 pitää vanhakantaisena. H4:n antama arvio näiden menetelmien hyödyllisyydestä alitti arvioinnissa käytettävän skaalan. Itse estimointia H4 piti hyödyllisenä keinona seurata projektin tilaa, ja

hyödyllisenä kommunikointimenetelmänä kehitystiimien ja yrityksen johdon tai asiakkaan välillä.

H5 piti estimointia melko tärkeänä kommunikointikeinona projektin tilasta yrityksen johdolle ja asiakkaille.

6.4 Estimoinnin tarkkuus

H1, H3, ja H5 pitivät haastattelun aikana käyttämäänsä estimointimenetelmän tuottamaa estimointitarkkuutta riittävän hyvänä. H1:n yrityksessä ei erityisesti kiinnitetty huomiota estimaattien tarkkuuteen. H2 ei pitänyt estimoitujen henkilötyötuntien perusteella tehtäviä aikaennusteita ollenkaan tarkkoina. H4 kertoi OL-menetelmän olevan vähintään yhtä tarkka kuin SP-menetelmän.

H2:n kokemuksen mukaan jatkokehitysprojektin työpanoksen ja valmistumisajan estimointia pidettiin huomattavasti haastavampana ja epätarkempana verrattuna täysin uuden projektin ominaisuuksien estimointiin. Syinä tähän pidettiin jatkoprojektin jo olemassa olevan koodipohjan muokkaamisen haastavuutta ja jatkoprojektiin liittyviä organisaatioiden välisiä riippuvuuksia, joihin kehitystiimi voi vaikuttaa huonosti. Näiden syiden takia jatkoprojekteissa on uusia projekteja huomattavasti enemmän tuntemattomia muuttujia, joihin varautuminen on haastavaa.

H3:n projekteissa projektin alussa estimointi ei ole ollut kovin tarkkaa mutta tarkentui huomattavasti projektin edetessä.

H4 piti ominaisuuksien lukumäärällä estimointia vähintään yhtä tarkkana estimointimenetelmänä kuin muitakin estimointimenetelmiä mutta huomattavasti yksinkertaisempana ja aikaa säästävämpänä johtuen muita estimointimenetelmiä pienemmästä työmäärästä.

6.5 Motivaatiot estimoinnille

H1:n esimerkkiprojektissa estimaatteja hyödynnettiin tarjouksen tekemisessä asiakkaalle. Asiakas arvioi estimaattien perusteella, onko hänellä resursseja (aikaa, rahaa) tiettyyn projektiin tai mitkä ominaisuudet asiakas pystyy saamaan tietyillä resursseilla. H1:n mukaan estimaatit ovat tarpeellisia ilmaisemaan, kuinka kauan jonkin tehtävän tekemiseen kuluu aikaa. Lisäksi

Taulukko 4: Haastateltavien motivaatio estimoinnille

	Motivaatio estimonnille
H1	Kustannusarvio, Työtehon parantaminen
H2	Kustannusarvio, Projektin ohjaus
H3	Kustannusarvio, Työn jäsennys, Aikataulun arviointi
H4	Kustannusarvio, Projektin ohjaus, Työn priorisointi, Iteraation suunnittelu
H5	Iteraation suunnittelu, Aikatauluarvio, Kustannusarvio

estimaattien kerrottiin kasvattavan työtehokkuutta luomalla painetta saada tehtäviä valmiiksi estimaatteihin mennessä. Ilman estimaattien luomaa painetta työnteko saattaisi olla mukavampaa, mutta työtehokkuus ja keskittyminen kärsisi.

H2 kertoi, että estimaatteja ei niinkään hyödynnetty projektin valmistumisajankohdan ennustamiseen vaan projektin kutannusten arviointiin asiakkaalle. H2:n esimerkkiprojektissa estimaattien toteutumista tarkasteltiin, kun noin puolet projektiin varatuista resursseista oli käytetty ja tarkastelun pohjalta projektille tehtiin ohjaavia toimenpiteitä. Estimaattien avulla huomatiin, että projektiin tarvittavat resurssit oli aliarvioitu ja pystyttiin kartoittamaan aliarviointiin johtavia syitä. Resursseja saatiin ohjattua asiakkaan muista projekteista, jolloin projektin kustannukset saatiin pidettyä asiakkaalle suotuisina.

H3:n esimerkkiprojektissa estimoitiin työn strukturoinnin takia eli estimoinnin tavoitteena oli selvittää, missä järjestyksessä eri työvaiheet on järkevää tehdä. Kehitystiimi teki estimaattien pohjalta päätöksiä siitä, mihin tulevassa iteraatiossa tulee keskittyä. Estimaatteja käytettiin tiimien työtahdin seurantaan ja projektin ominaisuuksien valmistumisaikataulun estimointiin. Estimoinnilla pyrittiin lisäksi muuttamaan ulkomaalaisten työntekijöiden työkulttuuria kohti suomalaisempaa työkulttuuria.

H4 kertoi, että estimaatteja tarvitaan projektin kustannusarvion tekemiseen ja projektin suunnan ohjaamiseen. Asiakas vaatii projekteissa kustannusarvioita, jotta asiakas voi arvioida, saako hän rahoilleen riittävästi

vastinetta. Estimaattien perusteella asiakas tekee muutoksia toteutettaviin ominaisuuksiin ja priorisoi ominaisuudet.

H5:n yrityksessä estimaatteja käytettiin apuna iteraation suunnittelussa ennustamaan kuinka paljon ominaisuuksia kehitystiimi tulee saamaan aikaan iteraation aikana. Tällä pyrittiin ehkäisemään liian suuren työmäärän varaamista iteraatiolle. Estimaatit helpottavat myös kommunikointia projektista ja yrityksestä vastaavien henkilöiden kanssa. Vastuuhenkilöt hyödyntävät estimaatteja hahmottamaan suuruusluokkia sille, miten paljon jonkin työtehtävän tekemiseen kuluu aikaa. Ilman estimaatteja aikataulujen suunnittelun kerrottiin olevan hankalaa.

6.6 Estimointiin käytetty aika

H1:n lyhyissä projekteissa (10 päivää) tehtiin vain tarjousvaiheen estimointi. Tarjousvaiheen jälkeen ei enää tehty estimointia. Lyhyissä projekteissa tarjousvaiheessa tehtävään estimointiin aikaa käytettiin noin tunti. H1:n pidempään jatkuvissa projekteissa estimoitiin vain suullisesti eikä estimaatteja kirjattu ylös.

H2:n esimerkkiprojektissa estimaattien tekoon käytettiin aikaa noin yksi henkilötyöpäivä. Projektin kokonainen koko oli noin 1000 henkilötyöpäivää.

H3:n esimerkkiprojektissa estimaattien tekemiseen käytettiin noin kaksi tuntia viikossa.

H4:n haastattelusta ei saatu esimerkkejä estimoitiin käytettävästä ajasta. Haastattelijan olisi pitänyt kiinnittää enemmän huomiota tähän haastattelun aikana.

H5:n projektissa estimaattien tekemiseen käytettiin aikaa kahden viikon pituisen iteraation aikana noin tunti.

6.7 Muita huomioita

H1:n pidemmissä projekteissa iteraation pitutta saatettiin muuttaa iteraation aikana, jotta saataisiin iteraation tavoitteet toteutumaan. Kesken jääneitä ominaisuuksia saatettiin myös siirtää seuraavaan iteraatioon. H1:n yrityksessä estimaatteja tehdään vain asiakasta varten. Yrityksen sisällä estimaatteja ei hyödynnetä.

Taulukko 5: Estimointiin käytetty aika

	Estimointiin käytetty aika
H1	1 tunti tarjousvaiheessa
H2	1 henkilötyöpäivä / 1000 HTP
H3	2 tuntia viikossa
H4	-
H5	1 tunti / kahden viikon iteraatio

H2 pitää estimaatteja tarpeellisena sen arvioimiseen, mitä asiakas voi saada käytettävissä olevilla resursseilla. Estimoinnin epävarmuus pitää kuitenkin olla asiakkaan tiedossa. Liian tarkkoja estimaatteja ja estimaatteihin suhtautumista lupauksena H2 pitää projektin kannalta vahingollisena.

H3 sitouttaa käyttäjätarinapisteen henkilötyöpäivään, vaikka luvussa 3.2.2 kerrottiin käyttäjätarinapisteiden olevan puhtaasti suhteellinen estimointitapa. H3 toivoi yhtenäisempiä estimointikäytänteitä yrityksiin, koska nykyään yritysten sisälläkin saatetaan käyttää useita eri estimointimenetelmiä.

H4 on siirtynyt uransa aikan projekteissaan vesiputousmallista ketteriin projektinhallintamenetelmiin. Vesiputousmallin kanssa käytettiin ominaisuuksien tuntiperusteista estimointia määrittelyprojektin kautta. Myöhemmin H4 siirtyi käyttämään vesiputousmallin sijasta ketteriä menetelmiä, jolloin tuntiperusteisesta estimoinnista siirryttiin estimoimaan ominaisuuden toteutuksen vaativuutta. Ketteriin menetelmiin siirryttäessä estimaatteja ei pidetty enää niin tärkeänä kuin vesiputousmallin kanssa. Estimaatteja alettiin hyödyntää valmiiden ominaisuuksien lukumäärän ennustamiseen tiettyyn aikaan mennessä koko projektin valmistumisajan ennustamisen sijasta. Estimaatteja hyödynnettiin tuotteen kehitysjonon priorisoinnissa pitämällä tärkeimpiä ominaisuuksia kehitysjonon kärjessä ja karsimalla vähemmän tärkeitä ominaisuuksia pois, mikäli tiimien kapasiteetti ei riitä ominaisuuksien toteuttamiseen. H4 piti Ketterää projektinhallintamenetelmää ja sen kanssa käytettäviä estimointimenetelmiä ylivoimaisena vesiputousmalliin verrattuna. Projektien kaikki osapuolet olivat siirtymään tyytyväisiä, ja projekteista

saatiin tiputettua määrittelyvaihe pois, mitä H4 piti pääosin epäoleellisena työnä.

H5:n mielestä estimaatit helpottavat tulevaisuuden suunnittelua, koska estimaattien avulla pystytään arvioimaan, kuinka kauan jonkin työtehtävän tekemiseen kuluu suunnilleen aikaa. H5 kertoi, että ohjelmistokehittäjille saatetaan joutua perustelamaan, miksi heidän tulee tehdä estimaatteja.

6.8 Validiteetti

Haastattelututkimuksen tuloksia pyrittiin soveltamaan kirjallisuuteen tarkoituksena löytää vahvistuksia tai kritiikkiä kirjallisuudessa esitettyihin väittämiin kuitenkaan unohtamatta uusia näkökulmia. Kaikki haastateltavat olivat suomalaisia, joten tutkimuksen tulokset eivät välttämättä ole sovellettavissa muiden kansallisuuksien edustajiin. Kaikki haastattelut suoritettiin puolen vuoden aikana, joten ajalla ei katsottu olevan vaikutusta tutkimustulosten eroavaisuuteen. Tutkimustuloksina ovat yksittäisten ihmisten kokemukset ja mielipiteet, joten tutkimuksen tulokset eivät ole välttämättä yleistettävissä kaikkiin ihmisiin tai tilanteisiin. Haastattelututkimuksesta kirjoitetut tulokset lähetettiin haastateltaville, jolloin haastateltavilla oli mahdollisuus ottaa kantaa tulosten oikeellisuuteen ja antaa parannusehdotuksia, mikäli jokin kohta tulosten raportoinnissa ei vastannut haastateltavan mielipiteitä tai kokemuksia. Haastateltavilta saatiin muutama parannusehdotus tulosten raportointiin liittyen, mitkä toteutettiin työn lopulliseen versioon.

7 Pohdinta

Haastattelututkimuksesta kerätyn tiedon soveltaminen tutkimuskysymyksiin osoittautui erittäin haasteelliseksi. Pelkän tutkielmaa varten tehdyn haastattelututkimuksesta saadun tiedon perusteella ei pystytä saamaan tarkkoja vastauksia tutkimuskysymyksiin, johtuen tutkimuksen pienestä otannasta, joka ei sopinut erityisen hyvin tässä tutkimuksessa muotoiltuihin tutkimuskysymyksiin. Suurimmalla osalla haastateltavista oli kokemusta vain yhden tai kahden estimointimenetelmän käytöstä, joten haastateltavan haastattelun aikana käytössä oleva estimointimenetelmä ei välttämättä kuvaa haastateltavan näkemystä "parhaasta"estimointikäytänteestä. Tutkimuksen tulokset

eivät ole erityisen vertailukelpoisia keskenään johtuen osittain haastatteluissa käydyn keskustelun hajonnasta. Haastattelijan kokemattomuudella haastattelutilanteista oli suuri vaikutus haastatteluista saatujen tulosten johdonmukaisuuteen eikä haastateltavien objektiivisuutta pystytty varmistamaan millään tavalla.

Estimoinnin hyödyllisyydestä kysyttäessä haastattelija olisi voinut määritellä hyödyllisyyden selkeämmin haastateltaville. Haastateltavat antoivat aiheesta hyvin erilaisia vastauksia riippuen siitä, miten kukin haastateltava ymmärsi aiheen. Estimoinnin hyödyllisyyttä arvioitaessa kehittäjät olisivat voineet esimerkiksi arvoida miten hyödylliseltä estimaatien tekemiseen käytettävä työ on tuntunut, kun taas johtajat olisivat voineet arvioida kuinka hyödyllisiä estimaatit ovat olleet päätöksenteossa. Hyödyllisyysarvioita olisi voinut antaa lisäksi eri estimointitapojen näkökulmasta.

Estimaattien toteutuminen ei kuulunut haastettelututkimuksen rakenteeseen mutta keskustelua käydessä estimaattien toteutuminen otettiin puheeksi osan haastateltavien kanssa. Estimaattien toteutumisesta ei saatu haastateltavilta tarkkaa vastausta tai esimerkkejä. Haastateltavien yrityksissä ei välttämättä pidetty kirjaa estimaattien toteutumisesta tai haastateltavilla ei ollut siitä tietoa. Tieto estimointien toteutumisesta saattoi olla myös salassapitovelvollisuuden alaista tietoa.

Kaikilta haastateltavilta saatiin arvokasta tietoa ohjelmistoprojektien estimointikäytänteistä. Tutkimuksessa korostettiin H4:n kokemuksia ja mielipiteitä. Tämän tutkimuksen haastateltavista H4 oli kokenein eri estimointimenetelmien käytöstä ja omasi vahvan näkemyksen käytännöllisimmästä estimointimenetelmästä. H4:n vastauksista saatiin lisätietoa NoEstimates-keskustelussa esitetystä OL-menetelmästä. Mainitsemisen arvoinen haastateltava oli myös H2, jolla oli haastattelun aikana vahva näkemys ohjelmistoprojektin estimointiin liittyvistä ongelmista kehittäjän näkökulmasta.

Suhteellisen estimointimenetelmän käytön vähyyteen ei löydetty selkeää syytä. Absoluuttiset menetelmät saattavat olla asiakkaan kanssa kommunikoidessa helpommin ymmärrettävä ilmaisutapa kuin suhteellinen menetelmä. Asiakkaalle kustannusten arviointi ja priorisointi saattaa olla yksinkertaisempaa tuntien tai henkilötyöpäivien perusteella kuin esim. käyttäjätarinapisteiden. Kehittäjien ja johtohenkilöstön tavat ja asenteet saattavat myös

vaikuttaa suhteellisen menetelmän käytön vähyyteen. Tämä kaipaisi vielä lisätutkimusta.

Luvussa 4 kerrottiin estimoinnin tarpeellisuudesta ohjelmistoprojektien virheiden vähentämiseksi. Haastatteluissa ei kuitenkaan käynyt ilmi, miten estimointi on ollut avuksi virheiden seulonnassa, jos on ollut ollenkaan. Haastattelijan osalta tähän olisi voinut kiinnittää enemmän huomiota. Tämä kaipaisi lisätutkimusta.

NoEstimates-keskusteluissa esitetyt väittämät ja tulokset eivät välttämättä ole vertaisarvioituja, eikä näin ollen niiden validiteetista voi olla täysin varma. NoEstimates-keskusteluun osallistujilla saattaa olla valmiiksi ennakkoasenteita ja kritiikittömyyttä omia ajatuksiaan kohtaan, mistä johtuen keskusteluihin tulee suhtautua varauksella. Haastattelututkimuksen eräänä tavoitteena oli löytää NoEstimates-keskusteluissa esitetystä OL-menetelmästä käyttökokemuksia ohjelmistoalan ammattilaisilta, joilla saataisiin lisätietoa ja tukea tai kritiikkiä internetkeskusteluissa esitetyille väittämille.

Asiakkaan näkökulman ottoa mukaan tutkimukseen harkittiin tutkielmaa tehdessä, mutta sitä ei kuitenkaan nähty tarpeellisena. Henkiökohtaisen kokemuksen perusteella asiakkaalla ei ole merkittävää vaikutusta estimoinnin toteutukseen ohjelmistoprojekteissa. Jatkotutkimuksissa voitaisiin mahdollisesti huomioida asiakkaan näkökulma estimointiin.

8 Johtopäätökset

Jokaisen haastateltavan kokemusten perusteella estimointi on jossain muodossa välttämätön osa ohjelmistoprojektia, mikäli projektissa on mukana asiakas. Pienissä asiakasprojekteissa ei mahdollisesti saada lisähyötyä tarjousvaiheen jälkeen tehtävästä projektinaikaissta estimoinnista. Tämä kuitenkin vaatii lisätutkimusta. Yrityksen sisäisien työkalujen kehittämiseen tarkoitetuissa projekteissa estimointi ei mahdollisesti ole välttämätöntä. Yrityksen sisäisistä projekteista puuttuu asiakas, jonka kanssa kommunikoimiseen estimointia pidetään välttämättömänä apuvälineenä.

Tutkimuksen tulosten perusteella tärkein yksittäinen motivaatio estimoinnin tekemiseksi on projektin kustannusten raportointi asiakkaalle ja yrityksen johdolle. Jokainen haastateltava mainitsi kustannusten arvioinnin erääksi

motivaatioksi estimoinnille. Tämän tutkimuksen perusteella estimointi on siis välttämätön työkalu kustannusten arvioimiseen.

Haastatteluissa kävi ilmi, että estimointia käytetään työmotivaation kohottamiseen. Työskentely saattaa olla tehokkaampaa, kun estimaatit luovat tavoitteen, jota kohti pyrkiä. Kirjallisuudessa esitetyn väitteen mukaan estimaatit saattavat kuitenkin joissain tapauksissa hidastaa tehtävien valmistumista, mikäli ohjelmistokehittäjä pyrkii käyttämään aina kaiken työtehtävälle estimoidun ajan. Tämä vaatii lisätutkimusta.

Aikaan perustuvat (HTP) estimointimenetelmät olivat tässä opinnäytetyössä tehdyn haastattelututkimuksen perusteella (taulukko 2) yleisimmin käytössä oleva estimointitapa yrityksissä, vaikka suhteellista estimointitapaa pidetään kirjallisuudessa tämän hetken ”parhaana” estimointikäytänteenä. Estimointitavoista tutkimuksessa nousi kuitenkin esille ominaisuuksien lukumäärällä (OL) estimointi, joka on opinnäytetyön kirjoituksen aikaan NoEstimates-liikkeen yleisenä keskustelun aiheena. Tutkimuksessa korostui H4:n kokemus ohjelmistokehitysalalta ja monista eri estimointimenetelmistä sekä vahva näkemys OL-menetelmän ylivertaisuudesta muihin estimointimenetelmiin nähden. OL-menetelmää pidettiin H4:n kertomusten ja NoEstimates-kirjoitusten perusteella yksinkertaisempana ja helpommin ymmärrettävämpänä kuin muita menetelmiä ja vähintään yhtä tarkkana. NoEstimates-keskusteluissa OL-menetelmä on saanut paljon huomiota, joten menetelmään olisi syytä kohdistaa lisätutkimusta.

Lähteet

Agile Alliance. (2019). *What is Agile?* Lainattu 5.5.2019, saatavilla `https://www.agilealliance.org/agile101/`

Agile Finland. (2001). *Ketterän ohjelmistokehityksen julistus.* Lainattu 5.5.2019, saatavilla `http://agilemanifesto.org/iso/fi/principles.html`

Arlen. (2019). *The throw down: Agile estimation vs. #noestimates.* Lainattu 19.7.2019, saatavilla `https://lithespeed.com/throw-agile-estimation-vs-noestimates/`

Beck, K. (2004). *Extreme programming explained.* Addison-Wesley Educational Publishers Inc.

Borade, J. G., & Khalkar, V. R. (2013). Software project effort and cost estimation techniques. Teoksessa *International journal of advanced research in computer science and software engineering* (osa 3). IJARCSSE.

Buglione, L., & Abran, A. (2007). Improving estimations in Agile projects: Issues and avenues. Teoksessa *Proceedings software measurement european forum (SMEF) 2007.* SMEF.

CA Technologies. (2017). The impact of Agile. quantified.. Lainattu 11.12.2019, saatavilla `https://docs.broadcom.com/docs/the-impact-of-agile-quantified`

Cao, L. (2008). Estimating Agile software project effort: An empirical study. Teoksessa *Americas conference on information systems.*

Cohn, M. (2004). *User Stories applied for Agile software development.* Addison-Wesley.

Cohn, M. (2006). *Agile estimating and planning.* Prentice Hall.

Cohn, M. (2016). *What are Story Points?* Lainattu 4.6.2019, saatavilla `https://www.mountaingoatsoftware.com/blog/what-are-story-points`

Colucci, L. (2016). *Forecasting software project's completion date through Monte Carlo Simulation.* Lainattu 17.3.2019, saatavilla http://blog.plataformatec.com.br/2016/08/forecasting-software-projects-completion-date-through-monte-carlo-simulation/

ConfEngine. (2015). *No estimates by Vasco Duarte.* Lainattu 14.10.2019, saatavilla https://www.youtube.com/watch?v=cgvB2wWvi8M

Conte, S. D., Dunsmore, H. E., & Shen, V. Y. (1986). *Software engineering metrics and models.* Benjamin-Cummings Publishing Co., Inc.

Dalcher, D. (2014). *Rethinking success in software projects: Looking beyond the failure factors.* Springer-Verlag Berlin Heidelberg.

Duarte, V. (2012). *Story Points considered harmful - or why the future of estimation is really in our past...* Lainattu 27.02.2017, saatavilla http://softwaredevelopmenttoday.blogspot.fi/2012/01/story-points-considered-harmful-or-why.html

Duarte, V. (2016). *Noestimates: How to measure project progress without estimating.* OikosofySeries.

Duarte, V. (2017). *Vasco Duarte: Noestimates.* Lainattu 21.5.2019, saatavilla https://www.youtube.com/watch?v=MhbT7EvYN0c

Eveleens, J., & Verhoef, C. (2010, 01). The rise and fall of the Chaos report figures. *IEEE Software*, *27*(1), 30-36. doi: 10.1109/MS.2009.154

Fageha, M., & Aibinu, A. (2013, 03). Managing project scope definition to improve stakeholders' participation and enhance project outcome. Teoksessa *Procedia - social and behavioral sciences* (osa 74, s. 154–164).

Fitsilis, P., Damasiotis, V., & F. O'Kane, J. (2015, 09). Scope management complexity in software projects. Teoksessa *British academy of management - BAM2014 conference, at Belfast, North Ireland.*

Haugen, N. C. (2006). An empirical study of using Planning Poker for User Story estimation. Teoksessa *Agile 2006 (Agile'06).* IEEE.

Heusser, M. (2013). *'No estimates' in action: 5 ways to rethink software projects.* Lainattu 11.9.2019, saatavilla https://www.cio.com/article/2381167/no-estimates-in-action-5-ways-to-rethink-software-projects.html

Hirsijärvi, S., & Hurme, H. (2000). *Tutkimushaastattelu: Teemahaastattelun teoria ja käytäntö.* Gaudeamus.

Hofstadter, D. (1979). *Gödel, escher, bach: An eternal golden braid.* Basic Books.

Jeffries, R. E. (2004). *A metric leading to agility.* Lainattu 2.4.2019, saatavilla https://ronjeffries.com/xprog/articles/jatrtsmetric/

Jeffries, R. E. (2013). *The noestimates movement.* Lainattu 11.10.2018, saatavilla https://ronjeffries.com/xprog/articles/the-noestimates-movement/

Johnson, J. (2018). *Chaos report: Decision latency theory: It is all about the interval.* The Standish Group.

Jørgensen, M. (2002). A review of studies on expert estimation of software development effort. Teoksessa *The journal of systems and software* (osa 70). Elsevier.

Karhatsu, H. (2015). *#NoEstimates - Alternative to estimate-driven software development.* Lainattu 20.12.2018, saatavilla http://www.methodsandtools.com/archive/noestimates.php

Keaveney, S., & Conboy, K. (2006). Cost estimation in Agile development projects. Teoksessa *Proceedings of the 14th european conference on information systems.* ECIS.

Mahnič, V. (2011). A case study on Agile estimating and planning using Scrum. Teoksessa *Electronics and electrical engineering* (osa 5).

Mahnič, V., & Hovelja, T. (2012). On using Planning Poker for estimating User Stories. Teoksessa *The journal of systems and software* (osa 85). Elsevier.

McConnell, S. (2006). *Software estimation: Demystifying the black art.* Microsoft Press.

Merriam-Webster, I. (2019). *English language learners definition of estimate.* Lainattu 27.9.2019, saatavilla `https://www.merriam-webster.com/dictionary/estimate`

Miranda, E., Bourque, P., & Abran, A. (2009). Sizing User Stories using paired comparisons. Teoksessa *Information and software technology* (osa 51).

Moskalenko, S. (2017). *What Scrum says about estimates.* Lainattu 27.11.2018, saatavilla `https://www.scrum.org/resources/blog/what-scrum-says-about-estimates`

Overeem, B. (2015). *The #noestimates movement.* Lainattu 16.9.2019, saatavilla `https://www.barryovereem.com/the-noestimates-movement/`

Parkinson, C. N. (1955). Parkinson's law. Teoksessa *The Economist* (osa November).

Popli, R., & Chauhan, N. (2013). A Sprint-point based estimation technique in Scrum. Teoksessa *2013 international conference on information systems and computer networks.* IEEE.

Popli, R., & Chauhan, N. (2014). Cost and effort estimation in Agile software development. Teoksessa *2014 international conference on reliability, optimization and information technology.*

Poppendieck, M. (2003). *Lean software development.* Addison-Wesley Educational Publishers Inc.

Rainsberger, J. B. (2013). *7 minutes, 26 seconds, and the fundamental theorem of Agile software development.* Lainattu 5.11.2018, saatavilla `https://www.youtube.com/watch?v=WSes_PexXcA`

Roach, T. (2011). *What does the Planning Onion mean to you?* Lainattu 8.9.2019, saatavilla `https://lithespeed.com/throw-agile-estimation-vs-noestimates/`

ScrumGuides.org. (2018). *The Scrum guide.* Lainattu 27.11.2018, saatavilla https://scrumguides.org/scrum-guide.html

The Standish Group. (1995). *The Standish Group report Chaos.* Lainattu 27.10.2018, saatavilla https://www.utdallas.edu/~chung/SYSM6309/chaos_report.pdf

Usman, M., Britto, R., Damm, L.-O., & Börstler, J. (2018). Effort estimation in large-scale software development:an industrial case study. Teoksessa *Information and software technology.*

Verner, J., Sampson, J., & Cerpa, N. (2008). What factors lead to software project failure? Teoksessa *Proceedings of the 2nd international conference on research challenges in information science.*

Version One Inc. (2019). *The 13th state of Agile report.* Lainattu 2.2.2019, saatavilla https://www.stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-report/473508

Verwijs, C. (2014). *What is this thing called (business) value?* Lainattu 4.6.2019, saatavilla https://medium.com/the-liberators/what-is-this-thing-called-business-value-3b88b734d5a9

Wake, B. (2003). *INVEST in good stories, and SMART tasks.* Lainattu 21.5.2019, saatavilla https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/

Wake, B. (2005). *Twenty ways to split stories.* Lainattu 19.7.2019, saatavilla https://xp123.com/articles/twenty-ways-to-split-stories/

Liitteet

A Haastattelurakenne

Esitiedot

Onko työskennellyt ketterissä projekteissa?
Onko ollut useita rooleja?

Valitse jokin tietty projekti
- Estimoitu, viimeaikainen, oma

Mikä rooli haastateltavalla on projektissa?

Pääkysymykset (valittu projekti)
Miten on estimoitu?
Mikä oli motivaatio estimoinnille?
Paljon aikaa estimointiin on käytetty?
Mihin estimaatteja on käytetty?
Tuntuiko estimointi tarpeelliselta?

Yleiset
Onko muissa projekteissa estimoitu ollenkaan,
kuinka tyypillinen valittu projekti,
miten estimointia on ajateltu jatkossa

Miten tärkeänä pidät estimointia (asteikko 1-5)?
- Projektin aikataulu/hallinta, rahalliset
- Asiakkaan näkökulmasta

Onko jotain mitä ei käsitelty, mutta haluaisit lisätä (estimointiin liittyen)?
Jos tulee jotain kysyttävää, niin voinko olla yhteydessä?