Jussi-Pekka Rytkönen

Ketterien menetelmien empiiriset hyödyt ja haasteet: systemaattinen kirjallisuuskatsaus

Tietotekniikan pro gradu -tutkielma

6. toukokuuta 2024

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Jussi-Pekka Rytkönen

Yhteystiedot: jussi.p.rytkonen@student.jyu.fi

Ohjaaja: Tommi Mikkonen

Työn nimi: Ketterien menetelmien empiiriset hyödyt ja haasteet: systemaattinen kirjallisuuskatsaus

Title in English: Empirical benefits and challenges of agile methodologies: a systematic literature review

Työ: Pro gradu -tutkielma

Opintosuunta: Ohjelmisto- ja tietoliikennetekniikan opintosuunta

Sivumäärä: 51+0

Tiivistelmä: Ketterät menetelmät ovat yleisesti ohjelmistokehityksessä käytettyjä projektinhallintaan liittyviä menetelmiä. Niiden käyttäminen on tänä päivänä todella yleistä ja niihin liittyy paljon etenkin positiivisia olettamuksia. Niitä käyttämällä esimerkiksi saavutetaan parempia tuloksia ja halvemmat kustannukset. Menetelmien yleisyyden ja niihin liittyvien olettamuksien vuoksi on tärkeää tutkia, mitä empiirisiä tutkimustuloksia menetelmistä oikeasti on olemassa.

Tutkielmassa toteuttiin systemaattinen kirjallisuuskatsaus, jonka tavoitteena oli selvittää, mitä empiirisesti tutkittuja hyötyjä tai haasteita ketteriin menetelmiin liittyy. Katsauksen tulosten perusteella voidaan todeta, että ketteriin menetelmiin liittyy paljon erilaisia hyötyjä ja haasteita. Katsauksessa löydettiin seitsemän eri hyötyä ja seitsemän eri haastetta. Selkeästi yleisin katsauksen artikkeleissa esiintynyt havainto oli ohjelmakoodin laadun paraneminen ketterien menetelmien ansiosta.

Avainsanat: ketterät menetelmät, ohjelmistokehitys, pro gradu -tutkielmat, systemaattiset kirjallisuuskatsaukset

Abstract: Agile methodologies are project management methodologies and they are commonly used in software development. Agile methodologies are currently very popular and

there are many especially positive assumptions associated with them. By using them you are supposed to get better results and lower the costs, for example. Because of the popularity of these methodologies and the assumptions associated with them, it is important to research if empirical data about agile methodologies actually exists.

In this study a systematic literature review was conducted in order find out if there are empirical data about benefits or challenges of agile methodologies. Based on the results of the review, it can be stated that there are many different benefits and challenges associated with agile methodologies. In the review seven different benefits and seven different challenges were found. The most common observation found in the articles of the review, was that the quality of the code increases by using agile methodologies.

Keywords: agile methodologies, software development, master's theses, systematic literature reviews

Kuviot

Taulukot

Sisällys

1 Johdanto

Ketterät menetelmät ovat erilaisia projektinhallintaan liittyviä menetelmiä, joita käytetään yleisesti ohjelmistokehitysprojekteissa. Ketterien menetelmien käyttäminen on nykyään todella yleistä. Esimerkiksi vuoden 2018 ketteryys kyselyyn vastanneiden organisaatioista 97% käytti ketteriä menetelmiä jollain organisaation osa-alueella (Hoda, Salleh ja Grundy 2018).

Ketterien menetelmien käyttöön liittyy paljon positiivisia olettamuksia. Esimerkiksi Highsmith ja Cockburn (2001) mainitsevat, että ketterät menetelmät mahdollistavat nopeat suunnanmuutokset projekteissa, jonka ansiosta saavutetaan parempia tuloksia ja halvemmat kustannukset. Ketterien menetelmien yleisyyden ja positiivisten olettamusten vuoksi menetelmiä on tärkeää tutkia tarkemmin.

Etenkin ketteriin menetelmiin liittyvien positiivisten olettamusten vuoksi on selvitettävä, onko olemassa empiirisiä tutkimustuloksia olettamusten paikkansapitävyyden vahvistamiseksi. Tässä tutkielmassa toteuttiin systemaattinen kirjallisuuskatsaus, jonka avulla pyrittiin selvittämään, mitä empiirisesti tutkittuja hyötyjä tai haasteita ketteriin menetelmiin liittyy. Katsauksen tutkimustulosten perusteella voidaan arvioida, pitävätkö ketteriin menetelmiin liittyvät olettamukset paikkaansa.

Tutkielman luvussa 2 käydään läpi tutkielman kannalta olennainen teoria eli ketterät menetelmät yleisesti sekä katsauksessa esiintyneet ketterät menetelmät yksitellen. Luvussa 3 esitellään tutkimusasetelma ja tutkielman tiedonkeruuprosessi. Luvussa 4 esitellään katsauksen tutkimustulokset. Luvussa 5 vastataan tutkimuskysymyksiin, pohditaan tutkimustulosten merkittävyyttä, arvioidaan tutkimustulosten luotettavuutta sekä esitetään mahdollisia jatkotutkimusaiheita. Tutkielma päättyy yhteenvetolukuun 6.

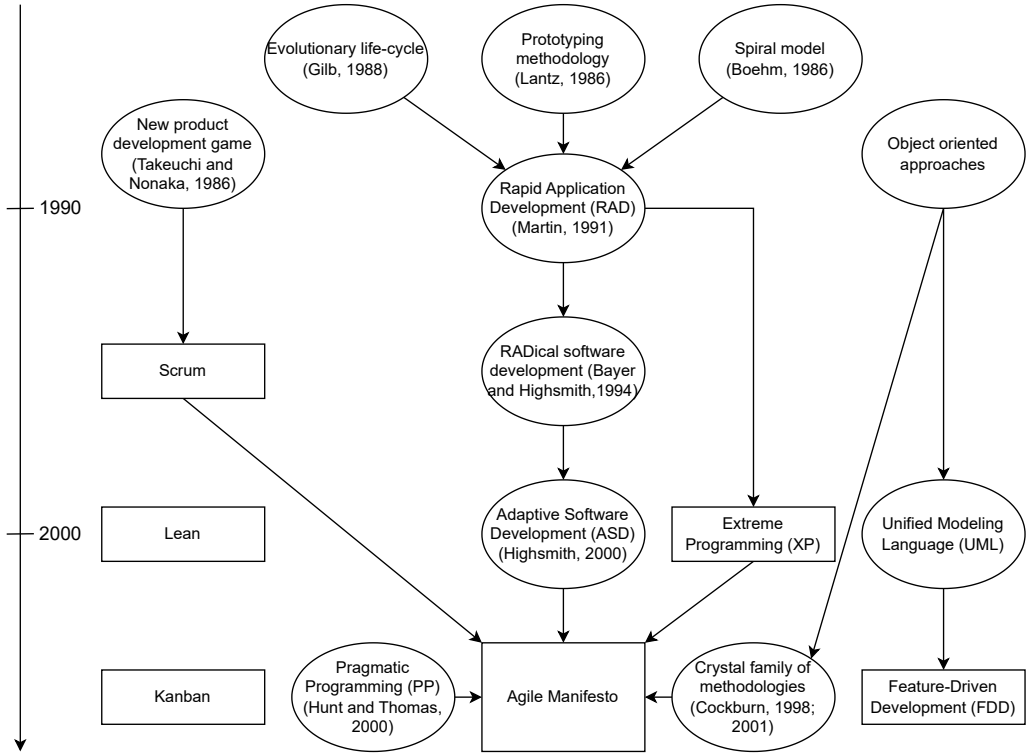
2 Ketterät menetelmät

Tässä luvussa käydään läpi tutkimuksen kannalta olennainen teoria. Aliluvussa 2.1 käsitellään ketteriä menetelmiä yleisellä tasolla kuten niiden historiaa ja perusperiaatteita. Aliluvuissa 2.2–2.6 esitellään tutkimuksessa esiintyvät ketterät menetelmät yksitellen. Jokaisessa aliluvussa esitetään muun muassa ketterän menetelmän yleisyys, käytännöt ja työnkulku.

2.1 Johdatus ketteriin menetelmiin

Williams ja Cockburn (2003) artikkelissa mainitaan että ennen ketterien menetelmien käyttöä ohjelmistokehityksessä käytettiin yleisesti suunnitteluvetoisia metodeja, joka tarkoitti sitä että kaikki projektin vaatimukset tuli olla täydellisesti dokumentoituna ennen kuin kehitystyö voitiin aloittaa. 90-luvun puolivälissä täydellisen alkudokumentaation luomiseen alettiin kuitenkin turhautumaan ja sellaisen luominen koettiin jopa mahdottomaksi. Teknologia- ja liiketoimintaympäristöt muuttuivat jatkuvasti projektien aikana, jonka vuoksi projektien suunnitelmat ja vaatimukset vanhenivat jopa lyhyiden projektien aikana. Williams ja Cockburn (2003) mainitsevat, että asiakkaat eivät myöskään aina kyenneet kertomaan kaikkia tarpeitaan etukäteen, joka vaikeutti projektin vaatimusten selvittämistä.

Jatkuvien muutosten ja epäselvien vaatimuksien vuoksi ammatinharjoittajat alkoivat kehittämään uusia metodeja ja käytäntöjä, jotka sallivat ja omaksuivat jatkuvat muutokset niiden välttelemisen sijaan. Erilaisia metodeja kehitettiin yhteensä kolmessa eri maanosassa: Euroopassa, Australiassa sekä Yhdysvalloissa. Vaikka näiden metodien käytänteet ja filosofiat olivat perustavanlaatuisesti samankaltaisia, jokainen niistä oli laadittu itsenäisesti tietämättä toisista metodeista. Williams ja Cockburn (2003) artikkelissa mainitaan kuinka helmikuussa vuonna 2001 seitsemäntoista näistä ammatinharjoittajista tapasi toisensa Utahissa keskustellakseen heidän metodiensa samankaltaisuuksista. He alkoivat kutsumaan metodejaan "ketteriksi"(engl. *agile*) ja kirjoittivat ketterän ohjelmistokehityksen julistuksen (engl. *manifesto for agile software development*). Kuviossa 1 on esitetty ketterien menetelmien evoluutio ja julistukseen johtaneet menetelmät. Kuvio 1 on toteutettu Uludağ ym. (2021) artikkelin vastaavaa kuviota mukaillen. Kuvion 1 suorakulmaisia olioita käsitellään tässä tutkielmassa.



Kuvio 1. Ketterien menetelmien evoluutio.

Ketterän ohjelmistokehityksen julistuksessa kuvattiin neljä ketterien menetelmien perusperiaatetta:

- yksilöt ja vuorovaikutukset ovat prosesseja ja työkaluja tärkeämpiä,
- toimiva ohjelmisto on tärkeämpi kuin kattava dokumentaatio,
- yhteistyö asiakkaan kanssa on sopimusneuvotteluja tärkempää ja
- muutoksiin reagointi on tärkeämpää kuin suunnitelman noudattaminen.

Highsmith ja Cockburn (2001) mainitsevat, että ketterissä menetelmissä toimiva koodi ja ihmisten tehokas sekä hyväntahtoinen työskentely keskenään ovat keskiössä. Toimivasta koodista kehittäjät ja asiakkaat aidosti näkevät mitä heillä on käsissään, sen sijaan että kerrottaisiin lupauksia tulevasta ohjelmasta. Ihmisten tehokas työskentely parantaa ohjattavuutta, nopeutta sekä vähentää kustannuksia. Kasvokkain käytävät keskustelut mahdollistavat ideoiden nopean vaihtamisen, ongelmien ratkaisemisen, prioriteettien muokkaamisen ja vaihtoehtoisten polkujen tarkastelun, jonka vuoksi on tärkeää että kehittäjät keskustelevat ja työsken-

televät yhdessä asiakkaiden kanssa.

Jotta tiimi voi olla ketterä, sen on saatava säännöllisesti palautetta asiakkailta ja johdolta. Highsmith ja Cockburn (2001) toteavat, että ketterissä menetelmissä suositaan lyhyitä, yleensä kahden-kuuden viikon mittaisia, iteraatioita, jonka aikana tiimi tekee jatkuvasti valintoja ja sopeutuu uuden tiedon mukaisesti. Iteraation lopuksi asiakas voi uudelleen priorisoida toteutettavia toimintoja, hylätä suunniteltuja toimintoja tai lisätä uusia toimintoja seuraavaa iteraatiota varten. Säännöllinen palaute teknisistä päätöksistä, asiakkaan vaatimuksista ja johdon rajoitteista on olennainen osa iteraatiota. Joissain ketterissä menetelmissä palautetta kerätään iteraation aikana, esimerkiksi pariohjelmoinnin avulla, kun taas toisissa menetelmissä palaute kerätään iteraation lopuksi esimerkiksi iteraation läpikäynnin avulla.

Nykyään on olemassa useita erilaisia ketteriä menetelmiä, kuten esimerkiksi Scrum, Extreme Programming ja Kanban, ja ne ovat yleisesti käytössä ohjelmistokehityksen lisäksi myös muilla aloilla. Hoda, Salleh ja Grundy (2018) artikkelissa mainitussa vuoden 2018 ketteryys kyselyssä vastanneiden organisaatioista 97% käytti ketteriä menetelmiä jollain organisaation osa-alueella, kun vuoden 2007 vastaavassa kyselyssä luku oli 84%. Vuoden 2018 kyselyn mukaan 84% organisaatioista ei ole vielä täysin omaksunut ketteriä menetelmiä, joka korostaa jatkuvia mahdollisuuksia tutkia ketterien menetelmien käyttöönottoon ja käytäntöihin liittyviä haasteita.

Haasteista huolimatta ketteriin menetelmiin liittyy myös paljon positiivisia olettamuksia. Hoda, Salleh ja Grundy (2018) mainitsevat, että ketterät menetelmät esimerkiksi auttavat ohjelmistoinsinöörejä rakentamaan, käyttöönottamaan ja ylläpitämään kompleksisia järjestelmiä. Highsmith ja Cockburn (2001) toteavat, että muutoksista aiheutuvat kustannukset vähenevät projektin aikana ketterien menetelmien ansiosta. Artikkelissa mainitaan myös että ketterissä menetelmissä keskitytään ensisijaisesti laatuun ja jokainen ketterä menetelmä käsittelee laatua omalla tavallaan. Esimerkiksi Scrumissa pidetään päivittäisiä viidentoista minuutin palavereita ja kattavia iteraatioiden läpikäyntejä iteraatioiden päätteeksi. Ketterässä ohjelmistokehityksessä luova tiimityöskentely sekä intensiivinen keskittyminen tehokkuuteen ja ohjautuvuuteen yhdistyvät.

2.2 Scrum

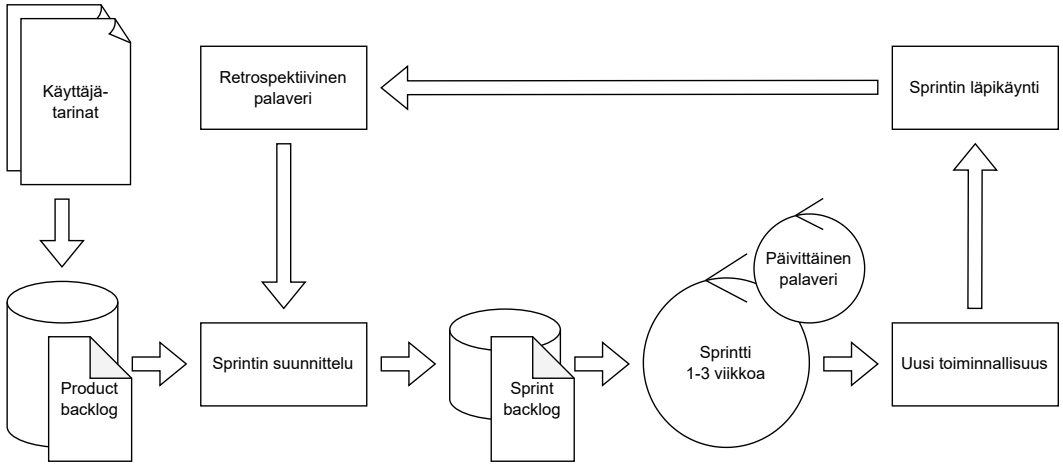
Scrum on yksi yleisimmistä ketteristä menetelmistä. Hoda, Salleh ja Grundy (2018) artikkelissa mainitussa vuoden 2018 ketteryys kyselyssä vastanneiden organisaatioista 56% käytti vain Scrumia, 8% käytti Scrumin ja Kanbanin yhdistelmää ja 6% käytti Scrumin ja Extreme Programmingin yhdistelmää. Vastanneiden organisaatioista siis yhteensä 70% käytti Scrumia jollain tavalla. Srivastava, Bhardwaj ja Saraswat (2017) mainitsevat, että Scrum on suunniteltu nopeuttamaan kehitystyötä, yhdistämään yksilöiden ja organisaatioiden periaatteita, luomaan suorituskeskeistä kulttuuria, luomaan lisäarvoa osakkaille, tukemaan tehokasta kommunikointia sekä parantamaan yksilön elämän laatua ja kehittymistä.

Srivastava, Bhardwaj ja Saraswat (2017) kuvaavat artikkelissaan Scrumin työskentelyä ja työnkulkua. Scrumissa Scrum tiimi, Scrum Master ja tuoteomistaja (engl. *product owner*) tekevät tiivistä yhteistyötä keskenään jatkuvien sovelluskehitys iteraatioiden aikana. Scrum tiimit ovat kokonaisvaltaisia ja ne koostuvat kehittäjistä, testaajista ja mahdollisesti muista eri alan ammattilaisista, joiden osaamista tarvitaan kehitystyön tukena. Scrum Masterin tärkein tehtävä on hankkiutua eroon mahdollisista työntekoa hidastavista esteistä ja tarvittaessa opastaa Scrumissa.

Scrumissa työnkulku etenee seuraavasti: sprintti on Scrumin työnkulun pienin osa, jonka aikana tiimi työskentelee määrättyjen tehtävien parissa yhdestä kolmeen viikkoon. Sprintin jokaisena päivänä pidetään lyhyt päivittäinen palaveri (engl. *daily*), jossa käydään läpi kuinka päivän tehtävissä on edistytty. Jokaisen sprintin tavoitteena on tuottaa tuote, joka olisi tarvittaessa toimitettavissa. Sprintin aikana toteutettavat tehtävät ovat sprintin tehtävälistalla (engl. *sprint backlog*), joka on dokumentaatio kuluvan sprintin kaikista tehtävistä. Sprintin tehtävälistan tehtävät määräytyvät tuotetehtävälistan (engl. *product backlog*) perusteella, joka sisältää tuoteomistajan määrittämät vaatimukset eli käyttäjätarinat (engl. *user story*).

Ennen sprintin alkamista järjestetään sprintin suunnittelu, jossa sovitaan tulevan sprintin tavoitteet ja pilkotaan tuotetehtävälistan käyttäjätarinat tehtäviksi sprintin tehtävälistalle. Sprintin aikana sprintin tavoitteita ei voida muuttaa. Sprintin päätteeksi järjestetään sprintin läpikäynti, jossa tuoteomistajalle esitellään tuotteen edistymistä. Sprintin lopuksi järjestetään myös retrospektiivinen palaveri, jossa kulunut sprintti käydään läpi ja tarvittaessa optimoi-

daan julkaisusuunnitelmaa ja prosesseja. Scrumin työnkulku on esitetty tarkemmin kuviossa 2.



Kuvio 2. Scrumin työnkulku.

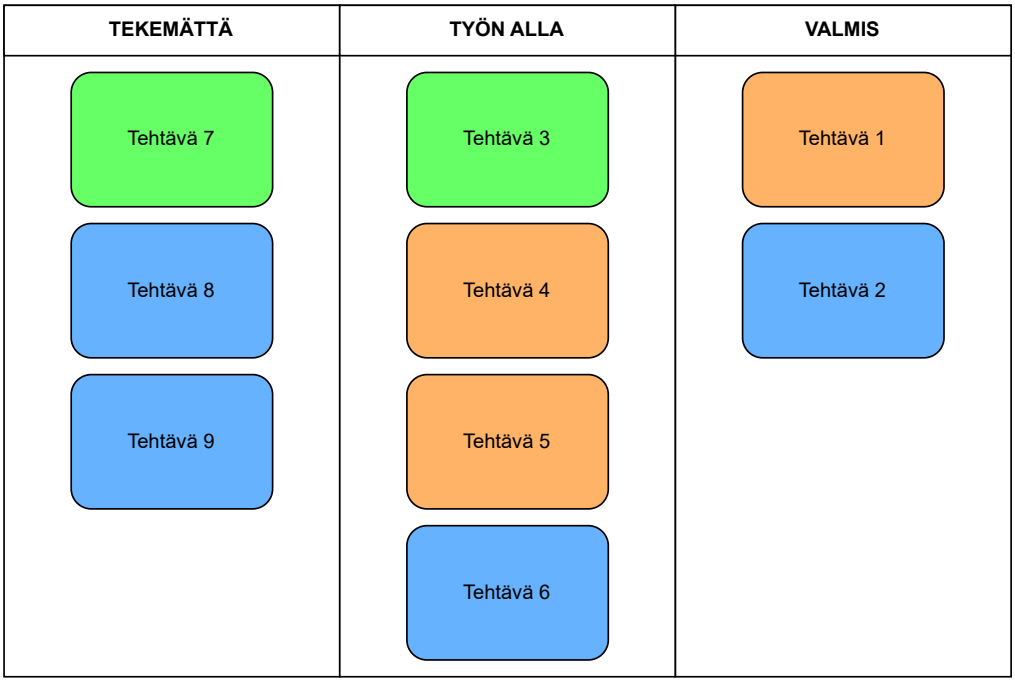
2.3 Kanban

Kanban on yleisesti käytössä oleva ketterä menetelmä, jota käytetään usein yhdessä myös muiden ketterien menetelmien kanssa. Esimerkiksi Hoda, Salleh ja Grundy (2018) artikkelissa mainitussa vuoden 2018 ketteryys kyselyssä vastanneiden organisaatioista 8% käytti Scrumin ja Kanbanin yhdistelmää ja 5% käytti vain Kanbania. Yhteensä 13% organisaatioista käytti siis Kanbania jossain muodossa. Ahmad, Markkula ja Oivo (2013) artikkelissa mainitut Kanbanin perusperiaatteet ovat: työnkulun visualisointi, mittaaminen ja hallitseminen, keskeneräisen työn määrän rajoittaminen, kehitysprosessin selkeyttäminen sekä yhteistyön parantaminen. Kanbanissa tavoitteena on sopeutua nopeasti prosessiin käyttämällä lyhyitä palautesilmukoita. Avainsyy Kanbanin käyttöön on se, että työn etenemiseen voidaan keskittyä täysillä ja se että siinä ei ole pakollisia iteraatioita.

Ahmad, Markkula ja Oivo (2013) mainitsevat artikkelissaan Kanban-taulun, joka on keskeisessä osassa Kanbania. Taulua käytetään ohjelmistokehitysprosessin visualisointiin. Visualisointi tapahtuu taulun avulla koska siitä näkee selkeästi jokaiselle kehittäjälle määrätyt työt, projektin prioriteetit ja mahdolliset pullonkaulat. Taulun tavoitteena on vähentää keskeneräi-

sen työn määrää keskittymällä vain pyydettyjen esineiden (engl. *work item*) kehittämiseen. Koska kehittäjät keskittyvät vain muutamaan annettuun esineeseen kerrallaan, asiakkaalle voidaan jatkuvasti toimittaa valmiita ja julkaistuja esineitä.

Kuviossa 3 on esitetty yksinkertainen Kanban-taulu. Taulu voidaan jakaa esimerkiksi kolmeen seuraavaan osaan: *tekemättä*, *työn alla* ja *valmis*. Kaikki tehtävät listataan aluksi vasemmalle *tekemättä*-osioon, josta tehtävät siirretään yksitellen oikealle seuraaviin osioihin tehtävän edetessä. Kun tekemätöntä tehtävää aletaan toteuttaa, se siirretään *työn alla*-osion alle. Työn alla olevan tehtävän toteuduttua, tehtävä siirretään lopuksi *valmis*-osioon. Kanban-taulussa tehtäville voidaan antaa värit havainnollistamaan esimerkiksi työn prioriteettia. Kuviossa 3 oranssit tehtävät ovat esimerkiksi prioriteetiltaan korkeimmat ja vihreät matalimmat.



Kuvio 3. Esimerkki Kanban-taulusta.

Ahmad, Markkula ja Oivo (2013) mainitsevat artikkelissaan useita hyötyjä, jotka saavutetaan käyttämällä Kanbania. Koska Kanban rajoittaa keskeneräisen työn määrää tiimin kapasiteetin mukaan, vaatimukset tasaantuvat tiimin todellisen suorituskyvyn mukaisesti. Ra-

joittamalla keskeneräisiä töitä kehitystyön tahtia voidaan ylläpitää, jonka ansiosta tuotteiden laatu ja tiimin tehokkuus paranee. Kehitysprosessin visualisoinnin ansiosta ongelmat tulevat helpommin esiin, joka vuorostaan vähentää virheiden määrää ja auttaa ylläpitämään tasaista työnkulkua. Kehittyneen työnkulun ja laadun paranemisen ansiosta julkaisuja voidaan tehdä säännöllisesti, jonka ansiosta luottamus asiakkaiden kanssa kasvaa.

2.4 Extreme Programming

Extreme programming (XP) ei ole tällä hetkellä yleisimpien ketterien menetelmien joukossa. Sen sijaan joitain XP:n periaatteita, kuten esimerkiksi pariohjelmointia, käytetään yhdessä muiden ketterien menetelmien kanssa. Vuoden 2018 ketteryys kyselyyn vastanneiden organisaatioista 6% käytti Scrumin ja XP:n yhdistelmää ja vain 1% käytti pelkkää XP:a (Hoda, Salleh ja Grundy 2018). Artikkelissaan Beck (1999) kuvaa XP käytäntöjä seuraavasti: asiakas päättää julkaisujen skaalan ja ajankohdan perustuen koodaajien arvioihin, julkaisut ovat pieniä ja niitä tehdään usein, kaikki ohjelmointi toteutetaan parin kanssa yhdellä työpisteellä eli pariohjelmointina, koodaajat luovat jatkuvasti työtehtävän mukaisia testitapauksia, testitapauksien tulee aina mennä läpi, duplikaatti koodia ei saa olla ja koodin pitää sisältää mahdollisimman vähän luokkia ja metodeja, uusi koodi integroidaan jatkuvasti nykyisen koodin kanssa säilyttäen testitapausten eheyden ja jokainen kehittäjä on vastuussa koodin parantamisesta jos näkee siihen tilaisuuden.

Beck (1999) on kuvannut artikkelissaan XP kehityssyklin kattavasti. Kehityssykli alkaa siitä kun asiakas valitsee seuraavaa julkaisua varten kaikkien toteutettavien toimintojen joukosta tärkeimmät. Toimintoja kutsutaan tarinoiksi (engl. *stories*) ja jokaisen tarinan tulee testattavissa, arvioitavissa ja liittyä liiketoimintaan. Asiakkaalle tiedotetaan jokaisen tarinan toteutukseen kuluva aika sekä kustannukset. Kun julkaisun tarinat on valittu, asiakas valitsee jäljellä olevien julkaisun tarinoiden joukosta tarinat, jotka toteutetaan seuraavan lyhyen iteraation aikana. Tämän jälkeen kehittäjät pilkkovat iteraation tarinat pieniksi tehtäviksi, jotka yksi henkilö pystyy toteuttamaan muutaman päivän aikana. Seuraavaksi tehtävät jaetaan kehittäjien kesken, jonka jälkeen tehtävästä vastuussa oleva henkilö arvioi tehtävään kuluvan ajan. Arviointien jälkeen tehtäviä voidaan jakaa uusiksi kehittäjien kesken, mikäli jollain kehittäjällä on enemmän tai vähemmän töitä kuin muilla.

Kun tehtävät on jaettu, jokainen tehtävä muunnetaan joukoksi testitapauksia, joiden läpäiseminen tarkoittaa että tehtävä on valmis. Testitapausten luomisen jälkeen aloitetaan kehitystyö pariohjelmointia noudattaen, luoden mahdollisimman yksinkertaista koodia. Kun tehtävä on saatu valmiiksi, sen koodi ja testitapaukset integroidaan nykyiseen järjestelmään sillä edellytyksellä, että kaikki järjestelmän testitapaukset menevät edelleen läpi myös integroinnin jälkeen. Iteraation aikana asiakas voi toimittaa toiminnallisia testejä, jotka lisätään järjestelmään. Iteraation lopuksi kaikki, niin kehittäjien luomat testitapaukset kuin asiakkaan toiminnalliset testit, tulee mennä läpi. Jokaisen iteraation tavoitteena on saada valmiiksi uusia tarinoita, jotka on testattu ja valmiina tuotantoon vientiä varten.

2.5 Lean

Leanin käyttäminen tänä päivänä on harvinaisempaa. Hoda, Salleh ja Grundy (2018) artikkelissa mainitussa vuoden 2018 ketteryys kyselyssä vastanneiden organisaatioista vain 1% käytti Leania. Poppendieck ja Cusumano (2012) käsittelevät artikkelissaan Leanin perusperiaatteita ja niiden soveltamista sovelluskehitykseen. Lean ei sisällä käytäntöjä tai harjoitteita, joita noudattamalla ohjelmistokehitysprosessia viedään eteenpäin. Leaniin ei esimerkiksi kuulu pakollisia iteraatioita eikä Lean määritä työnkulkua tai kehityssykliä. Sen sijaan Lean sisältää joukon perusperiaatteita, joita tulisi noudattaa ohjelmistokehitysprojektin aikana.

Poppendieck ja Cusumano (2012) artikkelissa mainitut Leanin perusperiaatteet ovat seuraavat: kokonaisuuden optimointi, hukan karsiminen, nopea julkaiseminen, laadun ylläpitäminen, kaikkien osallistaminen sekä jatkuva oppiminen ja kehittyminen. Kokonaisuuden optimoinnilla tarkoitetaan sitä, että sovelluksen arvoon eniten vaikuttava tekjä on kyky muokata ja optimoida koodia ajan myötä. Hukan karsimisen mukaisesti kaikki sellainen on hukkaa, joka ei suoraan luo arvoa asiakkaalle tai lisää tietämystä siitä kuinka arvoa luotaisiin tehokkaammin. Suurimmat syyt hukan muodostumiseen on tarpeettomat toiminnot, tiedon katoaminen, osittain tehdyt työt, luovuttaminen, multitaskaaminen sekä työajasta suurimman osan käyttäminen virheiden etsimiseen ja korjaamiseen. Leanissa julkaisuja pitää tehdä usein. Poppendieck ja Cusumano (2012) mainitsevat, että tuotantoon vientejä voidaan tehdä viikoittain, päivittäin tai jopa jatkuvasti. Tämä kuitenkin edellyttää että käytetään erilaisia virheiden estämismekanismeja, kuten esimerkiksi testiautomaatiota, havaitsemaan muutok-

sista mahdollisesti aiheutuvia virheitä.

Poppendieck ja Cusumano (2012) mainitsevat artikkelissaan että laatua ylläpidetään integroimalla jatkuvasti pieniä sovelluksen osia laadun parantamiseksi. Artikkelin mukaan sovelluskehitys osaston ei yksinään tulisi olla vastuussa sovelluskehityksestä. Kehittäjien lisäksi sovelluskehityksessä pitäisi olla mukana kaikki sellaiset henkilöt, jotka pystyvät tuomaan lisäarvoa projektille. Projektissa voi olla mukana esimerkiksi kehittäjiä, testaajia, suunnittelijoita, tukihenkilöitä, operatiivisia henkiöitä, rahoitusalan ammattilaisia sekä asiakkaita ymmärtäviä henkiöitä. Leanissa ihmisten voimaantuminen, tiimityöhön kannustaminen ja mahdollisimman matalan tason päätöksenteon mahdollistaminen on olennaista. Jatkuvan oppimisen periaatteen mukaisesti aluksi on luotava jotain minimivaatimusten mukaisesti alkuun pääsemiseksi. Sen jälkeen tehdään julkaisuja säännöllisesti, jotta päätöksenteossa voidaan hyödyntää asiakkaiden kokemuksista saatua palautetta. Prosessin ansiosta sellaisen työn määrä pysyy minimaalisena, jota asiakas ei koe arvokkaaksi. Leanin jatkuvan kehittymisen periaatteen mukaisesti jokaista järjestelmää tulisi kehittää jatkuvasti. Vaikka jokin järjestelmä toimisikin jo hyvin, tulee sitä silti pyrkiä kehittämään paremmaksi.

2.6 Feature-Driven Development

Feature-driven developmentia (FDD) ei mainittu ollenkaan vuoden 2018 ketteryys kyselyssä (Hoda, Salleh ja Grundy 2018), josta voidaan päätellä että kyseinen ketterä metodi ei ole tällä hetkellä yleisesti käytössä. Metodin perusperiaatteet käydään kuitenkin läpi, koska metodi esiintyi osassa tämän systemaattisen kirjallisuuskatsauksen artikkeleista. Chowdhury ja Huda (2011) käsittelevät artikkelissaan FDD:n ennalta määritettyjä tekniikoita ja käytäntöjä, joita tulee noudattaa sovelluskehityksen aikana.

FDD:n käytäntöihin kuuluu isojen ongelmien pilkkominen pienemmiksi hallittavan kokoisiksi ongelmiksi. FDD:n käytännöt mahdollistavat odottamattomiin muutoksiin reagoimisen jatkuvan integroinnin ja refaktoroinnin avulla. Käytäntöihin kuuluu myös ohjelmakoodin kollektiivinen omistus (engl. *collective code ownership*), jonka mukaan jokaisella tiimin jäsenellä on vastuu koodin päivittämisestä. Artikkelin mukaan FDD:n työnkulussa keskitytään pääasiassa kahteen eri vaiheeseen: toteutettavien toimintojen selvittämiseen ja listaamiseen

sekä listattujen toimintojen toteuttamiseen yksitellen.

Ensimmäinen vaihe eli toteutettavien toimintojen selvittäminen ja listaaminen on todella kriittinen. Projektin seurannan tarkkuus sekä ohjelmakoodin ylläpidettävyys ja laajennettavuus määräytyvät pääasiassa sen mukaan, kuinka hyvin ensimmäisen vaiheen työt on toteutettu. Asiakkaat osallistuvat täysipäiväisesti ensimmäiseen vaiheeseen, jonka aikana ongelmakohdista piirretään UML-kaavioita. Sen jälkeen lista toteutettavista toiminnoista johdetaan kaavioiden perusteella. Toiminnot tulee olla ilmaistuna sellaisella kielellä, jota sekä kehittäjät että asiakkaat ymmärtävät selkeästi. Listattujen toimintojen tulee olla niin pieniä, että ne on mahdollista toteuttaa lyhyiden, yleensä yhden-kolmen viikon mittaisten, iteraatioiden aikana.

Toisen vaiheen eli toimintojen toteuttamisen alkaessa luodaan niin sanottuja työpaketteja (engl. *work packages*). Kukin työpaketti sisältää joukon toisiinsa liittyviä toimintoja ja paketit ovat sen kokoisia, että yksi paketti voidaan toteuttaa yhden iteraation aikana. Toteutuksen aikana FDD:ssa käytetään muun muassa yksikkötestejä ja pariohjelmointia toteutuksen toimivuuden verifioimiseksi. Työpaketin toteutuksen jälkeen, paketti luovutetaan asiakkaalle testattavaksi. Kun ratkaisu on todettu toimivaksi, se integroidaan järjestelmään.

3 Tutkimusasetelma

Tässä luvussa kuvataan tutkimuskysymykset, tutkimusmenetelmä ja tutkimuksen suorituksen eri vaiheet. Aliluvussa 3.1 kuvataan tutkimuskysymykset. Aliluvussa 3.2 kuvataan tutkimusta varten valittu tutkimusmenetelmä, syyt tutkimusmenetelmän valitsemiseen ja tutkimusmenetelmän ohjesäännöt. Aliluvussa 3.3 kuvataan hakustrategia, jonka avulla tutkimuksen lähdeaineisto on haettu. Aliluvussa 3.4 kuvataan lähdeaineistolle määritellyt kriteerit, joiden perusteella aineisto tutkimusta varten on hyväksytty tai hylätty. Aliluvussa 3.5 kuvataan artikkeleiden valintaprosessi. Lopuksi aliluvussa 3.6 kuvataan tiedon poiminta prosessi, jota noudattamalla valituista artikkeleista on kerätty tutkimuksen kannalta merkittävä ja olennainen tieto.

3.1 Tutkimuskysymykset

Tämä tutkimus pyrkii vastaamaan seuraaviin tutkimuskysymyksiin:

- *Mitä hyötyjä ketterien menetelmien käytössä on havaittu?*
- *Mitä haasteita ketterien menetelmien käytössä on havaittu?*

Ensimmäisen tutkimuskysymyksen avulla on tarkoitus selvittää, mitä empiirisesti tutkittuja hyötyjä ohjelmistokehitysprojekteissa on saavutettu käyttämällä ketteriä menetelmiä. Toisen tutkimuskysymyksen on tarkoitus selvittää, mitä empiirisesti tutkittuja haasteita ketterien menetelmien käytössä on ohjelmistokehitysprojekteissa. Yhdessä kysymykset vastaavat myös siihen, mitä tutkimuskohteita sovelluskehitysprojekteissa liittyen ketteriin menetelmiin on empiirisesti tutkittu.

3.2 Tutkimusmenetelmä

Tutkimusmenetelmäksi valikoitui systemaattinen kirjallisuuskatsaus. Petersen, Vakkalanka ja Kuzniarz (2015) esittävät artikkelissaan ohjesäännöt systemaattisen kirjallisuuskatsauksen toteutukseen, joita mukailemalla tämä tutkimus on toteutettu. Tutkimusmenetelmäksi valittiin systemaattinen kirjallisuuskatsaus koska sen avulla ketterien menetelmien empii-

risten havaintojen nykytilan selvittäminen onnistuu parhaiden. Lisäksi ketteriä menetelmiä ylistävää aineistoa on runsaasti, joten on tärkeää tutkia systemaattisesti mitä positiivisia tai negatiivisia empiirisiä havaintoja ketteristä menetelmistä on tehty. Tutkimuksen tavoitteena on löytää ja kerätä pelkästään empiirisiä havaintoja koskien ketteriä menetelmiä. Tutkimuksen avulla pyritään luomaan tiivistelmä aiempien empiiristen tutkimusten olennaisesta sisällöstä, sen sijaan että luotaisiin systemaattinen kirjallisuuskatsaus ketterien menetelmien tutkimusaiheista.

3.3 Hakustrategia

Tutkimuksen aineisto on kerätty kolmesta eri tieteellisiä artikkeleita sisältävästä tietokannasta. Tietokannoista aineistoa voidaan hakea käyttämällä hakulauseita, jotka rajaavat tuloksia esimerkiksi otsikon, tiivistelmän tai tekstin perusteella. Tutkimuksessa käytetyt tietokannat ovat "IEEE Xplore Digital Library" (2024), "ACM Digital Library" (2024) ja "SpringerLink" (2024).

Tietokantojen hakukoneet ovat erilaisia, joten myös hakulauseet ja hakujen toteutus eroavat toisistaan hieman. IEEE Xplore-tietokannasta oli mahdollista hakea suoraan käyttäen vain hakulauseketta. ACM Digital Library-tietokannasta haettaessa piti ensin valita tarkempi haku, jonka jälkeen `Search Within` kohtaan valittiin `Title` ja syötettiin hakulauseke. SpringerLink-tietokannasta hakeminen vaati eniten manuaalista työtä. Ensin piti valita tarkempi haku, joka mahdollistaa haun otsikon perusteella. Otsikon pystyi kuitenkin rajaamaan vain yhden sanan perusteella, joka oli hakua tehdessä `empirical`. Sana syötettiin `where the title contains` kenttään, jonka jälkeen suoritettiin haku. Haun jälkeen tuloksia on mahdollista rajata esimerkiksi sisällön tyypin tai tieteenalan perusteella. Hakua tehdessä rajauksiksi valittiin `Content Type: Article` ja `Discipline: Computer Science`, joka rajasi tulokset tietojenkäsittelytiede artikkeleihin. Lopuksi tulosten rajausten jälkeen, syötettiin vielä hakulauseke.

IEEE Xplore- ja ACM Digital Library-tietokantahauissa käytetyt hakulausekkeet kohdistuivat vain otsikkoon, kun taas SpringerLink-tietokannan hakulauseke kohdistui tiivistelmään ja tekstiin. Tutkimusaineiston hakutulokset ja hakulausekkeet on esitetty tarkemmin taulukossa

1. Hakulausekkeissa esiintyvä lyhenne *asd* tarkoittaa ketterää ohjelmistokehitystä (engl. *agile software development*). Lyhenne *agsd* tarkoittaa vastaavasti globaalia ketterää ohjelmistokehitystä (engl. *agile global software development*). Lyhenteet lisättiin mukaan hakulausekkeisiin, koska joissain artikkeleissa kyseisiä lyhenteitä käytettiin termin sijaan ja tavoitteena oli löytää mahdollisimman paljon aiheeseen liittyviä artikkeleita.

Tietokanta	**Tulokset**	**Hakulauseke**
SpringerLink	77	`Title: empirical, Content Type: Article, Discipline: Computer Science, (agile AND software) OR asd OR agsd`
IEEE	23	`("Document Title":empirical AND (("Document Title":agile AND "Document Title":software) OR "Document Title":asd OR "Document Title":agsd)) OR ("Publication Title":empirical AND (("Publication Title":agile AND "Publication Title":software) OR "Publication Title":asd OR "Publication Title":agsd))`
ACM	13	`[Title: empirical] AND [[[Title: agile] AND [Title: software]] OR [Title: asd] OR [Title: agsd]]`

Taulukko 1. Tietokannat, hakutulosten määrät ja käytetyt hakulausekkeet.

3.4 Valintakriteerit

Tässä tutkimuksessa lähdeaineistolle määriteltiin seuraavat hyväksymiskriteerit:

- vastaa tutkimuskysymyksiin eli sisältää empiiristä tietoa ketteristä menetelmistä
- aineisto on englannin- tai suomenkielinen
- teksti on kokonaan saatavilla maksuttomasti Jyväskylän yliopiston lukuoikeuksilla

- kyseessä on akateeminen julkaisu
- otsikossa on sana `empirical` ja lisäksi yksi seuraavista termeistä: `agile software`, `asd` tai `agsd`.

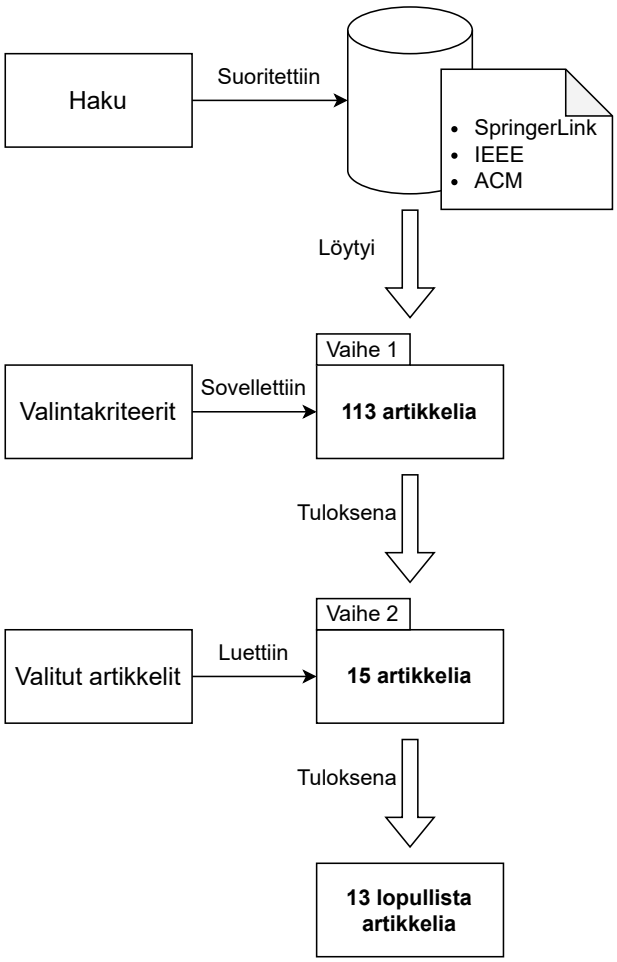
Lähdeaineistolle määrittellyt hylkäämiskriteerit taas olivat:

- aineiston empiirinen tieto ei koske ketteriä menetelmiä
- pituus alle viisi sivua
- aineisto on jo löytynyt aiemmassa tietokantahaussa.

Aineisto valittiin mukaan tutkimukseen vain silloin, jos kaikki hyväksymiskriteerit täyttyivät. Sen sijaan yhdenkin hylkäämiskriteerin täyttyminen johti aineiston hylkäämiseen. Tavoitteena oli kerätä empiiristen tutkimusten aineistoa, joka koski ketterien menetelmien käyttöä erilaisissa ohjelmistokehitysprojekteissa. Empiirisiä tutkimuksia, joissa käytettiin ketteriä menetelmiä, oli runsaasti, mutta usein tutkimuksen empiria kohdistui johonkin muuhun kuin ketteriin menetelmiin. Aineistoa jossa oli empiriisesti tutkittu ketteriä menetelmiä, oli vain vähän saatavilla.

3.5 Valintaprosessi

Lähdeaineiston haku systemaattista kirjallisuuskatsausta varten suoritettiin 17.1.2024. Aineiston haku- ja valintaprosessi on esitetty kuviossa 4. Hakutuloksia tuli yhteensä 113 kappaletta. Tämän jälkeen hakutuloksiin sovellettiin valintakriteereitä jonka jälkeen jäljelle jäi enää 15 artikkelia. Yleisin hylkäämisen syy oli se, että aineiston empiirinen tieto ei koskenut ketteriä mentelmiä. Seuraavaksi artikkelit luettiin läpi ja varmistettiin, että ne vastaavat tutkimuskysymyksiin. Lukemisen jälkeen kaksi artikkelia karsittiin vielä pois, koska ne eivät vastanneet tutkmuskysymyksiin. Toisessa artikkelissa oli esimerkiksi tutkittu empiirisesti mitä ketteriä menetelmiä käytetään eniten, mutta ei menetelmien hyötyjä tai haasteita. Valintakriteereiden soveltamisen jälkeen jäljelle jäi lopulta yhteensä 13 artikkelia, joista kahdeksan löytyi IEEE Xplore-tietokannasta ja viisi ACM Digital Library-tietokannasta. SpringerLink-tietokannasta ei valittu yhtään artikkelia lopulliseen tutkimukseen. Tutkimukseen valitut artikkelit on esitetty taulukossa 3.



Kuvio 4. Aineiston haku- ja valintaprosessi.

3.6 Tiedon poiminta

Artikkeleiden valitsemisen jälkeen suoritettiin tietojen poimiminen. Ensin artikkelit luettiin läpi, jonka jälkeen jokaisesta niistä kerättiin ja talletettiin olennaiset tiedot erilliseen tiedostoon. Petersen, Vakkalanka ja Kuzniarz (2015) esittävät artikkelissaan tiedonpoimintalomakkeen (engl. *data extraction form*), jonka mukaan luotiin taulukon 2 mukainen lomake tietojen poimintaa ja tallettamista varten.

Artikkeleista pyrittiin poimimaan kaikki merkittävä tieto. Jos tieto koettiin merkittäväksi, tiedostoon luotiin uusi kenttä, jonka alle tieto kirjattiin. Jos tiedostossa oli jo valmiiksi tie-

Lyhenne	Kenttä
K1	Tekijä(t)
K2	Nimi
K3	Vuosi
K4	Tutkimusmenetelmät
K5	Tutkimuskohteet
K6	Ketterät menetelmät
K7	Hyödyt
K8	Haasteet

Taulukko 2. Tietojen poimintalomake.

dolle sopiva kenttä, uutta kenttää ei luotu vaan tieto kirjattiin olemassa olevan kentän alle. Kerätty tieto säilytettiin mahdollisimman alkuperäisenä, jotta tiedon poiminta ei aiheuttaisi virheellistä tulkintaa.

Tiedoston kentät K1–K3 sisältävät artikkelin perustiedot. Tutkimusmenetelmillä (K4) tarkoitetaan menetelmiä, joilla artikkelin empiirinen tieto oli kerätty, kuten esimerkiksi *haastattelemalla*. Tutkimuskohteilla (K5) tarkoitetaan kohdetta, johon artikkelin empiirinen tutkimus kohdistuu, kuten esimerkiksi *henkilöstöön*. Ketterillä menetelmillä (K6) tarkoitetaan artikkelissa tutkittuja ketteriä menetelmiä. Kentät K7–K8 sisältävät vastaukset tutkimuskysymyksiin eli mitä empiirisesti tutkittuja hyötyjä tai haittoja artikkelissa mainittiin.

Tietojen poiminnan jälkeen tietoja tarkasteltiin ja arvioitiin kokonaisuutena, joka johti joidenkin tietojen yhtenäistämiseen. Esimerkiksi erilaisia tutkimuskohteita oli yli kymmenen kuten esimerkiksi *tehokkuus*, *tietoturva* ja *bugien määrä*. Paremman kokonaiskuvan saavuttamiseksi tutkimuskohteet yhtenäistettiin kolmeen uuteen kategoriaan jotka olivat: *projektin hallinta*, *tekninen artefakti* ja *henkilöstö*.

Tietojen poiminnan jälkeen joitain tietoja myös poistettiin tiedostosta. Tiedostoon kirjattiin aluksi myös artikkelissa tutkittujen tiimien koot ja roolit. Tiedot kuitenkin poistettiin koska ne eivät olleet merkittäviä tämän tutkimuksen kannalta. Lähes puolissa artikkeleissa (5 kpl) ei oltu mainittu tiimin kokoa ollenkaan, tai koolla ei ollut merkitystä tutkimuksen kannal-

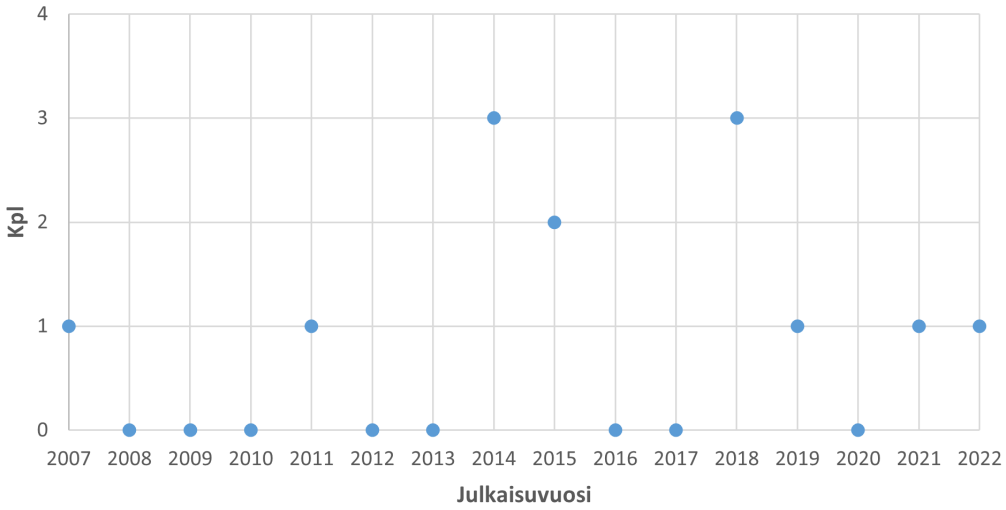
ta, esimerkiksi jos tutkimus perustui ohjelmakoodin analysointiin. Artikkeleiden perusteella, joissa tiimin koko mainittiin, sekä pienet että suuret tiimit esiintyivät tutkimuksissa yhtä paljon. Vastaavasti monessa artikkelissa (4 kpl) ei oltu mainittu tiimin rooleja, tai rooleilla ei ollut merkitystä tutkimuksen kannalta, esimerkiksi silloin kun tutkimus perustui ohjelmakoodin analysointiin. Artikkeleiden perusteella, joissa tiimin roolit oli mainittu, kaikki roolit, kuten esimerkiksi *tiimin jäsen*, *tiimin johtaja* ja *projektipäällikkö*, esiintyivät tutkimuksissa yhtä paljon.

4 Tulokset

Tässä luvussa esitetään systemaattisen kirjallisuuskatsauksen tulokset. Aliluvussa 4.1 esitetään katsaukseen valitut artikkelit. Aliluvussa 4.2 esitetään artikkeleissa käytetyt tutkimusmenetelmät. Aliluvussa 4.3 esitetään artikkeleissa tutkitut ketterät menetelmät. Aliluvussa 4.4 esitetään tutkimuskohteet. Aliluvussa 4.5 esitetään hyödyt, jotka on havaittu ketteriä menetelmiä käytettäessä. Lopuksi aliluvussa 4.6 esitetään ketteriin menetelmiin liittyvät haasteet.

4.1 Valitut artikkelit

Katsaukseen valittiin yhteensä 13 artikkelia, jotka on esitetty taulukossa 3. Kaikki valitut artikkelit on julkaistu vuosina 2007–2022, joista suurin osa (8 kpl) on julkaistu vuosina 2014–2018. Artikkeleiden kappalemäärät julkaisuvuosittain on esitetty kuviossa 5.



Kuvio 5. Artikkeleiden julkaisuvuodet.

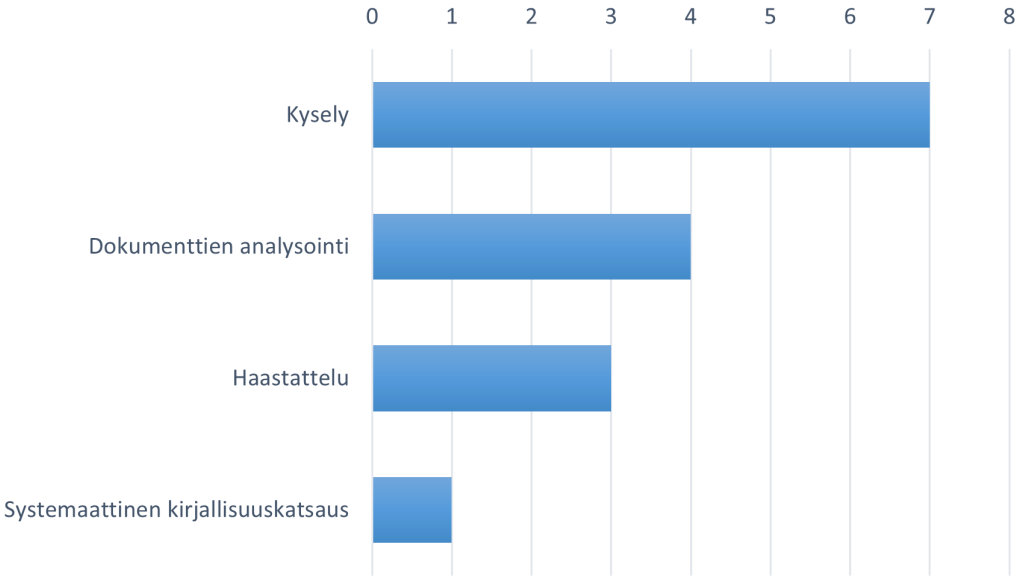
4.2 Tutkimusmenetelmät

Tiedon poiminnassa katsaukseen valituista artikkeleista kirjattiin ylös käytetyt tutkimusmenetelmät, joita noudattamalla tutkimuksen empiirinen tieto oli kerätty. Artikkeleissa esiintyi

Tekijä(t) ja vuosi	Artikkelin nimi
Hsu ja Lin (2018)	How Agile Impacts a Software Corporation: An Empirical Study
Britto, Mendes ja Börstler (2015)	An Empirical Investigation on Effort Estimation in Agile Global Software Development
Kassab (2014)	An Empirical Study on the Requirements Engineering Practices for Agile Software Development
Vithana, Asirvatham ja Johar (2018)	An Empirical Study on Using Agile Methods in Global Software Development
Iqbal, Omar ja Yasin (2019)	An Empirical Analysis of the Effect of Agile Teams on Software Productivity
Cao (2022)	Estimating Efforts for Various Activities in Agile Software Development: An Empirical Study
Shawky ja Abd-El-Hafiz (2014)	The impact of agile approaches on software quality attributes an empirical study
Saeed ym. (2021)	An Empirical Investigation on Cost Estimation Challenges in Agile Software Development (ASD) Context
Kelle ym. (2015)	An empirical study into social success factors for agile software development
Capiluppi ym. (2007)	An Empirical Study of the Evolution of an Agile-Developed Software System
Heijden, Broasca ja Serebrenik (2018)	An empirical perspective on security challenges in large-scale agile software development
Schmidt, Ganesha Venkatesha ja Heymann (2014)	Empirical insights into the perceived benefits of agile software engineering practices: a case study from SAP
Stettina ja Heijstek (2011)	Necessary and neglected? an empirical study of internal documentation in agile software development teams

Taulukko 3. Kirjallisuuskatsaukseen valitut artikkelit.

neljä erilaista tukimusmenetelmää, jotka olivat *kysely*, *haastattelu*, *dokumenttien analysointi* ja *systemaattinen kirjallisuuskatsaus*. Käytettyjen tutkimusmenetelmien kappalemäärät on esitetty kuviossa 6. Osassa artikkeleista oli käytetty useampia tutkimusmenetelmiä, jonka vuoksi kappalemäärä on suurempi kuin artikkeleiden määrä.



Kuvio 6. Artikkeleiden tutkimusmenetelmät.

Eniten käytetty tutkimusmenetelmä oli kysely, jota käytettiin seitsemän artikkelin tutkimuksessa. Esimerkiksi Kelle ym. (2015) toteuttivat verkkokyselyn, johon vastasi 141 henkilöä, jotka työskentelivät 40:n eri projektin parissa, 19 eri hollantilaisen organisaation alla. Jokaisesta projektista vähintään yksi tiimin jäsen, Scrum Master ja tuoteomistaja vastasivat kyselyyn, jotta rooleja ja projekteja voitaisiin verrata keskenään. Schmidt, Ganesha Venkatesha ja Heymann (2014) toteuttivat verkkokyselyn, johon vastasi yli 200 SAP työntekijää, joista 15 oli tuoteomistajia ja 174 kehittäjiä. Vastaajat työskentelivät 74:ssä eri tiimissä ja viidessä eri maassa. Kaikki heistä olivat osallistuneet samaan harjoitteluohjelmaan vähintään puoli vuotta ennen kyselyyn kutsumista. Stettina ja Heijstek (2011) toteuttivat verkkokyselyn, johon vastasi 79 ohjelmistöinsinööriä kahdeksasta eri tiimistä, jotka sijaitsivat 13 eri maassa. Kyselyyn vastanneista suurin osa oli kehittäjiä (47%) tai Scrum Mastereita (18%), mutta kyselyyn vastasi myös muun muassa tuoteomistajia, konsultteja ja ketteryyden valmentajia. Kysely lähetettiin vain henkilöille, jotka olivat aktiivisesti mukana Scrumia käyttävissä

kehitystiimeissä. Britto, Mendes ja Börstler (2015) toteuttivat verkkokyselyn, johon vastasi 51 työmääräarvioinnin parissa työskentelevää globaalin ketterän tiimin jäsentä yhteensä 12 eri maasta. Kassab (2014) toteutti verkkokyselyn, johon vastasi 247 henkilöä 23:sta eri maasta. Suurin osa vastanneista olivat kehittäjiä tai testaajia (61%) ja loput (39%) olivat sovellusarkkitehtejä, projektipäälliköitä, analyytikoita tai konsultteja. Vithana, Asirvatham ja Johar (2018) toteuttivat verkkokyselyn, johon vastanneista henkilöistä 334 olivat kelvollisia tutkimuksen kannalta. Vastaajat koostuivat kaikista ohjelmistokehitystiimin jäsenistä ja kaikki heistä olivat Sri Lankalaisia ohjelmistoammattilaisia, jotka käyttivät ketteriä menetelmiä ulkomaalaisissa ohjelmistokehitysprojekteissa. Iqbal, Omar ja Yasin (2019) toteuttivat verkkokyselyn, johon vastasi ketteryyden ammattilaisia 52:sta eri pakistanilaisesta ohjelmistoyrityksestä. Vastaajista suurin osa oli ohjelmoijia (46,15%) ja ryhmänvetäjiä (21,15%).

Seuraavaksi eniten käytetty tutkimusmenetelmä oli dokumenttien analysointi, jota käytettiin neljän artikkelin tutkimuksessa. Esimerkiksi Capiluppi ym. (2007) analysoivat pienen mainosyrityksen kehittämän sovelluksen ohjelmakoodia, jonka avulla verkkosivun sisällöstä voidaan päätellä käyttäjän mielenkiinnon kohteet. Tiedon keräämisen ja analysoinnin he toteuttivat käyttäen omia työkalujaan. Tietoa kerättiin noin kahden ja puolen vuoden ajalta kun yrityksen sovellusta kehitettiin ja sovelluksesta analysoitiin esimerkiksi sen kokoa ja kompleksisuutta. Hsu ja Lin (2018) analysoivat singaporelaisen ohjelmistoyrityksen, Titansoftin, tarjoamia tietoja ennen sitä ja sen jälkeen kun yritys alkoi käyttää ketteriä menetelmiä. Yrityksen tarjoama tietoaineisto sisälsi seuraavat tiedot: kuukausittaiset aktiiviset käyttäjät, palvelun häiriöaika minuutteina, raportoitujen virheiden määrä, irtisanomisten määrä ja työntekijöiden tuntemukset yhdellä sanalla kuvattuna jokaiselta vuodelta kun he ovat työskennelleet yrityksessä. Cao (2022) analysoi erään ohjelmistoyrityksen tarjoamia projektitietoja. Yritys kehitti projektinhallintajärjestelmää ketterien ohjelmistoprojektien hallintaan. Projektin tiedot olivat yli 21:n kuukauden ajalta ja ne sisälsivät muun muassa arvioidut ja toteutuneet työmäärät koskien seuraavia tehtäviä: ominaisuuksien luominen, bugien korjaaminen ja ohjelmakoodin refaktorointi. Shawky ja Abd-El-Hafiz (2014) analysoivat 20:n eri sovelluksen avointa lähtekoodia, joista yhdeksän oli kehitetty käyttäen ketteriä menetelmiä ja 11 käyttäen vesiputousmallia.

Haastattelua käytettiin tutkimusmenetelmänä kolmen artikkelin tutkimuksessa. Esimerkik-

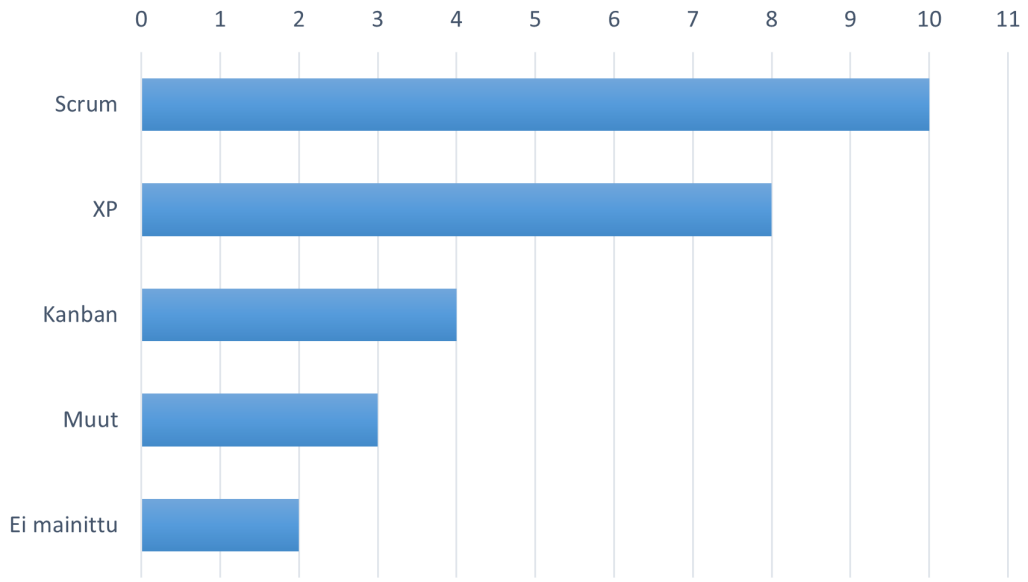
si Kelle ym. (2015) haastattelivat suurien ketterien projektien parissa työskenteleviä henkiöitä vahvistaakseen aiemmissa tutkimuksissa tunnistetut onnistumistekijät. Haastattelujen jälkeen Kelle ym. (2015) toteuttivat aiemmin mainitun verkkokyselyn. Heijden, Broasca ja Serebrenik (2018) järjestivät puolistrukturoituja haastatteluja hollantilaisen monikansallisen pankin, Rabobankin, kanssa. Haastateluja järjestettiin yhteensä kymmenen ja niihin osallistui eri rooleissa olevia ketterien tiimien jäseniä viidestä eri tiimistä. Schmidt, Ganesha Venkatesha ja Heymann (2014) haastattelivat useita SAP:in kehittäjiä ja ketteryyden valmentajia oppiakseen ketterien menetelmien omaksumisesta ja vaikutuksesta kehittäjien työhön. Haastattelujen lisäksi Schmidt, Ganesha Venkatesha ja Heymann (2014) toteuttivat aiemmin mainitun verkkokyselyn. Verkkokyselyn jälkeen he järjestivät vielä uusia haastatteluja kahden eri tiimin kehittäjien ja projektipäälliköiden kanssa keskustellakseen kyselyn tuloksista. Toinen tiimeistä sijtaitsi Saksassa ja toinen Intiassa.

Systemaattista kirjallisuuskatsausta käytettiin yhden artikkelin tutkimuksessa. Saeed ym. (2021) toteuttivat katsauksen, jota varten valittiin yhteensä 28 artikkelia. Artikkelit sisälsivät empiiristä tietoa kustannusten arvioinnista ja siihen liittyvistä haasteista ketterässä ohjelmistokehityksessä.

4.3 Ketterät menetelmät

Osana tutkimusta tarkasteltiin käytettyjä ketteriä menetelmiä. Artikkeleista kirjattiin ylös niissä esiintyneet ketterät menetelmät. Artikkeleissa esiintyi muun muassa seuraavat ketterät menetelmät: *Scrum*, *Extreme Programming (XP)*, *Kanban* ja *Feature-Driven Development (FDD)*. Koska artikkeleissa Scrum, XP ja Kanban esiintyivät selkeästi eniten (yhteensä 22 kertaa), jokainen niistä kirjattiin tietojen poimintalomakkeeseen omana arvonaan ja loput artikkeleissa esiintyneet ketterät menetelmät yhdistettiin kategorian *muut* alle. Kahdessa artikkelissa ei oltu mainittu ketteriä menetelmiä. Nämä artikkelit olivat Shawky ja Abd-El-Hafiz (2014) ja Saeed ym. (2021). Artikkeleissa esiintyneiden ketterien menetelmien kappalemäärät on esitetty kuviossa 7. Yhdessä artikkelissa saattoi esiintyä useita ketteriä menetelmiä, jonka vuoksi kappalemäärät eroavat artikkeleiden määrästä.

Eniten artikkeleissa esiintynyt ketterä menetelmä oli Scrum, joka esiintyi kymmenessä ar-



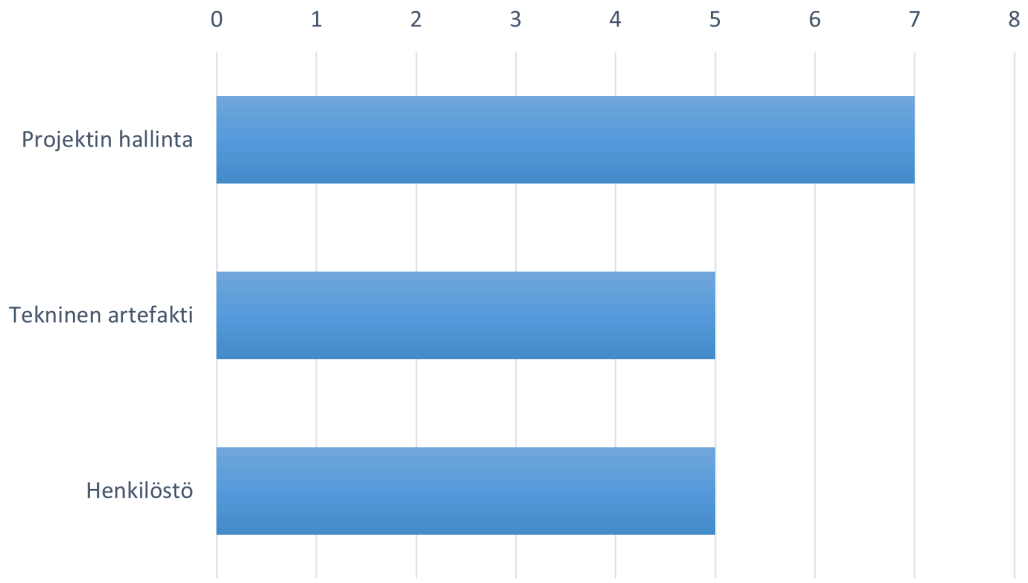
Kuvio 7. Artikkeleiden ketterät menetelmät.

tikkelissa. Esimerkiksi Kelle ym. (2015) toteuttamaan verkkokyselyyn vastasi Scrum tiimin jäseniä ja Scrum Masterereita. Stettina ja Heijstek (2011) lähettivät verkkokyselyn vain henkilöille, jotka käyttivät Scrumia. XP taas esiintyi kahdeksassa artikkelissa: esimerkiksi Capiluppi ym. (2007) toteuttamassa ohjelmakoodin analyysissä sovellus oli kehitetty käyttäen XP menetelmiä. Hsu ja Lin (2018) toteuttamassa analyysissä Titansoftissa käytettiin Scrumin ja Kanbanin lisäksi myös keskeisiä XP käytäntöjä, kuten esimerkiksi pariohjelmointia.

Kanban esiintyi neljässä artikkelissa kuten edellä mainitussa Hsu ja Lin (2018) artikkelissa, jonka mukaan Titansoftissa käytettiin Kanban tauluja kehitystiimien lisäksi myös henkilöstöosastolla. Britto, Mendes ja Börstler (2015) toteuttaman kyselyn mukaan suurin osa vastaajista oli käyttänyt Scrumia (84,31%), mutta myös Kanban (37,25%) ja XP (23,53%) olivat yleisesti käytössä. Muita kuin edellä mainittuja ketteriä menetelmiä esiintyi kolmessa artikkelissa. Esimerkiksi Kassab (2014) toteuttaman kyselyn mukaan vastaajista noin 10% oli käyttänyt Feature Driven Developmentia (FDD) ja noin 2% Leania. Hsu ja Lin (2018) mainitsevat, että Titansoftissa käytettiin myös Leania, Scrumin ja Kanbanin lisäksi.

4.4 Tutkimuskohteet

Artikkeleista kerättiin ylös tutkimuskohteet, joihin artikkelin empiirinen tutkimus kohdistui. Jotta kokonaiskuvan hahmottaminen olisi mahdollista, katsausta varten tutkimuskohteille luotiin kolme kategoriaa ja artikkeleissa esiintyneet tutkimuskohteet kirjattiin niiden alle. Katsauksessa käytetyt kategoriat ovat *projektin hallinta*, *tekninen artefakti* ja *henkilöstö*. Projektin hallinta kategoriaan tehtiin kirjaus esimerkiksi silloin kun artikkelissa tutkittiin työmäärän tai aikataulun arvioimista. Teknisen artefaktin alle kirjattiin esimerkiksi artikkelit, joissa tutkittiin sovellusten ohjelmakoodia. Henkilöstö kategoriaan sen sijaan kirjattiin artikkelit, jotka tutkivat esimerkiksi sosiaalisia tekijöitä tai henkilöiden välistä kommunikointia. Osassa artikkeleista oli useampi kuin yksi tutkimuskohde, jolloin artikkelista tehtiin kirjaus jokaiseen sitä koskevan kategorian alle. Artikkeleissa esiintyneet tutkimuskohteet kategorioittain ja niiden kappalemäärät on esitetty kuviossa 8.



Kuvio 8. Artikkeleiden tutkimuskohteet.

Artikkeleissa jokainen tutkimuskohde esiintyi lähes yhtä paljon. Projektin hallintaan kohdistuneita artikkeleita oli kuitenkin kaksi kappaletta enemmän kuin muita ja niitä oli yhteensä seitsemän. Esimerkiksi Schmidt, Ganesha Venkatesha ja Heymann (2014) ja Iqbal, Omar ja Yasin (2019) tutkivat kuinka ketterien menetelmien käyttö vaikuttaa projektin tuottavuuteen. Stettina ja Heijstek (2011) tutkivat ketterien menetelmien vaikutuksia projektin dokumentaa-

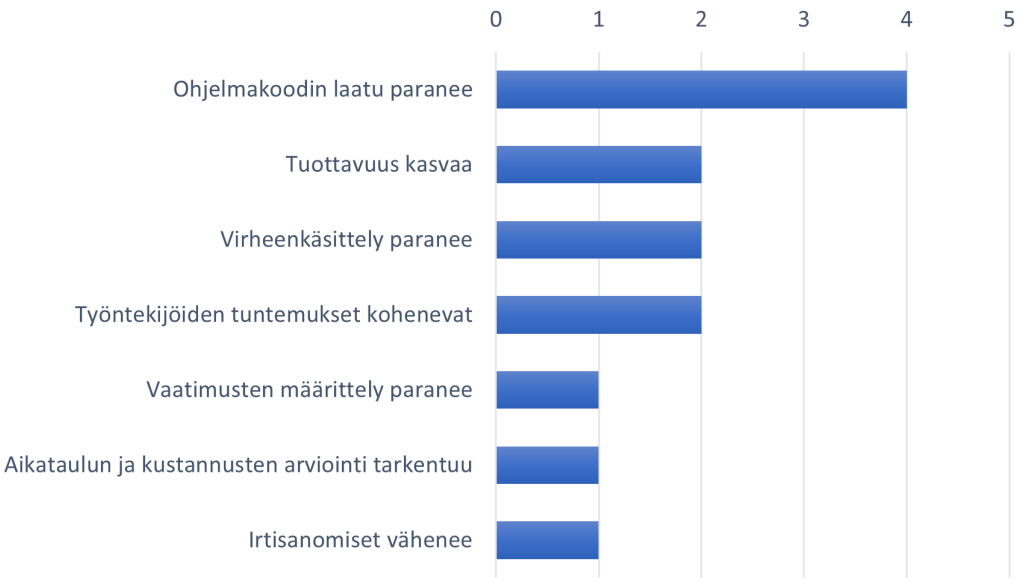
tioon. Britto, Mendes ja Börstler (2015) ja Cao (2022) tutkivat kuinka työmäärän arviointi onnistuu ketterissä projekteissa. Kassab (2014) ja Saeed ym. (2021) tutkivat vastaavasti vaatimusten määrittelyä sekä aikataulun ja kustannusten arvioimista.

Teknistä artefaktia ja henkilöstöä koskevia artikkeleita oli yhtä paljon ja niitä molempia oli viisi kappaletta. Teknistä artefaktia tutkivat esimerkiksi Capiluppi ym. (2007), Hsu ja Lin (2018), Shawky ja Abd-El-Hafiz (2014) ja Schmidt, Ganesha Venkatesha ja Heymann (2014), jotka pyrkivät selvittämään kuinka ketterien menetelmien käyttö vaikuttaa sovelluksen virheenkäsittelyyn ja sen ohjelmakoodin laatuun. Heijden, Broasca ja Serebrenik (2018) sen sijaan tutkivat kuinka ketterät menetelmät vaikuttavat sovelluksen tietoturvaan. Kelle ym. (2015), Britto, Mendes ja Börstler (2015) ja Vithana, Asirvatham ja Johar (2018) tutkivat henkilöstöä koskevia tekijöitä selvittäessään ketterien menetelmien vaikutuksia kommunikointiin ja menetelmien käyttöä isoissa projekteissa. Schmidt, Ganesha Venkatesha ja Heymann (2014) sekä Hsu ja Lin (2018) tutkivat ketteriä menetelmiä käyttävien työntekijöiden tuntemuksia ja sen vaikutuksia irtisanomisten määrään.

4.5 Hyödyt

Artikkeleissa mainittiin yhteensä seitsemän eri hyötyä, jotka saavutettiin käyttämällä ketteriä menetelmiä. Joissain artikkeleissa mainittiin useita eri hyötyjä kuten esimerkiksi Schmidt, Ganesha Venkatesha ja Heymann (2014) artikkelissa, jossa mainittiin neljä eri hyötyä. Artikkeleissa havaitut hyödyt olivat seuraavat: *ohjelmakoodin laatu paranee*, *tuottavuus kasvaa*, *virheenkäsittely paranee*, *työntekijöiden tuntemukset kohenevat*, *vaatimusten määrittely paranee*, *irtisanomiset vähenee* sekä *aikataulun ja kustannusten arviointi tarkentuu*. Havaitut hyödyt ja havaintojen kappalemäärät on esitetty kuviossa 9.

Yleisin havaittu hyöty, joka saavutettiin ketteriä menetelmiä käyttämällä, oli ohjelmakoodin laadun paraneminen. Kyseinen hyöty mainittiin neljässä eri artikkelissa: esimerkiksi Capiluppi ym. (2007) artikkelin mukaan päivittäiset palaverit mahdollistivat aktiivisen keskustelemisen ohjelmakoodin refaktoroinnista, joka vuorostaan johti koodin kompleksisuuden vähenemiseen. Artikkelin lopullisessa analyysissä sovelluksesta ei löydetty lähes yhtään kompleksista metodia kun verrattuna avoimen lähdekoodin sovelluksiin vastaavan analyysin



Kuvio 9. Artikkeleissa havaitut hyödyt.

mukaan ohjelmakoodeista 5%–10% oli kompleksisia metodeja. Myös Shawky ja Abd-El-Hafiz (2014) toteuttaman analyysin mukaan ketterien käytäntöjen, kuten esimerkiksi lyhyiden iteraatioiden, harjoittaminen johtaa sovelluksen kompleksisuuden vähenemiseen. Schmidt, Ganesha Venkatesha ja Heymann (2014) mainitsevat, että pariohjelmoinnissa kahden kehittäjän välinen jatkuva keskustelu ja työn läpikäyminen edistää sovelluksen laatua parantamalla esimerkiksi koodin modulaarisuutta ja ymmärrettävyyttä sekä vähentämällä bugien määrää. Hsu ja Lin (2018) täydentävät, että sovelluksen laatu parani koska raportoitujen bugien määrä väheni merkittävästi kun Titansoftissa alettiin käyttämään ketteriä menetelmiä.

Seuraavaksi yleisimmät hyödyt olivat tuottavuuden kasvu, virheenkäsittelyn paraneminen ja työntekijöiden tuntemusten koheneminen, jotka kaikki saivat kaksi mainintaa artikkeleissa. Schmidt, Ganesha Venkatesha ja Heymann (2014) artikkelissa tutkittiin ketterien menetelmien, etenkin pariohjelmoinnin, vaikutuksia tuottavuuteen ja virheiden käsittelyyn julkaisujen aikana. Julkaisuista kerättiin tiedot toteutettujen toimintojen ja kehitysvaiheen jälkeen havaittujen virheiden määrästä. Tutkimuksen mukaan ketterien menetelmien käyttöönoton jälkeen toteutettujen toimintojen määrä kasvoi yli 50% julkaisujen välillä vaikka uudet toiminnot olivat yhtä suuria ja kompleksisia kuin aiemmassa julkaisussa toteutetut toiminnot. Myös virheiden määrä laski merkittävästi (yli 60%) verrattuna aiempaan julkaisuun. Sch-

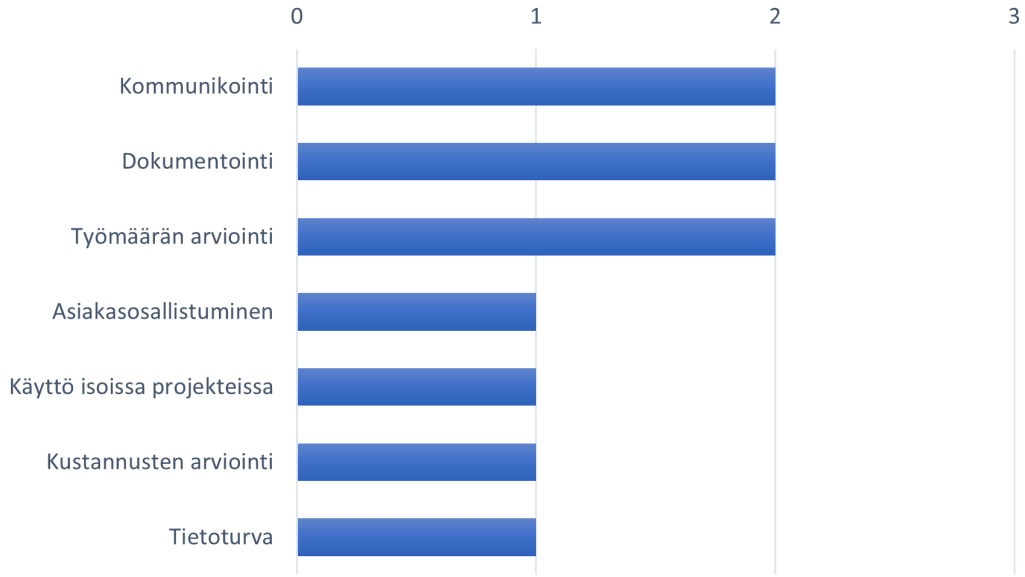
midt, Ganesha Venkatesha ja Heymann (2014) mainitsevat, että pariohjelmoinnilla oli myös positiivinen vaikutus tiimityöskentelyyn ja työntekijöiden tuntemuksiin. Pariohjelmoinnin ansiosta työntekijät olivat motivoituneempia tiimiinsä ja yhteisymmärrys tiimin jäsenten kesken parani. Lisäksi tiedon jakamisessa tiimin sisällä edistyttiin ja työntekijät olivat ylpeämpiä työpanoksestaan.

Iqbal, Omar ja Yasin (2019) toteuttaman kyselyn mukaan 57,69% vastanneista oli samaa mieltä väitteestä jonka mukaan ketterien menetelmien käyttäminen ohjelmistokehityksessä lisää tuottavuutta. Vastanneista 23,08% oli vahvasti samaa mieltä väitteestä ja 19,23% vastaajista olivat neutraaleja väittämän suhteen. Hsu ja Lin (2018) mainitsevat, että ketterien menetelmien käyttöönotto Titansoftissa laski merkittävästi palvelun häiriöaikaa sekä raportoitujen virheiden ja bugien määrää. Lisäksi välttämättömien tikettien käsittely vei vähemmän aikaa ja niihin pystyttiin vastaamaan aiempaa nopeammin. Artikkelin mukaan Titansoftin työntekijät olivat myös innostuneempia ja iloisempia otettuaan käyttöön ketterät menetelmät.

Loput havaitut hyödyt olivat vaatimusten määrittelyn paraneminen, irtisanomisten määrän väheneminen sekä aikataulun ja kustannusten arvioinnin tarkentuminen. Kaikki kyseiset hyödyt saivat vain yhden maininnan artikkeleissa. Hsu ja Lin (2018) mainitsevat, että ketterien menetelmien käyttöönotto voi johtaa irtisanomisten määrän hetkelliseen kasvuun jos työntekijät eivät ole samaa mieltä niiden käyttämisestä. Artikkelissa tutkittiin Titansoft Singaporen irtisanomisten määrää puolivuosittain kuuden vuoden ajalta ja huomattiin että irtisanomiset vähenivät alun hetkellisen kasvun jälkeen kun ketterät menetelmät otettiin käyttöön. Titansoft Taipein konttoria tutkittiin vastaavasti neljän vuoden ajalta ja myös siellä irtisanomiset vähenivät mutta ei yhtä huomattavasti. Kassab (2014) toteuttamassa kyselyssä tutkitiin kuinka vaatimusten määrittely onnistuu ketterissä projekteissa verrattuna vesiputousmallia hyödyntävään projektiin. Vastaajista noin 55% oli tyytyväisiä ketterän projektin vaatimusten määrittelyyn kun taas noin 35% oli tyytyväisiä vaatimusten määrittelyyn vesiputousmallia hyödyntävässä projektissa. Tutkimuksen mukaan ketterät projektit myös pysyivät paremmin aikataulussa ja budjetissa verrattuna vesiputousmallia hyödyntäviin projekteihin.

4.6 Haasteet

Artikkeleissa havaittiin seitsemän eri haastetta liittyen ketterien menetelmien käyttöön. Joissain artikkeleissa mainittiin useita eri haasteita kuten esimerkiksi Vithana, Asirvatham ja Johar (2018) artikkelissa, jossa mainittiin kolme erilaista ketteriin menetelmiin liittyvää haastetta. Artikkeleissa havaitut haasteet olivat seuraavat: *kommunikointi*, *dokumentointi*, *työmäärän arviointi*, *asiakasosallistuminen* (engl. *customer engagement*), *käyttö isoissa projekteissa*, *kustannusten arviointi* ja *tietoturva*. Havaitut haasteet ja havaintojen kappalemäärät on esitetty kuviossa 10.



Kuvio 10. Artikkeleissa havaitut haasteet.

Artikkeleissa eniten esiintyneet haasteet olivat kommunikointi, dokumentointi ja työmäärän arviointi, jotka kaikki saivat kaksi mainintaa. Esimerkiksi Vithana, Asirvatham ja Johar (2018) toteuttamassa kyselyssä pyrittiin selvittämään globaalien ketterien projektien haasteita. Kyselyn perusteella tunnistettiin kuusi seuraavaa haastetta, jotka vaikuttavat merkittävästi projektin onnistumiseen: kommunikointi, dokumentointi, asiakasosallistuminen, teknine pätevyys, teknologia ja työn koordinointi. Artikkelin mukaan tehokas kommunikointi on projektin kannalta elintärkeää, jonka vuoksi johdon pitää varmistaa että tiimin jäsenten kommunkointi on sujuvaa. Myös tiedon dokumentointi ja jakaminen on projektin onnistumisen kannalta tärkeää, joten dokumentit on luotava ja jaettava asianmukaisesti. Asiakkaan

tulee myös olla sitoutunut projektiin koko projektin ajan ja olla aktiivisesti mukana projektin päätöksissä. Tiimien tulisi olla teknisesti päteviä ja itsenäisiä, jotta työnjohtoa tarvitaan mahdollisimman vähän. Lisäksi etenkin maantieteellisesti jakautuneissa tiimeissä teknologian on mahdollistettava työnteko ja työt on pystyttävä koordinoimaan. Mielestäni artikkelissa mainituista haasteista kolme eivät koske pelkästään ketteriä projekteja, jonka vuoksi niitä ei erikseen kirjattu ylös eikä esitetty kuviossa 10. Kyseiset haasteet olivat tekninen pätevyys, teknologia ja työn koordinointi.

Stettina ja Heijstek (2011) toteuttaman kyselyn mukaan ketterissä projekteissa ei tuoteta tarpeeksi dokumentaatiota, koska ketterät menetelmät keskittyvät välittömään kommunikointiin painottaen että kattavasta sisäisestä dokumentaatiosta ei ole suoraa hyötyä loppuasiakkaalle. Kyselyyn vastanneista noin 22% koki projektin dokumentoinnin erittäin tärkeäksi ja noin 37,5% tärkeäksi. Vastanneista vain noin 5% koki dokumentoinnin merkityksettömäksi ja noin 1% erittäin merkityksettömäksi. Siitä huolimatta kyselyn vastaajista noin 36% oli sitä mieltä että dokumentaatiota on riittävästi ja noin 30% oli sitä mieltä että dokumentaatiota on liian vähän. Vastaajista vain noin 1% koki että dokumentaatiota on aivan liikaa ja noin 8% koki että dokumentaatiota on vähän liikaa.

Britto, Mendes ja Börstler (2015) artikkelissa tutkittiin työmäärän arviointia ketterissä ohjelmistokehitysprojekteissa. Tutkimuksen kyselyllä selvitettiin esimerkiksi kuinka tarkasti työmäärän arvioinnissa onnistutaan on ja mitä haasteita arviointiin liittyy. Kyselyyn vastanneista 33,33% oli sitä mieltä että toteutunutta työtä oli yli 25% tai enemmän kuin arvioitiin. 19,61% vastaajista oli sitä mieltä että työmääräarvio vastasi toteutunutta työmäärää eli toteutunut työmäärä erosi arviosta vain 0%–5%. Vastaajista yhteensä 54,90% oli sitä mieltä että projektien työmäärä oli aliarvioitu. Tutkimukseen vastanneista 39,2% koki työmäärän arvioinnin haastavaksi koska projektin vaatimukset oli dokumentoitu epäselvästi tai virheellisesti, jonka vuoksi projektin aikana tuli ilmi uusia odottamattomia töitä kasvattaen työmäärää. 27,5% vastanneista oli sitä mieltä että yhteenkuuluvuuden, teknisen pätevyyden ja kokemuksen puute tiimissä johtaa vääriin olettamuksiin vaikeuttaen työmäärän arviointia. Vastaajien mukaan vääriä olettamuksia syntyy myös silloin kun projektin työmäärän arvioinnin tekee jokin muu osapuoli kuin projektista vastuussa oleva kehitystiimi. 25,5% vastaajista koki että hajauteut tiimit vaikeuttavat työmäärän arviointia koska eroavaisuudet aikavyöhykkees-

sä, kielessä ja kulttuurissa vaikeuttaa kommunikointia, joka puolestaan vaikuttaa olennaisesti työmäärän arvioinnin tarkkuuteen. Vastaajista 4% oli sitä mieltä että työmäärän arviointi on haastavaa koska asiakkaan osallistuminen projektiin on vähäistä tai asiakas on priorisoinut tehtävät väärin.

Cao (2022) analysoi ketterän ohjelmistokehitysprojektin erilaisten työtehtävien työmäärän arviointia. Tutkimus pyrki selvittämään kuinka tarkasti työntekijät pystyvät arvioimaan työtehtävän työmäärän kun kyseessä on esimerkiksi uuden toiminnon kehittäminen, bugin korjaus tai ohjelmakoodin refaktorointi. Analyysin mukaan työntekijöiden arvioinnin tarkkuus ei parane ajan kanssa, vaikka heidän kokemuksensa kasvaisi. Artikkelissa mainitaan myös että työtehtävien välillä ei ole eroja siinä kuinka paljon arvio poikkeaa toteutuneesta työmäärästä, vaikka on yleisempää että bugien korjausten tai refaktoroinnin työmäärä yliarvioidaan verrattuna toiminnon kehittämiseen.

Loput artikkeleissa esiintyneet haasteet olivat asiakasosallistuminen, käyttö isoissa projekteissa, kustannusten arviointi ja tietoturva. Kaikki kyseiset haasteet saivat vain yhden mainin-nan artikkeleissa. Esimerkiksi Kelle ym. (2015) mainitsevat artikkelissaan että ketterien menetelmien käyttö isoissa projekteissa on mahdollista ja että projektin koko ei suoraan vaikuta projektin onnistumiseen. Artikkelin mukaan projektin koko sen sijaan vaikuttaa sosiaalisiin tekijöihin kuten esimerkiksi yhdenmukaisuuteen, ketteryyteen ja transformationaaliseen johtamiseen. Projektin koon kasvulla on negatiivinen vaikutus kyseisiin sosiaalisiin tekijöihin, jonka vuoksi isot projektit todennäköisemmin epäonnistuvat.

Saeed ym. (2021) toteuttivat systemaattisen kirjallisuuskatsauksen, jossa tutkittiin ketterien menetelmien käyttöön liittyviä kustannusten arvioinnin haasteita. Artikkelin mukaan ketterien menetelmien käyttö on yleistynyt, koska ne mahdollistavat vaatimusten muuttamisen projektin aikana. Muuttuvat vaatimukset vaikeuttavat kuitenkin merkittävästi kustannusten arviointia. Kirjallisuuskatsaukseen valittiin yhteensä 28 artikkelia ja jos kustannusten arviointiin vaikuttava tekijä mainittiin enemmän kuin puolissa artikkeleista, tekijä vaikutti kriittisesti kustannusten arviointiin. Tutkimuksen mukaan kustannusten arvioinnin tarkkuuteen eniten vaikuttava tekijä on kehittäjien taidot, joka mainittiin 78,57%:ssa artikkeleista. Seuraavaksi eniten mainintoja sai tiimin kokeneisuus, kommunikointi ja epäselvät tai muuttuvat vaatimukset, jotka kaikki mainittiin 64,28%:ssa artikkeleista. Tekninen pätevyys vai-

kuttaa myös kriittisesti kustannusten arvioinnin tarkkuuteen ja se mainittiin 57,14%:ssa artikkeleista.

Heijden, Broasca ja Serebrenik (2018) järjestivät haastatteluja selvittääkseen mahdollisia tietoturvahaasteita ketterissä ohjelmistokehitysprojekteissa. Haastatteluissa tuli ilmi neljä tietoturvahaastetta, jotka esiintyvät ketterissä projekteissa: tietoturvadokumentaation luominen, selkeiden tietoturvavaatimusten laatiminen, tuoteomistajan sitouttaminen tietoturvaan sekä tietoturvatietoisuuden ja osaamisen jakaminen tiimeissä. Vastaajien mukaan joissain tiimeissä ei esimerkiksi ole tietoturvavastaavaa, jonka vuoksi on epäselvää kenen vastuulla tietoturvaan liittyvät toimenpiteet, kuten esimerkiksi tietoturvadokumentaation luominen, on. Haastatteluissa tuli ilmi myös kolme haastetta, jotka koskevat erityisesti suuria ketteriä projekteja: tietoturvakäytäntöjen linjaaminen hajautetuissa järjestelmissä, matalakustannuksisten tietoturvatestaustyökalujen integrointi sekä yhteisymmärryksen kehittäminen tietoturvallisuustoiminnan rooleista ja vastuista. Haastateltavat mainitsivat että hajautetuilla, esimerkiksi globaaleilla, järjestelmillä voi olla useampi tietoturvavastaava, jonka seurauksena tiimit saavat ristiriitaisia pyyntöjä eri tietoturvavastaavilta.

5 Pohdinta

Tässä luvussa vastataan tutkimuskysymyksiin, pohditaan tutkimustulosten merkittävyyttä, arvioidaan tutkimustulosten luotettavuutta sekä esitetään mahdollisia jatkotutkimusaiheita. Aliluvussa 5.1 esitetään vastaukset tutkimuskysymyksiin eli empiirisissä tutkimuksissa havaitut hyödyt ja haasteet. Aliluvussa 5.2 pohditaan tutkimustulosten merkittävyyttä eli mitä hyötyä tutkimustuloksista voi olla. Aliluvussa 5.3 arvioidaan ovatko tutkimustulokset luotettavia ja esitetään mitkä tekijät vaikuttavat tutkimustulosten luotettavuuteen. Lopuksi aliluvussa 5.4 esitetään mahdolliset jatkotutkimusaiheet.

5.1 Tutkimuskysymyksien vastaukset

Katsauksessa pyrittiin vastaamaan seuraaviin tutkimuskysymyksiin:

- *Mitä hyötyjä ketterien menetelmien käytössä on havaittu?*
- *Mitä haasteita ketterien menetelmien käytössä on havaittu?*

Katsaukseen valittujen artikkeleiden perusteella voidaan todeta, että havaittuja hyötyjä on hieman enemmän kuin havaittuja haasteita. Katsauksen artikkeleissa esiintyi yhteensä seitsemän erilaista hyötyä, ja havaittujen hyötyjen kappalemäärä oli yhteensä kolmetoista (aliluku 4.5). Artikkeleissa esiintyi vastaavasti yhteensä seitsemän erilaista haastetta, mutta havaittujen haasteiden kappalemäärä oli sen sijaan yhteensä vain kymmenen (aliluku 4.6). Havaitut hyödyt saivat siis artikkeleissa kolme kappaletta enemmän mainintoja kuin havaitut haasteet.

Selkeästi eniten havaittu hyöty oli ohjelmakoodin laadun paraneminen, joka mainittiin yhteensä neljässä eri artikkelissa (Capiluppi ym. 2007; Schmidt, Ganesha Venkatesha ja Heymann 2014; Hsu ja Lin 2018; Shawky ja Abd-El-Hafiz 2014). Capiluppi ym. (2007) toteavat, että päivittäisten palavereiden ansiosta ohjelmakoodin refaktoroinnista voidaan keskustella aktiivisesti, jonka seurauksena ohjelmakoodin kompleksisuus vähenee ja laatu paranee. Myös Shawky ja Abd-El-Hafiz (2014) toteavat, että ohjelmakoodin kompleksisuus vähenee lyhyiden iteraatioiden ansiosta. Schmidt, Ganesha Venkatesha ja Heymann (2014) mainitsevat artikkelissaan, että pariohjelmoinnin ansiosta ohjelmakoodin modulaarisuus ja ymmär-

rettävyys paranee samalla kun bugien määrä vähenee.

Lopuissa havaituissa hyödyissä ei ollut suuria eroavaisuuksia, koska kaikki ne mainittiin vain yhdessä tai kahdessa artikkelissa. Hyödyt, jotka mainittiin kahdesti olivat: tuottavuus kasvaa (Schmidt, Ganesha Venkatesha ja Heymann 2014; Iqbal, Omar ja Yasin 2019), virheenkäsittely paranee (Schmidt, Ganesha Venkatesha ja Heymann 2014; Hsu ja Lin 2018) ja työntekijöiden tuntemukset kohenevat (Schmidt, Ganesha Venkatesha ja Heymann 2014; Hsu ja Lin 2018). Loput hyödyt, jotka mainittiin vain kerran olivat: vaatimusten määrittely paranee (Kassab 2014), irtisanomiset vähenee (Hsu ja Lin 2018) sekä aikataulun ja kustannusten arviointi tarkentuu (Kassab 2014).

Havaituissa haasteissa ei ollut suuria eroavaisuuksia, koska kaikki haasteet mainittiin vain kerran tai kahdesti. Eniten havaitut, eli kahdesti mainitut, haasteet olivat: kommunikointi (Britto, Mendes ja Börstler 2015; Vithana, Asirvatham ja Johar 2018), dokumentointi (Stettina ja Heijstek 2011; Vithana, Asirvatham ja Johar 2018) ja työmäärän arviointi (Britto, Mendes ja Börstler 2015; Cao 2022). Stettina ja Heijstek (2011) toteavat, että ketterissä projekteissa dokumentaatiota ei tuoteta tarpeeksi, koska niissä keskitytään enemmän välittömään kommunikointiin. Lisäksi Stettina ja Heijstek (2011) toteavat, että sisäisestä dokumentaatiosta ei ole suoranaisesti hyötyä loppuasiakkaalle verrattuna esimerkiksi ohjelmakoodin luomiseen, jonka vuoksi on vaikeaa perustella miksi dokumentaation luomiseen pitäisi käyttää resursseja. Myös Vithana, Asirvatham ja Johar (2018) toteavat, että tehokas kommunikointi ja asianmukainen dokumentointi on projektin onnistumisen kannalta elintärkeää. Heidän toteuttaman kyselyn mukaan kommunikointi ja dokumentointi on haastavaa etenkin globaaleissa ketterissä projekteissa.

Britto, Mendes ja Börstler (2015) toteavat, että työmäärän arviointi on haastavaa ja että suurimmassa osassa projekteista työmäärä aliarvioidaan. Heidän toteuttaman kyselyn mukaan työmäärän arviointi on haastavaa koska esimerkiksi: projektin vaatimukset on dokumentoitu epäselvästi tai virheellisesti, kommunikointi on vaikeaa hajautetuissa tiimieissä kielellisten tai kulttuuristen eroavaisuuksien vuoksi ja koska asiakkaan osallistuminen projektiin ei ole riittävää tai asiakas ei onnistu tehtävien priorisoinnissa. Cao (2022) täydentää, että vaikka työntekijöiden kokemus kasvaa, heidän työmäärän arvioinnin tarkkuus ei parane. Loput havaitut, eli kerran mainitut, haasteet olivat: asiakasosallistuminen (Vithana, Asirvatham

ja Johar 2018), käyttö isoissa projekteissa (Kelle ym. 2015), kustannusten arviointi (Saeed ym. 2021) ja tietoturva (Heijden, Broasca ja Serebrenik 2018).

5.2 Tulosten merkittävyys

Tutkimustuloksista on hyötyä kaikille, jotka työskentelevät ketterien menetelmien parissa tai harkitsee ketterien menetelmien käyttämistä. Koska katsauksen tuloksista tulee ilmi ketteriin menetelmiin liittyviä hyötyjä ja haasteita, kyseisiä tietoja voidaan käyttää apuna erilaisissa tilanteissa, joissa tehdään ketteriin menetelmiin liittyviä päätöksiä. Esimerkiksi jos pohditaan, kannattaako ketteriä menetelmiä käyttää, voidaan tarkastella mitä hyötyjä (aliluku 4.5) menetelmiä käyttämällä saavutetaan ja mitä haasteita (aliluku 4.6) niiden käyttämiseen liittyy.

Katsauksen tulosten perusteella voidaan todeta, että ohjelmakoodin laatu paranee ketteriä menetelmiä käyttämällä. Kyseinen hyöty mainittiin yhteensä neljässä eri artikkelissa (Capiluppi ym. 2007; Schmidt, Ganesha Venkatesha ja Heymann 2014; Hsu ja Lin 2018; Shawky ja Abd-El-Hafiz 2014). Sen vuoksi ketteriä menetelmiä on perusteltua käyttää projekteissa, joissa halutaan parantaa ohjelmakoodin laatua. Tutkimustulosten perusteella voidaan myös todeta, että työmäärän ja kustannusten arviointi ketterissä projekteissa on haastavaa. Työmäärän arvioinnin haasteet mainittiin kahdessa artikkelissa (Britto, Mendes ja Börstler 2015; Cao 2022) ja kustannusten arvioinnin haasteet yhdessä artikkelissa (Saeed ym. 2021). Siksi ketteriä menetelmiä ei välttämättä kannata ottaa käyttöön projekteissa, joissa työmäärä ja kustannukset on tiedettävä täsmällisesti.

Tutkimustuloksista voi myös olla apua ketterien menetelmien käyttäjille. Ketterien menetelmien käyttäjät voivat saada tutkimustuloksista uutta tietoa, jota he voivat hyödyntää ohjelmistokehitysprojekteissa. Kun käyttäjät tiedostavat esimerkiksi ketteriin menetelmiin liittyvät haasteet, haasteista voidaan pyrkiä pääsemään eroon. Tutkimustulosten mukaan esimerkiksi dokumentointi on haastavaa ketterissä projekteissa (Stettina ja Heijstek 2011; Vithana, Asirvatham ja Johar 2018). Käyttäjien tiedostaessa kyseisen haasteen, voidaan projektin dokumentointiin vaatia lisäresursseja dokumentointi haasteen ratkaisemiseksi. Katsauksen tulosten perusteella voidaan myös todeta, että työmäärän ja kustannusten arviointi on vai-

keaa ketteriä menetelmiä käytettäessä (Britto, Mendes ja Börstler 2015; Cao 2022; Saeed ym. 2021). Kun käyttäjät ovat tietoisia arvioinnin haasteista, he voivat tällöin esimerkiksi ottaa käyttöön erilaisia työkaluja työnteon tueksi arvioinnin helpottamiseksi.

Katsauksen tuloksia voidaan myös hyödyntää ketterien menetelmien kehittämisessä. Kun tiedetään mitä haasteita ketteriin menetelmiin liittyy, voidaan haasteita pyrkiä ratkaisemaan menetelmiä kehittäessä. Esimerkiksi Stettina ja Heijstek (2011) mainitsevat, että ketterissä projekteissa ei tuoteta tarpeeksi dokumentaatiota. Dokumentaation määrän kasvattamiseksi ketteriä menetelmiä voitaisiin kehittää esimerkiksi siten, että iteraation loppuun lisättäisiin uusi työvaihe, jonka aikana olemassa olevaa dokumentaatiota täydennettäisiin tai luotaisiin uutta dokumentaatiota kuluvan iteraation tietojen perusteella. Tällöin myös ketterissä projekteissa tuotettaisiin enemmän dokumentaatiota.

5.3 Tulosten luotettavuus

Katsauksen tutkimustuloksia voidaan mielestäni pitää luotettavina, koska artikkeleissa havaitut hyödyt (kuvio 9) ja haasteet (kuvio 10) voidaan perustella ketterien menetelmien teorian ja perusperiaatteiden (aliluku 2.1) avulla. Katsauksen tulosten perusteella voidaan todeta, että ketterien menetelmien ansiosta esimerkiksi ohjelmakoodin laatu ja virheenkäsittely paranee sekä tuottavuus kasvaa (Capiluppi ym. 2007; Schmidt, Ganesha Venkatesha ja Heymann 2014; Hsu ja Lin 2018; Shawky ja Abd-El-Hafiz 2014; Iqbal, Omar ja Yasin 2019). Ketterien menetelmien perusperiaatteisiin kuuluu toimivan ohjelmakoodin ja ohjelmiston tuottaminen, jonka vuoksi siihen käytetään enemmän resursseja. Kun ohjelmakoodin kehittämiseen käytetään enemmän resursseja, on loogista että ohjelmakoodin laatu paranee. Ohjelmakoodin laadun parantuessa virheiden määrä vähenee, joka vuorostaan johtaa virheenkäsittelyn paranemiseen. Koska ohjelmakoodin luomiseen käytetään paljon resursseja, koodin määrä kasvaa, joka voidaan kokea myös tuottavuuden kasvuna.

Tutkimustulosten mukaan myös työntekijöiden tuntemukset kohenivat ja irtisanomiset vähenivät ketteriä menetelmiä käyttämällä (Schmidt, Ganesha Venkatesha ja Heymann 2014; Hsu ja Lin 2018). Ketterien menetelmien perusperiaatteiden mukaisesti yksilöt ja vuorovaikutukset ovat prosesseja ja työkaluja tärkeämpiä. Koska ketterissä menetelmissä keskitytään yksi-

löihin ja yksilöiden välisiin vuorovaikutuksiin, tiimin yhteisymmärrys ja yhteishenki pääsee kehittymään. Yhteisymmärryksen ja tiimin hengen parantuessa myös työntekijöiden tuntemukset kohenevat, joka vuorostaan johtaa irtisanomisten vähenemiseen. Tutkimustuloksissa havaittiin myös vaatimusten määrittelyn paraneminen (Kassab 2014), jota voidaan perustella esimerkiksi sillä, että ketterissä menetelmissä säännöllistä yhteistyötä asiakkaan kanssa pidetään tärkeänä. Kun yhteistyö asiakkaan kanssa on säännöllistä, asiakkaalta saadaan säännöllisesti palautetta päätöksistä ja vaatimuksista, jonka ansiosta vaatimusten määrittely on helpompaa.

Katsauksessa ketterien menetelmien haasteiksi mainittiin esimerkiksi työmäärän ja kustannusten arviointi (Britto, Mendes ja Börstler 2015; Cao 2022; Saeed ym. 2021). Koska ketterät menetelmät sallivat jatkuvat muutokset ja korostavat muutoksiin reagoinnin tärkeyttä, myös työmäärä ja kustannukset voivat muuttua jatkuvasti projektin aikana. Jatkuvien muutosten vuoksi todellisen työmäärän ja kustannusten arviointi on vaikeaa.

Ketterien projektien haasteiksi koettiin myös dokumentointi (Stettina ja Heijstek 2011; Vithana, Asirvatham ja Johar 2018) ja tietoturva (Heijden, Broasca ja Serebrenik 2018). Ketterät menetelmät luotiin alunperin muun muassa siksi, että voitaisiin vältellä täydellisen alkudokumentaation luomista. Ketterien menetelmien perusperiaatteiden mukaisesti kattava dokumentaatio ei ole yhtä tärkeää kuin esimerkiksi toimiva ohjelmisto. Ketterissä projekteissa dokumentointiin ei siis käytetä tarpeeksi resursseja, jolloin dokumenttien määrä jää vähäiseksi ja dokumentointi koetaan haastavaksi. Dokumentointiin liittyvät haasteet vaikuttavat myös tietoturvaan. Heijden, Broasca ja Serebrenik (2018) mainitsevat, että yksi ketterissä projekteissa esiintyvistä tietoturvahaasteista on tietoturvadokumentaation luominen.

Tutkimustulosten mukaan ketterissä projekteissa kommunikointi voi olla haastavaa ja sen lisäksi ketteriä menetelmiä voi olla haastavaa käyttää isoissa projekteissa (Britto, Mendes ja Börstler 2015; Vithana, Asirvatham ja Johar 2018; Kelle ym. 2015). Jotkut yksilöt voivat kokea sosiaalisesti kuormittavana sen, että ketterissä menetelmissä keskitytään vahvasti yksilöihin ja yksilöiden välisiin vuorovaikutuksiin. Lisäksi jos tiimeissä ei synny selkeää yhteisymmärrystä tai hyvää yhteishenkeä, ketterien menetelmien vaatima jatkuva kommunikointi voi olla haastavaa. Kommunikaatiohaasteet näkyvät myös isoissa ketterissä projekteissa. Isoissa projekteissa tiimit voivat olla hajautettuja, jolloin on mahdollista että tiimin jäsenet

sijaitsevat esimerkiksi eri aikavyöhykkeillä tai puhuvat eri kieliä. Tällöin ketterille menetelmille tyypillistä jatkuvaa kommunikointia ei välttämättä pääse syntymään ja kommunikointi on haastavaa.

Ketterissä projekteissa asiakasosallistuminen voi myös olla haastavaa (Vithana, Asirvatham ja Johar 2018). Ketterät menetelmät edellyttävät säännöllistä yhteistyötä asiakkaan kanssa säännöllisen palautteen saamiseksi. Joillekin asiakkaille projektiin sitoutuminen ei kuitenkaan välttämättä sovi, jolloin asiakkaan kanssa ei ole säännöllistä yhteistyötä. Jos asiakkaalta ei saada säännöllisesti palautetta, voi päätöksenteko ja vaatimusten määrittely ketterissä projekteissa olla haastavaa.

Vaikka tutkimustulokset voidaan perustella ketterien menetelmien teorian ja perusperiaatteiden avulla, on silti useita tekijöitä jotka vaikuttavat negatiivisesti tutkimustulosten luotettavuuteen. Katsauksen tiedonkeruu ja artikkeleiden rajaus on toteutettu vain yhden henkilön, tutkielman laatijan, toimesta. Sen vuoksi on mahdollista, että tutkielmasta on jäänyt pois joitain artikkeleita, jotka olisivat olleet tutkielman kannalta olennaisia. Tiedonkeruun ja artikkeleiden rajauksen vaikutusta luotettavuuteen on kuitenkin pyritty minimoimaan esittämällä tiedonkeruustrategia sekä valintakriteereiden soveltaminen artikkeleihin avoimesti ja selkeästi (luku 3).

Tutkielman aineisto on myös jäänyt suppeaksi, koska tiedonkeruu on toteutettu vain tutkielman laatijan toimesta. Koska tutkielman aineisto on suppea, myös tutkielmassa mainittuja erilaisia hyötyjä (kuvio 9) ja haasteita (kuvio 10) on vähän. Jos aineistoa olisi ollut enemmän, olisi sieltä voinut löytyä uusia hyötyjä tai haasteita, joita ei ole tutkielmassa mainittu. Jotta tutkielman aineistoa voitaisiin pitää täysin luotettavana, tulisi aineistoa olla enemmän ja se pitäisi validoida useamman henkilön toimesta. Pro gradu -tutkielman puitteissa tiedonkeruu ja validointi kuitenkin suoritettiin vain tutkielman laatijan toimesta.

On kuitenkin myös mahdollista, että ketteristä menetelmistä ei välttämättä olisi löytynyt enempää empiirisesti tutkittuja hyötyjä tai haasteita ainakaan tutkielmassa käytetyistä tietokannoista. Tiedonkeruuprosessin alussa kävi jo nopeasti ilmi, että vaikka on olemassa paljon empiirisiä tutkimuksia, joissa ketterät menetelmät ovat jollain tapaa mukana, on silti vaikeaa löytää empiirisiä tutkimuksia joissa tutkittaisiin itse menetelmien hyötyjä tai haasteita.

Tutkimustulosten kategorisointi ja yhtenäistäminen on myös voinut laskea tutkimustulosten luotettavuutta. Koska kategorisointi ja yhtenäistäminen toteutettiin vain tutkielman laatijan toimesta, on mahdollista että kyseiset toimenpiteet olisi voitu suorittaa eri tavalla, jolloin tutkimusten havaintojen kappalemäärät olisi myös erilaiset. Joitain tutkimustuloksia oli kuitenkin pakko kategorisoida ja yhtenäistää paremman kokonaiskuvan saavuttamiseksi. Esimerkiksi tutkielmassa yleisin havaittu hyöty *ohjelmakoodin laatu paranee*, on tutkielman laatijan luoma kategoria, jonka alle on kirjattu useita erilaisia tutkimuksissa esiintyneitä ohjelmakoodin laatuun positiivesti vaikuttavia tekijöitä.

Jos jokainen ohjelmakoodin laatuun positiivesti vaikuttava tekijä olisi kirjattu sellaisenaan ilman tulosten yhtenäistämistä, tutkimustuloksista olisi vaikeaa tehdä johtopäätöksiä. Vaikka tutkimustuloksia yhtenäistettiin, havaituissa hyödyissä ja haasteissa ei silti esiintynyt suurta hajontaa. Hajontaa ei päässyt syntymään mahdollisesti suppean tutkimusaineiston vuoksi. Aineiston määrän kasvaessa myös havaintojen kappalemäärät kasvaa, jolloin on mahdollista että jotkut havainnot nousisivat selkeämmin esiin ja havaintojen välille muodostuisi hajontaa.

5.4 Jatkotutkimusaiheita

Yksi jatkotutkimusaihe voisi olla vastaavanlaisen systemaattisen kirjallisuuskatsauksen toteuttaminen tai nykyisen katsauksen täydentäminen suuremmalla aineistolla. Uudessa katsauksessa artikkeleita voitaisiin hakea myös sellaisista tieteellisistä tietokannoista, joita ei tässä katsauksessa käytetty. Tämän katsauksen artikkelit haettiin “IEEE Xplore Digital Library” (2024), “ACM Digital Library” (2024) ja “SpringerLink” (2024) tietokannoista, joten uudessa katsauksessa artikkeleita voitaisiin lisäksi hakea esimerkiksi Scopus- ja ScienceDirect-tietokannoista. Uudessa katsauksessa voitaisiin myös käyttää erilaisia hakulauseita tai hakea myös muiden tieteenalojen artikkeleita hakutulosten määrän kasvattamiseksi. Tässä katsauksessa esimerkiksi SpringerLink-tietokannasta haettiin vain tietojenkäsittelytieteen alan artikkeleita. Uusista artikkeleista voisi esimerkiksi löytyä sellaisia uusia hyötyjä tai haasteita, joita ei tässä katsauksessa löydetty. Vaikka uusista artikkeleista ei löytyisikään uusia havaintoja, voisi artikkeleiden havainnot vahvistaa tässä katsauksessa mainittuja hyötyjä tai haasteita.

Toinen jatkotutkimusaihe voisi olla tässä katsauksessa esitettyjen havaintojen syvempi tut-

kiminen tai vahvistaminen. Etenkin havainnot, jotka esiintyivät vain yhdessä tai kahdessa artikkelissa, olisi hyvä vahvistaa ja tutkia tarkemmin uuden tutkimuksen avulla. Esimerkiksi Heijden, Broasca ja Serebrenik (2018) mainitsevat, että ketterät menetelmät aiheuttavat tietoturvahaasteita. Tietoturvahaasteet mainitaan kuitenkin vain yhdessä katsauksen artikkeleista, joten olisi tärkeää tutkia väitteen paikkaansapitävyyttä tai sitä, mistä kyseinen haaste johtuu ja mitä toimenpiteitä ketterien projektien tietoturvan parantamiseksi voitaisiin tehdä. Toinen mielenkiintoinen havainto jatkotutkimusta varten on esimerkiksi irtisanomisten väheneminen ketterien menetelmien ansiosta. Hsu ja Lin (2018) toteavat, että Titansoft-yrityksen irtisanomisten määrä väheni kun ketterät menetelmät otettiin käyttöön. Havainnon vahvistamiseksi irtisanomisten määrän muuttumista olisi hyvä tutkia myös muissa yrityksissä, joissa on otettu käyttöön ketteriä menetelmiä. Irtisanomisia tutkittaessa on kuitenkin olennaista selvittää, johtuvatko muutokset ketteristä menetelmistä vai onko yrityksessä tapahtunut samaan aikaan jotain muuta merkittävää kun menetelmä on otettu käyttöön, joka voisi selittää irtisanomisten määrän muuttumisen.

6 Yhteenveto

Tämän tutkielman tavoitteena oli selvittää, mitä empiirisesti tutkittuja hyötyjä tai haasteita ketteriin menetelmiin liittyy ketterien menetelmien olettamuksien vahvistamiseksi. Tutkielman systemaattista kirjallisuuskatsausta varten artikkeleita haettiin kolmesta keskeisestä tietokannasta. Tietokannoista löytyi yhteensä 113 katsaukseen sopivaa artikkelia, jonka jälkeen artikkeleihin sovellettiin aliluvussa 3.4 mainittuja valintakriteereitä. Valintakriteereiden soveltamisen jälkeen jäljelle jäi enää 15 artikkelia. Seuraavaksi kyseiset artikkelit luettiin läpi ja varmistettiin, että ne vastaavat tutkimuskysymyksiin. Artikkelien lukemisen jälkeen kaksi artikkelia karsittiin vielä pois, jolloin katsaukseen jäi lopulta yhteensä 13 artkkelia.

Katsauksen tutkimustulosten perusteella voidaan todeta, että ketteriin menetelmiin liittyy paljon erilaisia hyötyjä ja haasteita. Katsauksen artikkeleista löydettiin seitsemän eri hyötyä ja seitsemän eri haastetta liittyen ketteriin menetelmiin. Vaikka osa hyödyistä ja haasteista esiintyi vain yhdessä artikkelissa, voidaan tutkimustuloksista silti tehdä jotain johtopäätöksiä. Yleisin havainto, joka erottui selkeästi muista havainnoista, oli ohjelmakoodin laadun paraneminen ketterien menetelmien ansiosta. Ohjelmakoodin laadun paraneminen mainittiin yhteensä neljässä eri artikkelissa (Capiluppi ym. 2007; Schmidt, Ganesha Venkatesha ja Heymann 2014; Hsu ja Lin 2018; Shawky ja Abd-El-Hafiz 2014) kun seuraavaksi yleisimmät havainnot saivat vain kaksi mainintaa. Tutkimustuloksista ei selkeästi erottunut muita havaintoja, koska kaikki muut havainnot mainittiin vain yhdessä tai kahdessa artikkelissa.

Tutkimustuloksia voidaan pitää merkittävänä, koska niistä on hyötyä kaikille ketterien menetelmien parissa työskenteleville henkilöille. Tuloksia voidaan esimerkiksi käyttää apuna pohdittaessa ketterien menetelmien käyttöönottoa. Tutkimustuloksista on myös apua menetelmien käyttäjille ja kehittäjille. Käyttäjät voivat esimerkiksi muuttaa toimintamallejaan tulosten haasteiden välttämiseksi kun taas ketterien menetelmien kehittäjät voivat muokata kehitysprosessia haasteiden ratkaisemiseksi.

Tutkimustuloksia voidaan pitää luotettavina, koska ne ovat perusteltavissa ketterien menetelmien perusperiaatteiden avulla. Ohjelmakoodin laadun paraneminen ketterien menetelmien ansiosta voidaan esimerkiksi perustella sillä, että menetelmien perusperiaatteisiin kuuluu toi-

mivan ohjelmakoodin tuottaminen. Toimivan ohjelmakoodin ollessa keskiössä, siihen myös käytetään enemmän resursseja, jolloin ohjelmakoodin laatu todennäköisemmin paranee. On myös kuitenkin useita tekijöitä, jotka vaikuttavat negatiivisesti tutkimustulosten luotettavuuteen. Katsauksen tiedonkeruu ja artikkeleiden rajaus on esimerkiksi toteutettu vain yhden henkilön toimesta, jonka vuoksi tutkielman aineisto on voinut jäädä suppeaksi. Tutkimustulosten kategorisointi ja yhtenäistäminen kokonaiskuvan saavuttamiseksi on myös voinut vaikuttaa havaintojen määrään.

Mahdollinen jatkotutkimusaihe on vastaavanlaisen katsauksen toteuttaminen uuden aineiston löytämiseksi. Uudessa katsauksessa artikkeleita voitaisiin esimerkiksi hakea uusista tietokannoista tai käyttää erilaisia hakulauseita artikkeleiden löytämiseksi. Toinen jatkotutkimusaihe on tässä katsauksessa esitettyjen havaintojen syvempi tutkiminen. Etenkin vähemmän esiintyneitä havaintoja olisi hyvä tutkia tarkemmin havaintojen vahvistamiseksi.

Lähteet

"ACM Digital Library". 2024. Viitattu 12. helmikuuta 2024. https://dl.acm.org/.

Ahmad, Muhammad Ovais, Jouni Markkula ja Markku Oivo. 2013. "Kanban in software development: A systematic literature review". Teoksessa *2013 39th Euromicro Conference on Software Engineering and Advanced Applications,* 9–16. https://doi.org/10.1109/SEAA.2013.28.

Beck, K. 1999. "Embracing change with extreme programming". *Computer* 32 (10): 70–77. ISSN: 1558-0814. https://doi.org/10.1109/2.796139.

Britto, Ricardo, Emilia Mendes ja Jürgen Börstler. 2015. "An Empirical Investigation on Effort Estimation in Agile Global Software Development". Teoksessa *2015 IEEE 10th International Conference on Global Software Engineering,* 38–45. https://doi.org/10.1109/ICGSE.2015.10.

Cao, Lan. 2022. "Estimating Efforts for Various Activities in Agile Software Development: An Empirical Study". *IEEE Access* 10:83311–83321. https://doi.org/10.1109/ACCESS.2022.3196923.

Capiluppi, A., J. Fernandez-Ramil, J. Higman, H. C. Sharp ja N. Smith. 2007. "An Empirical Study of the Evolution of an Agile-Developed Software System". Teoksessa *Proceedings of the 29th International Conference on Software Engineering,* 511–518. ICSE '07. USA: IEEE Computer Society. ISBN: 0769528287. https://doi.org/10.1109/ICSE.2007.14.

Chowdhury, Ashraf Ferdouse ja Mohammad Nazmul Huda. 2011. "Comparison between Adaptive Software Development and Feature Driven Development". Teoksessa *Proceedings of 2011 International Conference on Computer Science and Network Technology,* 1:363–367. https://doi.org/10.1109/ICCSNT.2011.6181977.

Heijden, Amber van der, Cosmin Broasca ja Alexander Serebrenik. 2018. “An empirical perspective on security challenges in large-scale agile software development”. Teoksessa *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement,* 1–4. ESEM ’18. Oulu, Finland: Association for Computing Machinery. ISBN: 9781450358231. https://doi.org/10.1145/3239235.3267426.

Highsmith, J. ja A. Cockburn. 2001. “Agile software development: the business of innovation”. *Computer* 34 (9): 120–127. ISSN: 1558-0814. https://doi.org/10.1109/2.947100.

Hoda, Rashina, Norsaremah Salleh ja John Grundy. 2018. “The Rise and Evolution of Agile Software Development”. *IEEE Software* 35 (5): 58–63. ISSN: 1937-4194. https://doi.org/10.1109/MS.2018.290111318.

Hsu, Hwai-Jung ja Yves Lin. 2018. “How Agile Impacts a Software Corporation: An Empirical Study”. Teoksessa *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC),* 02:20–25. https://doi.org/10.1109/COMPSAC.2018.10197.

“IEEE Xplore Digital Library”. 2024. Viitattu 12. helmikuuta 2024. https://ieeexplore.ieee.org/Xplore/home.jsp.

Iqbal, Javed, Mazni Omar ja Azman Yasin. 2019. “An Empirical Analysis of the Effect of Agile Teams on Software Productivity”. Teoksessa *2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET),* 1–8. https://doi.org/10.1109/ICOMET.2019.8673413.

Kassab, Mohamad. 2014. “An Empirical Study on the Requirements Engineering Practices for Agile Software Development”. Teoksessa *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications,* 254–261. https://doi.org/10.1109/SEAA.2014.77.

Kelle, Evelyn van, Per van der Wijst, Aske Plaat ja Joost Visser. 2015. “An empirical study into social success factors for agile software development”. Teoksessa *Proceedings of the Eighth International Workshop on Cooperative and Human Aspects of Software Engineering,* 77–80. CHASE ’15. Florence, Italy: IEEE Press. https://doi.org/10.1109/CHASE.2015.24.

Petersen, Kai, Sairam Vakkalanka ja Ludwik Kuzniarz. 2015. “Guidelines for conducting systematic mapping studies in software engineering: An update”. *Information and Software Technology* 64:1–18. ISSN: 0950-5849. https://doi.org/https://doi.org/10.1016/j.infsof.2015.03.007.

Poppendieck, Mary ja Michael A. Cusumano. 2012. “Lean Software Development: A Tutorial”. *IEEE Software* 29 (5): 26–32. ISSN: 1937-4194. https://doi.org/10.1109/MS.2012.107.

Saeed, Syed Abu, Junaid Ali Khan, Saira Naeem ja Saif-Ur-Rehman Khan. 2021. “An Empirical Investigation on Cost Estimation Challenges in Agile Software Development (ASD) Context”. Teoksessa *2021 International Conference on Frontiers of Information Technology (FIT),* 188–193. https://doi.org/10.1109/FIT53504.2021.00043.

Schmidt, Christoph Tobias, Srinivasa Ganesha Venkatesha ja Juergen Heymann. 2014. “Empirical insights into the perceived benefits of agile software engineering practices: a case study from SAP”. Teoksessa *Companion Proceedings of the 36th International Conference on Software Engineering,* 84–92. ICSE Companion 2014. Hyderabad, India: Association for Computing Machinery. ISBN: 9781450327688. https://doi.org/10.1145/2591062.2591189.

Shawky, Doaa M. ja Salwa K. Abd-El-Hafiz. 2014. “The impact of agile approaches on software quality attributes an empirical study”. Teoksessa *2014 9th International Conference on Software Paradigm Trends (ICSOFT-PT),* 49–57. Viitattu 15. helmikuuta 2024. https://ieeexplore.ieee.org/document/7292574.

“SpringerLink”. 2024. Viitattu 12. helmikuuta 2024. https://link.springer.com/.

Srivastava, Apoorva, Sukriti Bhardwaj ja Shipra Saraswat. 2017. “SCRUM model for agile methodology”. Teoksessa *2017 International Conference on Computing, Communication and Automation (ICCCA),* 864–869. https://doi.org/10.1109/CCAA.2017.8229928.

Stettina, Christoph Johann ja Werner Heijstek. 2011. “Necessary and neglected? an empirical study of internal documentation in agile software development teams”. Teoksessa *Proceedings of the 29th ACM International Conference on Design of Communication,* 159–166. SIGDOC ’11. Pisa, Italy: Association for Computing Machinery. ISBN: 9781450309363. https://doi.org/10.1145/2038476.2038509.

Uludağ, Ömer, Abheeshta Putta, Maria Paasivaara ja Florian Matthes. 2021. “Evolution of the Agile Scaling Frameworks”. Teoksessa *Agile Processes in Software Engineering and Extreme Programming,* toimittanut Peggy Gregory, Casper Lassenius, Xiaofeng Wang ja Philippe Kruchten, 123–139. Cham: Springer International Publishing. ISBN: 978-3-030-78098-2.

Williams, L. ja A. Cockburn. 2003. “Agile software development: it’s about feedback and change”. *Computer* 36 (6): 39–43. ISSN: 1558-0814. https://doi.org/10.1109/MC.2003.1204373.

Vithana, V. N., D. Asirvatham ja M.G.M. Johar. 2018. “An Empirical Study on Using Agile Methods in Global Software Development”. Teoksessa *2018 18th International Conference on Advances in ICT for Emerging Regions (ICTer),* 150–156. https://doi.org/10.1109/ICTER.2018.8615505.