Kustannustehokkaat ja ketterät ohjelmistokehitysmenetelmät uusissa ohjelmistoyrityksissä: tarkastelussa Scrum, Lean-ohjelmistokehitys, Kanban ja MVP

Turun yliopisto
Tietotekniikan laitos
TkK-tutkielma
Tietotekniikka
Toukokuu 2025
Santeri Lukka

TURUN YLIOPISTO
Tietotekniikan laitos

Santeri Lukka: Kustannustehokkaat ja ketterät ohjelmistokehitysmenetelmät uusissa ohjelmistoyrityksissä: tarkastelussa Scrum, Lean-ohjelmistokehitys, Kanban ja MVP

TkK-tutkielma, 28 s.
Tietotekniikka
Toukokuu 2025

Uudet ohjelmistoja kehittävät yritykset toimivat usein nopeasti muuttuvassa ja epävarmassa markkinaympäristössä. Kustannustehokkaita ja ketteriä menetelmiä on kehitetty vastaamaan näiden yritysten vaatimuksiin.
Tässä kandidaatintutkielmassa tarkastellaan kustannustehokkaita ja ketteriä ohjelmistokehitysmenetelmiä uusien ohjelmistoja kehittävien yritysten näkökulmasta. Tavoitteena oli selvittää, miten menetelmät tukevat uusien yritysten toimintaa ja millaisia haasteita niiden soveltamiseen liittyy. Tutkimuskohteina tässä tutkimuksessa oli neljä menetelmää: Scrum, Lean-ohjelmistokehitys, Kanban ja MVP (Minimum Viable Product). Tutkimus toteutettiin kirjallisuuskatsauksena.
Tutkielman aiheena oli tutkia edellä mainittuja menetelmiä, tarkastella niiden hyötyjä ja haasteita sekä analysoida ja pohtia niiden mahdollisia käyttökohteita ja yhteiskäyttöä.
Scrum tuo projektille selkeän rakenteen, ja se soveltuu kokeneille tiimeille monimutkaisissa projekteissa. Se voi kuitenkin vaatia lisäresursseja. Lean-ohjelmistokehityksen avulla pyritään parempaan tehokkuuteen ja kustannussäästöihin. Sen periaatteiden soveltaminen voi kuitenkin ohjelmistojen ja tietotyön aineettoman luonteen vuoksi olla haastavaa. Kanban auttaa työn visualisoinnissa ja reagoi hyvin muuttuviin vaatimuksiin, mutta edellyttää hyvää viestintää. MVP mahdollistaa tuotteen nopean lanseeraamisen ja testaamisen markkinoilla, mutta siihen voi liittyä riski liian laajasta tai suppeasta ensiversiosta.
Tutkielman päätelmänä on, että valitut menetelmät tukevat kukin eri tavoin uusien ohjelmistoyritysten tarpeita, ja niitä voidaan käyttää sekä yksittäin että yhdistettynä yrityksen tarpeiden ja tilanteen perusteella. Menetelmän valinnan tulisi perustua yrityksen kontekstiin, tavoitteisiin ja resursseihin. Onnistunut menetelmän soveltaminen edellyttää huolellista suunnittelua, koulutusta ja jatkuvaa kehitystä.

Asiasanat: Scrum, Lean-ohjelmistokehitys, Kanban, MVP, Ketterät menetelmät

Sisällys

1 Johdanto

Nykypäivän nopeatempoisessa ja kilpailukeskeisessä liiketoimintaympäristössä uusilla ohjelmistoyrityllä on erilaisia haasteita. Merkittävimpiin haasteisiin lukeutuu tarve kehittää laadukkaita ja luotettavia ohjelmistoja nopeasti ja edullisesti mukautumiskyvystä tinkimättä [1]. Kustannustehokkaat ja ketterät ohjelmistokehitysmenetelmät tarjoavat siihen loistavia mahdollisuuksia, joiden avulla resurssien optimointi, joustavuus ja asiakastarpeisiin reagointi tulevat saavutettavimmiksi, etenkin uusissa ohjelmistoyrityksissä.

Tässä työssä tutkitaan kustannustehokkaita ja ketteriä menetelmiä, joita uusilla ohjelmistoyrityksillä on käytettävissään. Menetelmiä on lukuisia ja siksi tähän työhön on valittu tarkasteluun Scrum, Lean-ohjelmistokehitys, Kanban ja pienin toimiva tuote. Muut menetelmät, kuten XP (Extreme Programming) on laajuussyistä rajattu tutkimuksen ulkopuolelle [2] [3]. Tutkielma toteutetaan kirjallisuuskatsauksena. Työssä ei ainoastaan keskitytä startup-yrityksiin, vaan työssä käsitellään yleisesti uusia ohjelmistoalan yrityksiä. Työn tutkimuskysymykset ovat:

- TK1: Mitkä ovat keskeisimmät menetelmät ja käytännöt, joilla uudet ohjelmistoyritykset voivat toteuttaa kustannustehokasta ja ketterää ohjelmistokehitystä?

- TK2: Miten nämä ketterät ohjelmistokehitysmenetelmät tukevat uusien ohjelmistoyritysten toimintaa ja kehitysprojekteja?

- TK3: Mitkä ovat kustannustehokkaan ja ketterän ohjelmistokehityksen keskeisimmät haasteet uusissa ohjelmistoyrityksissä?

Tiedonhakua on suoritettu Web of Science-, IEEE Xplore-, ACM, Turun yliopiston Volter sekä Google Scholar -tietokannoista. Tiedonhaku on suoritettu käyttämällä hakulauseita:

	Hakulauseke
•	(cost-eff* OR cost eff* OR “cost reduction”) AND ((agile OR lean) AND software develop*) AND (new OR small) AND (compan* OR team*)
•	"Software engineering"AND (New AND (Team? OR Busines* OR Enterprise* OR Company*))
•	Scrum AND Software
•	Kanban AND Software AND Startup
•	MVP OR "Minimum Viable Product"AND Software
•	Lean Software Development

Taulukko 1.1: Hakulausekkeet

Aineiston valikoinnissa kiinnitettiin ensin huomiota tulosten otsikkoon sekä hakusanan palauttaneeseen kohtaan. Otsikkotarkastelun jälkeen aineisto joko valittiin tarkempaan tarkasteluun tai rajattiin pois. Tarkemmassa tarkastelussa arvioitiin ensin aineiston tiivistelmän sisältöä, josta epäsopivat rajattiin pois, jonka jälkeen siirryttiin koko tekstin arviointiin.

Tutkielman luvussa 2 esitellään valitut menetelmät Scrum, Lean-ohjelmistokehitys, Kanban ja MVP sekä perehdytään lyhyesti niiden historiaan. Menetelmien esittelyssä keskitytään niiden keskeisimpiin piirteisiin ja käytäntöihin. Luvussa käsitellään myös kunkin menetelmän hyötyjä ja haasteita. Luvussa 3 analysoidaan ja pohditaan menetelmien sopivuutta uusille ohjelmistoalan yrityksille sekä keskutellaan eri näkökulmista ja toiminnallisuuksista menetelmien avulla. Tutkielman viimeisessä luvussa 4 tuodaan esille tutkielman havainnot ja vastataan työn tutkimuskysymyksiin.

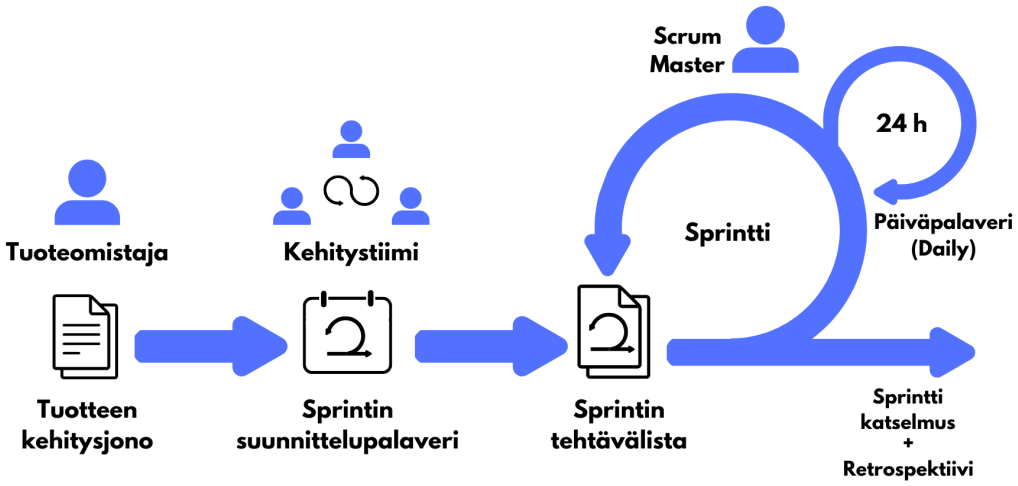
2 Kustannustehokkaat ja ketterät menetelmät

Ketterä ohjelmistokehitys (engl. agile software development) on nippu ohjelmistotuotannon projekteissa käytettäviä menetelmiä. Niiden tarkoituksena on kehittää ohjelmistoja joustavasti ja iteratiivisesti. [4] Ketterä ohjelmistokehitys sai alkunsa 1900- ja 2000- luvun vaihteessa, kun 17 ohjelmistokehittäjää laativat vuonna 2001 Ketterän manifestin [5] (engl. Agile Manifesto). Manifesti toimii ketterän ohjelmistokehityksen perustana ja määrittelee neljä pääarvoa ja 12 periaatetta ketterälle ohjelmistokehitykselle. Manifestin arvoissa ja periaatteissa korostetaan joustavuutta, yhteistyötä ja toimivien ohjelmistojen tehokasta tuottamista ja sitä, kuinka ihmiset ja vuorovaikutus, toimivat ohjelmistot, asiakasyhteistyö sekä sopeutuminen muutoksiin asetetaan vanhojen, jäykkien ja raskaiden mallien edelle. [5]

Uudet ohjelmistoalan yritykset toimivat usein ympäristössä, jossa epävarmuus ja nopea muutos ovat arkipäivää [1] [6]. Ketterän manifestin periaatteiden mukaisesti ketterän kehityksen menetelmät ovat erityisen tärkeitä uusille, kasvuhakuisille ja joustaville yrityksille, jotka pyrkivät vastaamaan nopeasti ja kustannustehokkaasti markkinoiden sekä asiakkaiden alati muuttuviin tarpeisiin.

2.1 Scrum

Scrum on yksi keskeisimmistä ja laajimmin käytetyistä ketteristä menetelmistä. Se on tuotekehitykseen soveltuva viitekehys, joka on alkujaan kehitetty erityisesti ohjelmistotuotantoa varten. Termi Scrum on peräisin rugbysta, ja se viittaa yhteistä tavoitetta kohti etenevään joukkueeseen. Scrumin viitekehyksessä yksi monitaitoinen tiimi suorittaa kehitysprosessia alusta loppuun tiiviissä yhteistyössä. Tuotekehitysprosessi jaetaan pienempiin kehitysjaksoihin, jotka toteutetaan sykleittäin toistavasti ja lisäävästi (engl. iterative-incremental). [7] [8]



Kuva 2.1: Havainnekuva Scrumin toimintaperiaatteesta

Tuotteen kehitysprosessi alkaa tavoitteiden, ominaisuuksien ja vaatimuksen suunnittelulla [8]. Scrumissa tämän suunnittelun perusteella tuotteelle luodaan **tuotteen kehitysjono** (engl. Product Backlog), kuvassa 2.1 vasemmalla. Tuotteen kehitysjono on lista kaikista tehtävistä, ominaisuuksista ja parannuksista, joita tuotteeseen tullaan tekemään. Tätä listaa päivitetään projektin edetessä ja mahdollisten parannusehdotusten perusteella. **Tuoteomistaja** (engl. Product Owner) hallinnoi tätä

listaa. [7] [9]. Tuoteomistaja on vastuussa työn tuloksena syntyvän tuotteen arvon maksimoimisesta [7].

Scrumissa on kolme pääroolia, **Scrum Master**, **tuoteomistaja** (engl. Product Owner) ja **kehitystiimi** (engl. Development Team), kuva 2.1. Näiden kolmen pääroolin yhteistermi on Scrum-tiimi [7]. Scrum Master ohjaa ja johtaa Scrum-tiimiä. Hän on vastuussa Scrumin mukaisen toiminnan noudattamisesta, koko Scrum-tiimin yhteistoiminnasta sekä kehitystiimin tehokkuudesta. Hän parantaa Scrum-tiimin tuottavuutta ohjaamalla kehitystiimin käytäntöjä ja poistamalla edistymistä haittaavia haasteita ja esteitä. Scrum Master on yhteydessä kehitystiimiin, tuoteomistajaan, ympäröivään organisaatioon sekä projektin sidosryhmiin. Hän johtaa kokouksia, hallinnoi sprintin tehtävälistaa, seuraa edistymistä ja puuttuu esteisiin. [7] [8] [9]. Kehitystiimi on itseohjautuva monialainen tiimi, joka tekee projektin varsinaisen kehitystyön [7]. Kehitystiimi määrittelee itse miten työ toteutetaan ja miten työaika jaetaan, kunhan tavoite saavutetaan [7] [8]. Scrum-tiimit koostuvat yleensä enintään 10 jäsenestä [8]. Suuremmissa projekteissa useat Scrum-tiimit voivat tehdä yhteistyötä.[8] [9]

Tuotekehitysprosessin suunnitteluvaiheen jälkeen alkaa projektin varsinainen kehitysvaihe. Scrumissa tuotekehitysprojektia suoritetaan iteratiivisesti ja inkrementaalisesti. **Sprintti** (engl. Sprint), kuvassa 2.1 oikealla, on itse Scrumin sydän. Se on ennalta aikarajattu iteraatio, jossa tietylle aikajaksolle valitut ominaisuudet toteutetaan. [7] Tyypillisesti yksi sprintti kestää 1-4 viikkoa [8]. Sprinttejä toistetaan kunnes projekti saavuttaa halutun lopputuleman. Sprintin aikana keskitytään tuottamaan ennalta määritellyt toiminnallisuudet tuotekokonaisuuteen. [7] [8] [9].

Sprintin alussa pidetään **sprintin suunnittelupalaveri** (engl. Sprint Planning), kuva 2.1 Se aloittaa sprintin ja siinä määritellään kyseisessä sprintissä tehtävä työ. Tuoteomistaja varmistaa, että keskeiset tuotteen tehtävälistan kohteet ovat valmiina palaveria varten ja pyrkii sovittamaan ne yhteen tuotetavoitteen kanssa. Sprintin

suunnittelupalaverin päätteeksi on selvillä mm. sprintin tavoite ja sprinttiin valittujen tuotteenkehitysjonon osat. [7] [9]

Sprintin tehtävälista (engl. Sprint Backlog) on reaaliaikainen lista niistä tehtävistä, jotka Scrum-tiimi on valinnut sprintin suunnittelupalaverissa tehtäväksi kyseisessä sprintissä. Sprintin tehtävälista sisältää kyseisen Sprintin tavoitteen, sprinttiin valitut tuotteen kehitysjonon kohdat sekä suunnitelman seuraavan valmiin askeleen tuottamiseksi. [7] [9]

Sprintin aikana pidetään yleensä lyhyitä palavereja. **Päivittäinen palaveri** (engl. daily) kestää n. 15 minuuttia, ja siinä käsitellään projektin edistymistä tavoitetta kohti.[7] [9] Näitä palavereja pidetään päivittäin koko sprintin elinkaaren ajan. Näissä päivittäisissä palavereissa käydään läpi, mitä kukin tiimin jäsen on edellisenä päivänä tehnyt, tulee kyseisenä päivänä tekemään, mitä esteitä tekemiselle on ja mitä kukin aikoo seuraavana päivänä tehdä. [9] Palaverien tavoitteena on tuoda mahdollisimman nopeasti esille mahdollinen ongelmakohta tai haaste, joka haittaa ohjelmistokehitystä. Näin ongelma on kaikille helposti läpinäkyvä ja siihen pystytään reagoimaan nopeasti ketterästi ja kustannustehokkaasti. [8] Pääasiassa päivittäiseen palaveriin osallistuu kehitystiimi, mutta tarvittaessa muutkin Scrumin toimijat voivat osallistua [7] [8] [9].

Sprintin päätteeksi, kuvassa 2.1 oikealla, **suoritetaan sprinttikatselmus** (engl. Sprint Review) ja **sprintin retrospektiivi** (engl. Sprint Retrospective). Sprinttikatselmuksen tarkoituksena on tarkastaa päättyvän sprintin tulos ja määrittää sen pohjalta aiheutuvat mukautukset. Sprinttikatselmuksen aikana Scrum-tiimi ja sidosryhmät tarkastelevat, mitä sprintissä saavutettiin ja mitä muutostarpeita on ilmennyt. Näiden tietojen perusteella osallistujat tekevät mahdolliset muutokset seuraavaa sprinttiä varten. [7] [9]. Sprintin retrospektiivin tarkoituksena on suunnitella keinoja Scrum-tiimin laadun ja tehokkuuden parantamiseksi. Scrum-tiimi tarkastelee, miten edellinen sprintti sujui yksilöiden, vuorovaikutuksen, prosessien ja työkalujen

osalta. Scrum-tiimi keskustelee siitä, mikä meni hyvin sprintin aikana, mitä ongelmia kohdattiin ja miten nämä ongelmat ratkaistiin (tai jätettiin ratkaisematta). Sprintin retrospektiivi päättää sprintin.[7] [8] [9]

Tuoteversio (engl. Increment) on jokaisen sprintin lopussa tehtävä konkreettinen askel kohti lopullista tuotetta. Jokainen tuoteversio rakentuu edeltävien tuoteversion päälle. Jotta tuoteversio voidaan toimittaa asiakkaalle, on sen täytettävä tilannekohtaisesti määritellyt laatuvaatimukset. [7] [8]

Valmiin määritelmä (engl. Definition of Done) on formaali kuvaus siitä tilasta tuoteversiolle, jossa ominaisuus tai osa tavoittaa ennalta määrätyt vaatimukset. Ominaisuuden saavuttaessa nämä tavoitteet syntyy tuoteversio. Valmiin määritelmä tuo Scrumin kaikille osapuolille läpinäkyvyyttä ja varmistaa, että kaikille on selvää, missä kohdassa tietty ominaisuus on kelvollinen tuotantoon. Ennen kuin ominaisuudet saavuttavat ennalta määrätyt määritelmät “valmiille”, ei kyseistä tuoteversiota voida julkaista eikä sitä myöskään voida esittää sprinttikatselmuksessa. [7]

Scrumin avulla ohjelmistoprojektit voidaan jakaa pienempiin ja hallittavampiin osiin, mikä mahdollistaa jatkuvan kehityksen sekä palautteenannon koko projektin elinkaaren ajan [7].

Scrumin hyödyt ja haasteet

Scrumin tavoitteena on parantaa työn tehokkuuta ja sitä kautta myös asiakastyytyväisyyttä. Se korostaa joustavuutta, tiimityötä ja jatkuvaa kehittymistä. [9]

Scrumin etuna on sen tarjoama selkeä ja rakenteellinen kehys iteratiiviselle kehitykselle yrityksissä ja ohjelmistotiimeissä. Sen lisäksi Scrumissa projekti jaetaan lyhyisiin sprintteihin, mikä tekee tuotekehitysprosessista ketterää. [7] Scrumissa kannustetaan myös tiiviseen yhteistyöhön ja viestintään tiimin sisällä. Päivittäiset palaverit varmistavat, että kaikki osapuolet ovat ajan tasalla projektin etenemisestä. [9] . Hyödyllistä tämä on uusille yrityksille, jotka tarvitsevat selkeää sekä nopeaa

tuotteen tai palvelun kehittämistä.

Scrumissa on myös omat haasteensa. Scrum masterin rooli on keskeinen Scrumin onnistumisen kannalta. Pienissä tiimeissä voi olla vaikeaa löytää henkilöä, jolla on riittävästi aikaa ja kokemusta toimia Scrum masterina. Scrum-tiimi tarvitseekin osaavan Scrum masterin osakseen toimiakseen oikein. [8] Sprinttien onnistuminen edellyttää myös tehtävien selkeää määrittelyä ja realistista aikataulutusta. Pienillä tiimeillä voi olla haasteita sprintin tavoitteiden asettamisessa ja työmäärän arvioinnissa. On tärkeää määritellä tehtävät riittävän pieniksi, jotta ne voidaan saattaa loppuun sprintin aikana. Jos tehtävien seuranta on liian monimutkaista, se voi viitata siihen, että prosessista yritetään tehdä liian vaikeaa. [8] Myös menetelmän periaatteiden väärinymmärtäminen voi johtaa tehottomaan menetelmien käyttöön [10].

Scrum keskittyy ensisijaisesti työmäärän (backlogin) suorittamiseen sprinteissä, eikä niinkään suoraan hukan minimointiin, sillä Scrum ei itsessään ole Lean-prosessi [11]. Vaikka Scrum-tiimit pyrkivätkin työskentelemään tehokkaasti, menetelmä itsessään ei tarjoa työajan ja kustannustehokkuuden mittaamiseen valmiita työkaluja [11].

Scrumissa ei myöskään ole valmiina erillistä asiantuntijoiden erottelua, mikä voi johtaa siihen, ettei tehtäviä anneta oikeille henkilöille. Scrum-tiimin jäsenet voivat siis saada tehtäviä osaamisalueensa ulkopuolelta. Tämä voi johtaa tilanteeseen, jossa tehtävään kuluu enemmän aikaa, kuin jos se olisi annettu osaavammalle yksilölle. Täten myös kehitystyön kustannukset kasvavat. [11]

Yhtenä ratkaisuna Scrumin haasteisiiin on Scrum masterille ja tiimin jäsenille tarjottava koulutus ja valmennus. Ne auttavat ymmärtämään paremmin Scrumin periaatteita ja käytäntöjä sekä kehittämään taitoja, joita tarvitaan Scrum-tiimin tehokkaaseen toimintaan. [11] Mahdollisten ongelmien ennaltaehkäisemiseksi Scrumissa on tärkeää varmistaa selkeä ja tehokas viestintä tiimin jäsenten välillä. Sään-

nölliset tapaamiset, yhteiset työkalut ja selkeät vastuualueet auttavat tiimiä työskentelemään saumattomasti yhdessä. [11] [12]. Tiimin on myös säännöllisesti arvioitava omaa toimintaansa ja etsittävä tapoja parantaa sitä lieventääkseen sen haasteita. Tilaisuus, jossa voi pohtia sprinttiä ja tunnistaa siinä ilmeneviä parannuskohteita, on retrospektiivi. [13] Scrumia käyttävissä yrityksissä on myös tärkeää ottaa käyttöön jokin kustannusten mittausjärjestelmä, joka auttaa tunnistamaan ja minimoimaan mahdollisesti syntyvää hukkaa [11].

Scrumin tehokkuus perustuu osaltaan myös avoimeen kommunikaatioon ja tiiviiseen yhteistyöhön tiimin jäsenten ja sidosryhmien välillä. Pienissä tiimeissä kommunikaatio voi olla helpompaa kuin suurissa, mutta on silti tärkeää varmistaa, että kaikilla on yhtenäinen käsitys asioista ja että tietoa jaetaan tehokkaasti. [8]

2.2 Lean-ohjelmistokehitys

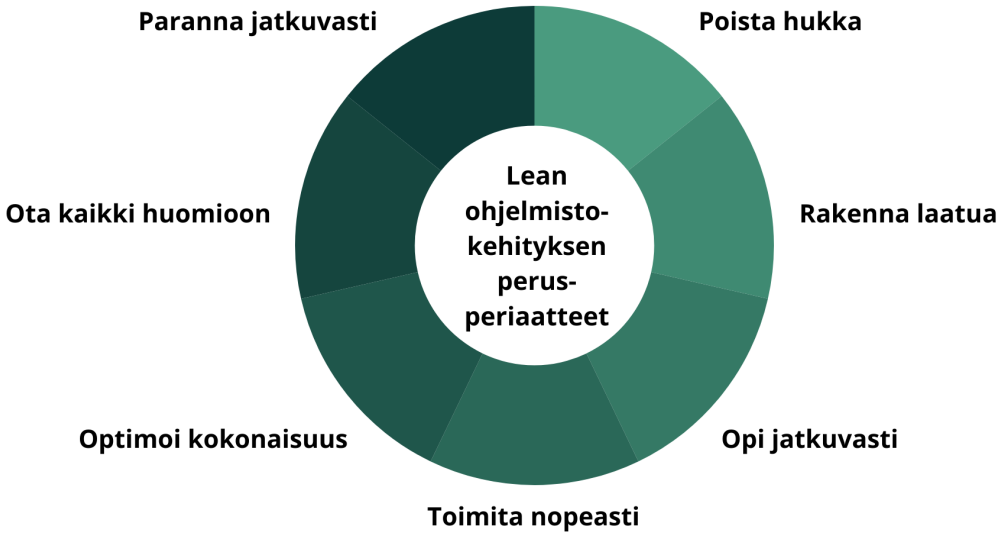
Lean on laajalti tunnettu ja käytetty johtamisfilosofia. Se keskittyy arvoa tuottamattomien toimintojen poistamiseen sekä toiminnan parantamiseen. Leanin uranuurtaja on Toyota Production System (TPS), joka toimi ajoneuvojen valmistusteollisuudessa 1900-luvun puolivälissä. Lean keskittyy työnkulun virtaviivaistamiseen, hukan vähentämiseen sekä toimintatapojen jatkuvaan parantamiseen. [2]

Lean-ohjelmistokehitys (engl. Lean Software Development) on ohjelmistokehityksen lähestymistapa, joka perustuu Leaniin ja sen perusperiaatteisiin [2] [14]. Se luotiin Leanin pohjalta ohjelmistoalan tarpeita varten. Siinä keskitytään kehitysprosessien optimointiin, hukan minimointiin ja maksimaalisen arvon tuottamiseen asiakkaille. [2]

Lean-ohjelmistokehityksessä on seitsemän perusperiaatetta, kts. kuva 2.2 [2]:

Kanbanin viisi perusperiaatetta ovat:

- *Optimointi kokonaisuus* (engl. optimize the whole) Lean-ohjelmistokehitys ko-



Kuva 2.2: Lean kehityksen seitsemän perusperiaatetta

rostaa kehitysprosessin kokonaisvaltaista tarkastelua. Ohjelmisto tai sen osat eivät ole irrallisia, vaan osa laajempaa kokonaisuutta. Asiakkaan prioriteettien selvittäminen ja siten oman työnkuvan ymmärtäminen sekä arvon tuottaminen ovat monimutkaisia prosesseja. Ne ulottuvat kehittämistä pidemmälle, kuten mm. suunnitteluun, käyttöönottoon ja pitkän aikavälin mukautuvuuteen. Kokonaisvaltainen optimointi varmistaa, että koko järjestelmä toimii tehokkaasti ja tuottaa arvoa asiakkaalle. [2] [15]

- *Poista hukka* (engl. eliminate waste). Hukalla tarkoitetaan kaikkea, mikä ei suoraan lisää arvoa tai paranna kykyä tuottaa tätä arvoa. Hukkaa voi ohjelmistokehityksessä esiintyä monessa muodossa, kuten tarpeettomina ominaisuuksina, keskeneräisenä työnä, virheinä ja vihreiden korjaukseen käytettynä aikana. Lean-ohjelmistokehitys keskittyy näiden tehottomuuksien tunnistamiseen ja niiden poistamiseen, mikä virtaviivaustaa kehitysprosessia. Kehittämällä osa kerrallaan valmiiksi saakka, kunnolla ja vain sellaisia ominaisuuksia,

joita asiakkaat haluavat, estetään ajan ja resurssien tuhlaaminen. [2] [15]

- *Rakenna laatua* (engl. build quality in). Laatua ei voi toimittaa jälkikäteen, vaan se on sisäänrakennettava osaksi kehitysprosessia. Automaattisen testauksen, koodin tarkistusten ja virheiden varhaisen havaitsemisen kaltaiset käytännöt varmistavat, että laatuongelmat havaitaan ja ratkaistaan nopeasti, mikä vähentää kalliiden jälkitöiden tarvetta myöhemmin. [2] [15]

- *Opi jatkuvasti* (engl. learn constantly). Jatkuva oppiminen on elintärkeää prosessien ja tulosten parantamiseksi. Lean-ohjelmistokehityksessä pyritään tilanteeseen, jossa suuret ja kalliit muutospäätökset ovat ennalta harkittuja ja puntaroituja, kun taas kehitystyössä oppimista tapahtuu jatkuvasti. [2] [15]

- *Toimita nopeasti* (engl. deliver fast). Jotta käyttäjille voidaan tuottaa arvoa nopeasti, olennaisen tärkeitä ovat lyhyet kehityssyklit, tiheät julkaisut ja iteratiiviset lähestymistavat. Ne antavat tiimeille mahdollisuuden reagoida palautteeseen ja muuttuviin vaatimuksiin ketterästi ja pienillä kustannuksilla. [2] [15]

- *Ota kaikki mukaan* (engl. engage everyone). Tiimien rohkaisu, kannustus ja tiimin jäsenten panoksen arvostus sekä kunnioitus ovat yksi Lean-ohjelmistokehityksen kulmakivi. Kaikkia on kannustettava päätöksentekoon heidän omalla tasollaan ja osaamisalueellaan. Monialainen yhteistyö, luottamus ja yksilöllisten ja yhteisten ponnistelujen arvostaminen luovat ympäristön, jossa innovointi kukoistaa. [2] [15]

- *Paranna jatkuvasti* (engl. keep getting better). Työjärjestelmiä ja prosessia on jatkuvasti hiottava organisaation alimmalta tasolta asti. Lean-ohjelmistokehityksessä esimerkiksi Scrumin kaltaisia käytäntöjä kehotetaan pitämään lähtökohtana, joita tiimit voivat ajan mittaan mukauttaa ja parantaa omiin tarpeisiinsa.

Muualla mahdollisesti toimivien menetelmien jäykkää soveltamista on vältettävä. [2] [15]

Lean-ohjelmistokehityksen hyödyt ja haasteet

Lean-ohjelmistokehityksen periaatteiden soveltaminen on erityisen tärkeää pienille ja uusille ohjelmistoyrityksille, koska ne tarvitsevat joustavuutta, nopeaa arvon tuottamista ja resurssien optimaalista käyttöä [6]. Lean-ohjelmistokehitysen periaatteet tuovat monia hyötyjä, joita voivat olla esimerkiksi lyhyemmät toimitusajat, parempi laatu ja tehokkaampi työskentely [2] [14] [15].

Jatkuva integrointi ja ohjelmiston nopea toimitus asiakkaalle mahdollistavat palautteen saamisen aikaisemmin, mikä osaltaan vähentää virheiden määrää ja niiden korjausta. Lisäksi laatu paranee ja asiakastyytyväisyys kasvaa, kun virheitä on vähemmän. Hukan vähentäminen vapauttaa resursseja arvoa tuottavaan työhön ja siten parantaa yrityksen tehokkuutta. Tämä osaltaan taas johtaa kustannussäästöihin.[2] [14] [15]

Lean-periaatteiden soveltaminen voi myös johtaa lyhyempiin toimitusaikoihin, koska prosessit tehostuvat ja turhat työvaiheet poistuvat. Tehokkaampaan työskentelyyn päästään vähentämällä hukkaa ja optimoimalla prosesseja, jolloin resursseja voidaan kohdentaa paremmin. [2] [14] [15]

Lean-ajattelussa keskeistä on myös asiakkaan tarpeiden ymmärtäminen. Tämä parantaa tuotteen onnistuvuutta, ja sitä kautta tuote on merkityksellisempi käyttäjille. Lisäksi tiimityö ja kaikkien osallistaminen päätöksentekoon vahvistavat osapuolten sitoutumista ja yhteistyötä.[2] [14] [15] Lean-ohjelmistokehityksen periaatteet auttavat siis yrityksiä parantamaan tehokkuuttaan, vähentämään hukkaa ja kehittämään parempia ohjelmistoja nopeammin ja kustannustehokkaammin.

Lean-ohjelmistokehityksellä on monia etuja, mutta siihen liittyy myös haasteita. Lean-periaatteet korostavat mm. tiimien autonomiaa ja nopeaa reagointia muutok-

siin.[14] Tämä voi tuottaa haasteita perinteisen hierarkian organisaatioille, joissa päätöksenteko voi olla hidasta ja kohdistua pääosin ylhäältä alas [14]. Periaatteiden soveltaminen voi olla haastavaa etenkin suurissa organisaatioissa, joissa jo entuudestaan on olemassa monimutkaisia prosesseja ja hierarkkisia rakenteita.[15] Myös Lean-ohjelmistokehitysken periaatteiden liian nopea käyttöönotto voi johtaa haasteisiin. Liian nopeasta menetelmien käyttöönotosta johtuva suoraviivaistaminen tai yksinkertaistaminen voi johtaa tärkeiden näkökulmien unohtumiseen tai väärinymmärtämiseen. [2] [14] [16]

Myös ohjelmistojen aineeton luonne ja tietoon perustuva työ voivat aiheuttaa erinäisiä vaikeuksia Lean-periaatteiden soveltamisessa. Vaikeuksia voi esiintyä esimerkiksi virtauksen määrittelyssä. [14] Lean-toteutuksissa on myös tärkeää mitata oikeita asioita. Vääränlaiset mittarit voivat johtaa vääriin johtopäätöksiin ja sitä kautta tehottomiin käytäntöihin. [15].

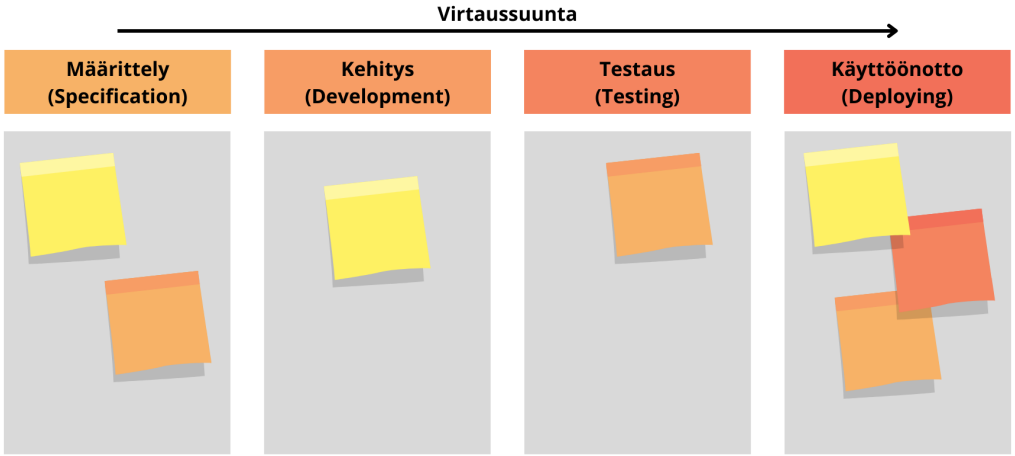
Lean-ajattelutavan omaksuminen voi osoittautua haasteelliseksi, ja siksi se vaatiikin organisaatiolta merkittävää kulttuurin muutosta, johon on varattava riittävästi aikaa. [14] Periaatteiden soveltaminen vaatii organisaatiolta huolellista suunnittelua, sopeutumista, sitoutumista ja jatkuvaa oppimista kaikilta organisaation tasoilta. [2] [14] [15]. Yritysten on oltava tietoisia Lean-ohjelmistokehitykseen liittyvistä haasteista ja varauduttava niihin, jotta Lean-periaatteiden hyödyntäminen onnistuu [15]. Parhaat tulokset saavutetaan, kun organisaatio valitsee ja soveltaa Lean-periaatteita, jotka parhaiten sopivat omaan kontekstiin ja tarpeisiin [2].

Lean-ohjelmistokehitys ei ole vain työkaluja ja tekniikoita, vaan se on kokonaisvaltainen ajattelutapa, joka vaatii sitoutumista ja jatkuvaa oppimista. Pienet yritykset, jotka omaksuvat Lean-ohjelmistokehityksen periaatteita, voivat saavuttaa merkittäviä etuja ja siten parantaa kilpailukykyään. [14]

2.3 Kanban

Kanban on laajalti käytetty ketterä projektinhallinta- ja tuotannon ajoitusmenetelmä. Se perustuu työnkulun visualisointiin, hallinnointiin ja optimointiin. Se on peräisin Toyota Production Systemsiltä ja perustuu Lean-johtamisfilosofiaan. Sen nykyinen muoto tietotyötä varten sai pääosin alkunsa Corbisilla[1] vuonna 2006, ja siitä on sittemmin kehittynyt maailmanlaajuinen, yhteistoiminnallinen kehys prosessien optimointiin. [18]

Kanban on strategia, jonka avulla optimoidaan arvon kulkua prosessin läpi visuaalisen, vetopohjaisen (engl. pull-based) järjestelmän avulla. [18] Keskeistä Kanbanissa on virtauksen käsite, joka tarkoittaa potentiaalisen arvon tehokasta siirtämistä järjestelmän läpi. Näin tehokkuus, toimivuus ja ennustettavuus paranevat. Kanbanin keskeisiä käytäntöjä ovat työnkulkujen määrittely ja visualisointi Kanban-taulun avulla, työkohteiden aktiivinen hallinta ja työnkulun jatkuva parantaminen hankittujen tietojen perusteella. [18] [19]



Kuva 2.3: Havainnekuva Kanbantaulusta

Ohjelmistotekniikassa Kanban-järjestelmä toteutetaan yleensä seinällä olevana

[1](Alunperin Interactive Home Systems, jonka perusti vuonna 1989 Bill Gates [17])

tauluna, kts. kuva 2.3, jossa on sarakkeita, jotka kuvaavat kehitysprosessin eri vaiheita eli arvovirtaa. Työvaiheita ja tehtäviä kuvaamaan käytetään kortteja, joita siirretään taulun sarakkeissa. [2] [20]. Tyypillinen ohjemistokontekstissa käytettävä Kanban-taulu sisältää ainakin sarakkeet määrittely-, kehitys-, testaus- ja käyttöönottovaihetta varten. Jokaiselle kortille määritellään kussakin sarakkeessa ehdot, joiden täyttyessä korttia siirretään virtaussuunnassa eteenpäin seuraavaan sarakkeeseen. [20]

Kanbanin viisi perusperiaatetta ovat:

- *Visualisoi työnkulku:* Tiimit käyttävät fyysisiä tai virtuaalisia tauluja, joissa on kortteja tehtävien esittämiseen. Se mahdollistaa edistymisen visualisoinnin ja edistää itseohjautuvuutta. [9] [20] Kts. kuva 2.3

- *Rajoita keskeneräistä työtä:* (engl. Work in Progress, WIP) WIP:llä hallinnoidaan työnkulkua ja sen määrää. Selkeät WIP-rajat estävät ylikuormitusta ja varmistavat keskittymisen aktiivisiin tehtäviin. [20]

- *Mittaa ja hallinnoi virtausta:* Työkalut, kuten arvovirtakartat, kumulatiiviset virtauskaaviot ja Kanban-taulut, auttavat seuraamaan virtauksen mittareita (jonon koko, sykli, läpimenoaika) ja varmistamaan työn sujuvan edistymisen. [20]

- *Tee prosessikäytännöistä selkeitä:* Tehtäville määritellyt sisään- ja ulosmenokriteerit lisäävät avoimuutta ja helpottavat prosessin optimointia ja työnkulun hallintaa. [20]

- *Käytä malleja parannusmahdollisuuksien tunnistamiseen:* Erilaiset tekniikat auttavat systemaattisesti tunnistamaan parantamisen tarpeessa olevat alueet. [19] [20]

Yleisesti virtaus ja pullonkaulat ovat tärkeimpiä päivittäisissä kokouksissa käsiteltäviä asioita, ja niillä on ratkaiseva merkitys parannusmahdollisuuksien tun-

nistamisessa. Visuaalisen taulun avulla on helpompi hahmottaa, seurata ja johtaa projektin etenemistä. [20]

Kanban on ohjelmistokehityksessä monipuolinen ja tehokas kehys työnkulun optimointiin, tiimin yhteistyön parantamiseen ja laadukkaiden tuotteiden sekä palveluiden tehokkaaseen tuottamiseen. Tehtäviä visualisoimalla, keskeneräistä työtä rajoittamalla ja jatkuvaa parantamista edistämällä Kanban antaa tiimeille mahdollisuuden mukautua dynaamisesti muuttuviin vaatimuksiin ja ylläpitää tasaista tuottavuutta. Kanbanin joustavuus ja tehokkuuteen keskittyminen antavat mahdollisuuden vastata käyttäjien tarpeisiin. [2] [20]

Kanbanin hyödyt ja haasteet

Kanban on joustava ja mukautuva menetelmä, joka sopii hyvin aloittavien yritysten muuttuviin vaatimuksiin. Erityisen hyödyllinen se on sellaisille uusille yrityksille, joilla ei välttämättä ole vielä selkeää käsitystä tulevan tuotteen tai palvelun kaikista vaatimuksista ja ominaisuuksista. [10] Kanban visualisoi työnkulkua, mikä helpottaa pullonkaulojen tunnistamista ja poistamista. Kanban-taulu, kts. kuva 2.3, tarjoaa selkeän kuvan siitä, missä vaiheessa mikäkin tehtävä on ja mitä on tulossa. Kanban rajoittaa keskeneräisten töiden määrää (WIP), mikä auttaa keskittymään tärkeimpiin tehtäviin. Tämä auttaa vähentämään hukkaa ja nopeuttamaan läpimenoaikaa. [9] [19]

Ohjelmistokehityksessä Kanbanin käyttöön liittyy silti myös useita haasteita. Nämä haasteet voidaan jakaa karkeasti kolmeen kategoriaan: prosessiin, ihmisiin ja organisaatioon liittyviin haasteisiin. [19]

Itse Kanban-prosessin käyttöönotto ja ylläpito voi aiheuttaa haasteita. Organisaatioiden on varattava aikaa ohjelmistotiimeille, jotta ne voivat suunnitella ja ylläpitää Kanbania iteratiivisesti ja yhdistää sen osaksi omaa toimintaansa. [19]

Ihmisiin liittyvistä merkittävistä haasteista Kanbanin käytössä voidaan maini-

ta johdon tuen puute sekä organisaation haluttomuus omaksua uusia menelmiä. Myös Kanbanin käsitteiden ja käytäntöjen heikko ymmärrys sekä johdon että tiimien tasolla voi haitata sen tehokasta käyttöönottoa. Viestinnän puutteet tiimien ja asiakkaiden välillä voivat vaikeuttaa kehitystyötä, ja tiimien vastustus voi johtua joko muutosvastarinnasta tai Kanbanin väärinymmärryksestä. Lisäksi osaamisvaje, erityisesti pienissä tiimeissä, saattaa hidastaa Kanbanin hyödyntämistä. [19] [10]

Organisaatiotasolla Kanbanin käyttöönotto muodostuu haasteelliseksi, mikäli tukikäytännöissä, koulutuksessa tai tiedonhallinnassa esiintyy puutteita. Kanban vaatii avoimuutta, yhteistyötä ja jatkuvaa parantamista, mikä voi olla ristiriidassa olemassaolevan organisaatiokulttuurin kanssa. Organisaatiokulttuurissa tapahtuvat muutokset vaatitavat aikaa ja sitoutumista. [19]

Kanban-taulu on pidettävä yksinkertaisena ja siinä on keskittyttävä olennaiseen [10]. Johdon koulutus ja mentorointi ovat keskeisiä koko organisaation Kanban-osaamisen kehittämisessä. Koulutus auttaa ymmärtämään Kanbanin hyödyt, sitouttaa johdon ja tiimin jäsenet sekä varmistaa, että viestintä toimii ja kaikki osapuolet pysyvät ajan tasalla. [19] Kanbanin käyttöönottoon on varattava riittävästi aikaa ja se on mukautettava organisaation ja tiimin tarpeisiin, jotta tiimit voivat sopeutua uuteen työskentelytapaan. Koulutus ja mentorointi ovat myös tärkeitä menestystekijöitä. Kanban vaatii sitoutumista ja jatkuvaa arviontia sekä parantamista. [18] [19]

On tärkeää luoda selkeät käytänteet ja koulutusohjelmat Kanbanin tueksi sekä varmistaa tehokas tiedonjako tiimien ja sidosryhmien välillä. Lisäksi integrointi olemassa oleviin prosesseihin voi olla haastavaa, erityisesti pienille tiimeille. Suunnitelmallinen lähestymistapa on välttämätön onnistumiselle. [19]

2.4 MVP

Pienin toimiva tuote (engl. Minimum Viable Product, MVP) on markkinoille pyrkivä uusi tuote tai palvelu, joka sisältää käyttäjän kannalta ainoastaan välttämättömät ominaisuudet [3] [21] [22]. Käsitteen esitteli ensimmäistä kertaa Frank Robinson vuonna 2001. MVP on jatkuvasti kehittyvä käsite, jonka Eric Ries määritteli uudelleen vuonna 2011 seuraavasti: ”[MVP on] versio uudesta tuotteesta, jonka avulla tiimi voi kerätä mahdollisimman paljon validoitua tietoa asiakkaista mahdollisimman vähällä vaivalla”. [3] [21]

Vähimmäistavoitteet täyttävä uusi tuote tai palvelu antaa organisaatiolle mahdollisuuden kerätä mahdollisimman suuren määrän käyttäjälähtöistä palautetta pienimmällä mahdollisella vaivalla tulevaa tuotekehitystä varten. Näin voidaan testata olettamuksia, saada syvällisempää ymmärrystä käyttäjien tarpeista ja myös määrittää, onko tuotteelle todellista markkinasoveltuvuutta. [21]

MVP:n keskeinen tarkoitus on validoida ideat nopeasti, varmistaa niiden elinkelpoisuus ja minimoida samalla tarpeettomien ominaisuuksien kehittämiseen liittyvät riskit ja kustannukset. Tämän lähestymistavan ansiosta tiimit voivat keskittyä palautteena saatujen ongelmien ratkaisemiseen ja tuotteen hiomiseen todellisten käyttäjien näkemysten perusteella, ilman että arvokkaita resursseja tuhlataan turhiin ominaisuuksiin tai täysin turhaan tuotteeseen. [3] [21] [22]

MVP ei varsinaisesti ole halvin tai yksinkertaisin mahdollinen tuote, eikä se myöskään välttämättä vastaa prototyyppiä. MVP:n ei myöskään välttämättä tarvitse ohjelmistoalallakaan aluksi olla vielä oikea ohjelmisto, vaan se voi olla jonkinlainen kokeilu siitä, miten idea vastaa tarvetta markkinassa ja miten hypoteesit saadaan muutettua faktapohjaisiksi tiedoiksi. Esimerkki tästä voisi olla mahdollisen käyttäjäkunnan haastattelut ja ensiasteinen ideoiden yksinkertaistettu kokeiluversio. [21]

MVP:n rakentaminen alkaa ongelman selkeällä tunnistamisella ja kohderyhmän ymmärtämisellä. MVP:llä pyritään vastaamaan käyttäjien tarpeisiin. Kohdennet-

tu lähestymistapa edistää suoraviivaista ja käyttäjäkeskeistä tuotekehitysprosessia ja parantaa uuden markkinoille pyrkivän tuotteen onnistuvuutta. Sen rakentaminen edellyttää tasapainon löytämistä yksinkertaisuuden ja toiminnallisuuden välillä. MVP ei ole lopullinen tuote, mutta se on tärkeä ponnahduslauta kohti käyttäjäkeskeistä ratkaisua. Ohjelmistokehitystiimit voivat merkittävästi parantaa mahdollisuuksiaan luoda onnistunut tuote-markkinasovitus keskittymällä olettamusten varhaiseen validointiin ja hyödyntämällä käyttäjäpalautetta. Näin on myös mahdollista luoda skaalautuva perusta tulevalle tuotekehitykselle onnistuneesti. [3] [21] [22]

Ohjelmistotuotannon MVP:ssä ei ole kyse vain siitä, että rakennetaan vähemmän, vaan siitä, että rakennetaan fiksusti. MVP yhdenmukaistaa kehityksen todellisten tarpeiden kanssa, vähentää riskejä ja mahdollistaa kohdennetun innovoinnin. Iteratiivisen oppimisen ja mukautuvuuden ansiosta MVP onkin, varsinkin uusille yrityksille ja startupeille, nykyaikaisen ohjelmistotuotekehityksen yksi kulmakivi. [22] [23]

MVP:n hyödyt ja haasteet

Koska pienet ja uudet yritykset omaavat yleisesti vain rajallisen markkinatuntemuksen, on MVP erityisen tärkeä niille, sillä se voi auttaa optimoimaan resurssien käyttöä ja minimoimaan riskejä [3] [21] [22].

MVP:n keskeisiä etuja ovat:

- *Nopea oppimisprosessi:* MVP:n avulla yritykset voivat nopeasti testata liiketoimintahypoteesiaan ja kerätä arvokasta tietoa asiakkaistaan minimaalisella panostuksella. Sen sijaan, että keskityttäisiin täydellisen tuotteen kehittämiseen, MVP:n avulla voidaan kerätä tietoa asiakkaiden tarpeista ja mieltymyksistä jo varhaisessa kehitystyön vaiheessa. [3] [21] [22]

- *Riskien minimointi:* MVP:n avulla yritykset voivat vähentää riskejään testaa-

malla ideoitaan ja keräämällä palautetta ennen kuin ne investoivat merkittävästi aikaa ja rahaa tuotteen kehittämiseen. Näin voidaan varmistua siitä, että tuotteelle on kysyntää ja että se vastaa asiakkaiden tarpeita. [3] [21] [22]

- *Asiakaskeskeisyys:* Yritykset voivat keskittyä asiakkaiden tarpeisiin ja kehittää tuotetta, joka vastaa näihin tarpeisiin. Keräämällä palautetta asiakkailta MVP:n avulla voidaan varmistaa, että tuote on hyödyllinen ja että se ratkaisee asiakkaiden ongelmia. [3] [22]

- *Matalammat kehityskustannukset:* MVP:n avulla yritykset voivat vähentää kehityskustannuksia keskittymällä vain välttämättömimpien ominaisuuksien kehittämiseen. Tämä auttaa minimoimaan hukkaan heitetyn työn määrää ja varmistamaan, että resurssit käytetään tehokkaasti. [22]

MVP:ssä esiintyy myös omat haasteensa. MVP:n määritelmä on ajanmittaa hämärtynyt ja sitä käytetään usein virheellisesti synonyyminä prototyypille tai tuotteen ensimmäiselle versiolle [3]. Alkuperäinen MVP:n tarkoitus, eli nopean oppimisen ja hypoteesien testaamisen työkalu, on monimuotoistunut [3]. Määritelmän väärinymmärrykset voivat johtaa siihen, että MVP:stä tulee liian laaja ja sen avulla ei testatakaan keskeisimpiä ominaisuuksia. Tai toisaalta MVP:stä saattaa tulla liian suppea, jolloin ei saada tarkkaa dataa asiakaskunnasta. [3] Monet startupit keskittyvät MVP:tä kehittäessään enemmän tuotteen valmistamiseen kuin ideoidensa testaamiseen hypoteesien avulla [3].

3 Analyysi ja pohdinta

Uusiin ja pieniin ohjelmistoja kehittäviin yrityksiin kohdistuu suuria riskejä, jotka liittyvät pääasiassa henkilöstöön, teknisiin haasteisiin ja markkinasoveltuvuuteen, samalla kun ne hallitsevat pääomarajoituksiaan [10] [6]. Ketterän manifestin arvojen keskiössä ovat asiakasyhteistyö, jatkuva parannus, kyky reagoida muutoksiin nopeasti sekä yksilöiden ja vuorovaikutusten huomioiminen prosessien työkalujen sijaan [5]. Kun resursseja on rajallisesti, on keskityttävä vain olennaiseen ja vähennettävä turhia kustannuksia [6]. Näitä kuvauksia arvomaailmasta voitaisiin hyvin soveltaa myös uuden ohjelmistoyrityksen arvomaailmaan.

Kussakin menetelmässä on omat hyötynsä ja haasteensa. Pienten ja uusien ohjelmistoalan yritysten onkin voitava sovittaa näiden hyödyt omien tavoitteiden ja rajoitteidensa kanssa. Menetelmien erojen ymmärtäminen aloittaville ohjelmistoyrityksille on tärkeää. Niiden avulla yritykset voivat räätälöidä lähestymistapaansa mm. parantamaan tehokkuuttaan, minimoimaan hukkaa ja sovittamaan kehitystoimintaansa yhteen organisaation tavoitteiden kanssa.

Scrum soveltuu parhaiten silloin, kun:

- *Tiimi on pieni ja kokenut:* Tiimin ideaalikoko on tyypillisesti seitsemän henkeä [9], ja yleensä enintään 10 henkeä [7] [8]. Pienemmille tiimeille Scrumin rakenne voi olla liian raskas, kun taas isommille tiimeille se voi olla haastavaa hallita [9]. Scrum toimii parhaiten kokeneiden tiimien kanssa, jotka osaavat työskennellä itsenäisesti ja tehokkaasti [8].

- *Projekti on monimutkainen ja vaatii selkeää rakennetta:* Scrum tarjoaa selkeän kehyksen ja prosessin, joka voi auttaa hallitsemaan monimutkaisia projekteja [7]. Sprintit, backlog ja roolit auttavat tiimiä pysymään aikataulussa, hallitsemaan kompleksisuutta ja varmistamaan, että kaikki työskentelevät kohti yhteistä tavoitetta [8] [9].

- *Yritys haluaa parantaa ennustettavuutta ja läpinäkyvyyttä:* Scrumissa projektin eteneminen on selkeästi nähtävissä sprinttien ja backlogien kautta, joka auttaa parantamaan ennustettavuutta ja läpinäkyvyyttä sekä tiimin sisällä, että asiakkaille [8] [9].

Lean-ohjelmistokehitys soveltuu parhaiten silloin, kun:

- *Tavoitteena on vähentää kustannuksia ja lisätä tehokkuutta*

- *On tarve kehittää ja optimoida prosesseja sekä vähentää turhaa työtä*

- *Halutaan keskittyä asiakaslähtöiseen kehitykseen*

Lean-ohjelmistokehitys on hyvä valinta tiimeille, joilla on tarve jatkuvasti kehittää prosessejaan ja vähentää turhaa työtä. Sen periaatteet, kuten jatkuva oppiminen ja parantaminen, auttavat tiimejä mukautumaan projektin muuttuviin vaatimuksiin.

Kanban soveltuu parhaiten silloin, kun:

- *Tiimin koko ja kokemus vaihtelevat:* Kanban on joustava menetelmä, joka sopii hyvin erikokoisille ja -tasoisille tiimeille. Kanbania voidaan käyttää myös silloin, kun tiimissä on jäseniä, joilla on eri roolit, osaaminen ja vastuualueet. [9]

- *Projektin vaatimukset muuttuvat usein:* Kanban on erittäin mukautuva menetelmä, joka sopii hyvin projekteihin, joissa vaatimukset muuttuvat usein. Kanban-taulun avulla tiimi voi helposti priorisoida ja muuttaa tehtäviä tarpeen mukaan. [9]

- *Yritys haluaa keskittyä työnkulun optimointiin:* Kanbanin keskeinen periaate on työnkulun visualisointi ja pullonkaulojen poisto, joka auttaa tiimiä parantamaan tehokkuutta [9] [15].

MVP soveltuu parhaiten silloin, kun:

- *Yrityksellä on tarve testata nopeasti uuden tuotteen tai ominaisuuden markkinakelpoisuutta:* Yritys haluaa testata nopeasti uuden tuotteen tai ominaisuuden markkinakelpoisuutta. MVP-menetelmä on hyödyllinen erityisesti markkinoille pyrkiville startup-yrityksille ja innovatiivisille projekteille, joissa on tärkeää validoida idea ennen suurempia investointeja.

- *Käyttäjien tarpeita ei vielä täysin tunneta:* Luomalla kevyen, mutta toimivan version tuotteesta yritys voi saada arvokasta palautetta ja tehdä tietoon perustuvia päätöksiä jatkokehityksestä.

- *Halutaan minimoida kehitystyöhön liittyviä riskejä ja kustannuksia:* MVP:n avulla voidaan varmistaa, että vain olennaiset toiminnot rakennetaan ensin, jolloin resurssit käytetään viisaasti ja vältetään epävarmat investoinnit tarpeettomiiin ominaisuuksiin.

MVP ensiversion jälkeinen iteratiivinen parantaminen mahdollistaa yrityksille oikean suunnan tuotteelle heti tuotteen elinkaaren alusta alkaen. MVP:n avulla voidaan minimoida kehitysriskiä kehittämällä vain välttämättömät ominaisuudet alkuvaiheessa. Tämä auttaa välttämään resurssien tuhlaamista ja mahdollistaa nopean reagoinnin muuttuviin vaatimuksiin.

Scrum sopii hyvin strukturoituihin, tiimikeskeisiin projekteihin, Kanban tarjoaa joustavuutta dynaamisiin työnkulkuihin, Lean-ohjelmistokehitys auttaa optimoimaan resurssien käyttöä, kun taas MVP on strateginen valinta nopeaan ensikehitykseen ja idean validointiin. Taukukkoon 3.1 on kerätty mahdollisia tilanteita menetelmien käytölle.

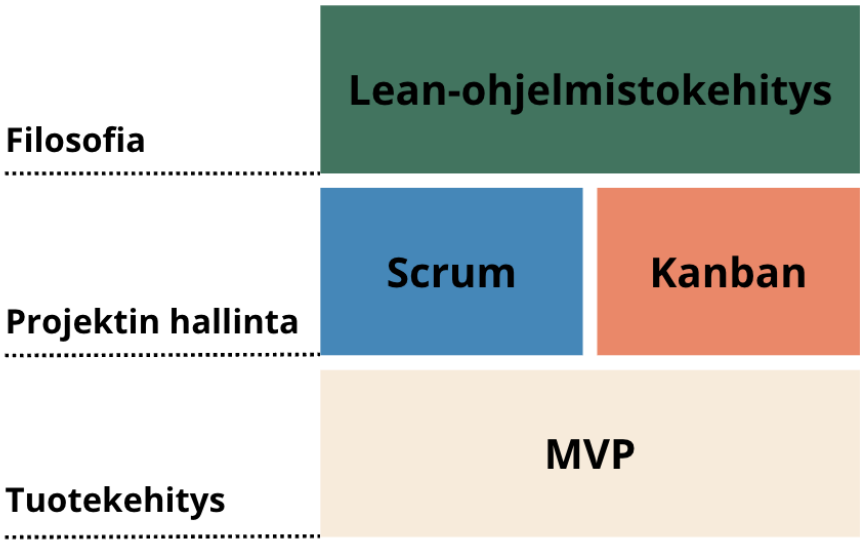
Tilanne	**Suositeltu menetelmä**	**Miksi?**
Uuden tuotteen kehitys alussa	MVP + Lean	Auttaa validoimaan idean nopeasti ilman turhaa kehitystä.
Iteratiivinen kehitys selkeillä tavoitteilla	Scrum	Sprinttien avulla voidaan toteuttaa ja mitata edistystä tehokkaasti.
Muuttuvat ominaisuudet, jatkuva ylläpito ja bugikorjaukset	Kanban	Mahdollistaa jatkuvan työnkulun ilman sprinttirajoituksia.
Nopeasti muuttuva ympäristö, jossa vaaditaan joustavuutta	Kanban + Lean	Minimoidaan hukkaa ja mukautetaan kehitystä tarpeen mukaan.
Kasvava startup, jossa asiakaslähtöisyys on keskiössä	Lean + Scrum	Yhdistelmä auttaa kasvavaa yritystä mukautumaan asiakastarpeisiin, samalla säilyttäen ketteryytensä.

Taulukko 3.1: Eri tilanteisiin suositellut kehitysmenetelmät

Scrumin ja Kanbanin yhteiskäyttöä kutsutaan nimellä Scrumban [13]. Se on hybridimalli, joka yhdistää Scrumin iteratiivisen rakenteen ja Kanbanin joustavuuden sekä työnkulun hallinnan [13] [15]. Se on kehittynyt, kun Scrum-tiimit ovat alkaneet hyödyntää Kanban-taulua ja sen periaatteita työnkulun visualisointiin ja ohjaamiseen. Scrumban on usein hyödyllinen tiimeille, jotka ovat jo käyttäneet Scrumia ja haluavat siirtyä joustavampaan malliin. [13] Yleensä tämä tarkoittaa, että tiimi säilyttää sprinttien kaltaisia aikarajoja, mutta poistaa tiukat sprinttisuunnittelukokoukset ja keskittyvät enemmän työnkulun jatkuvaan virtaukseen [13].

Scrumbanin etuja ovat joustavuus ja mukautuvuus. Scrumban on joustavampi kuin perinteinen Scrum ja se soveltuu paremmin tilanteisiin, joissa vaatimukset muuttuvat usein. [13] Scrumbanin keskeisin haaste on soveltaminen käytäntöön, ja se voi vaatia kokeilua. Koska se on hybridimalli, tiimin jäsenten on ymmärrettävä sekä Scrumin että Kanbanin periaatteet, jotta Scrumban voi toimia tehokkaasti. [13]

Myös muunlaista menetelmien samanaikaista käyttöä voidaan kasvavassa uudessa ohjelmistoyrityksessä harkita. Kaikkia tässä tutkimuksessa mainittuja menetelmiä on mahdollista käyttää samanaikaisesti, huomioiden resurssimahdollisuudet.



Kuva 3.1: Havainnekuva menetelmien yhteiskäytöstä

Kuvassa 3.1 on havainnollistettu menetelmien mahdollista yhteiskäyttöä sekä eroavaisuuksia niiden luonteessa. Lean-ohjelmistokehitys on ylemmän tason ajattelun filosofia, Scrum on projektinhallinnan toimintamalli, Kanban on projektinhallinnan työkalu tuotannon ajoitukselle ja MVP on taas tapa kehittää uutta tuotetta. Jokainen menetelmä keskittyy omaan osa-alueeseensa. Menelmien luonteiden ja osa-alueiden eroavaisuuksien ansiosta ne eivät ole toisiaan poissulkevia.

Pienelle aloittavalle ohjelmistoyritykselle on tärkeä valita menetelmä, joka sopii parhaiten juuri kyseisen yrityksen kontekstiin, tarpeisiin ja resursseihin. Riippumatta kuitenkin valitusta menetelmästä, uuden ohjelmistoyrityksen on keskityttävä monien haasteiden selättämiseen. Kustannustehokkaat ja ketterät menetelmät tarjoavat mahdollisuuksia vastata näihin haasteisiin. Menetelmien onnistunut soveltaminen kuitenkin edellyttää mukautumiskykyä, jatkuvaa oppimista ja avointa viestintää.

4 Yhteenveto

Kustannustehokkaat ja ketterät menetelmät voivat olla uudelle ohjelmistokehitysyritykselle elintärkeitä keinoja. Ne voivat auttaa uutta ohjelmistokehitysyritystä tuottamaan kilpailukykyisiä tuotteita ja palveluita nopeasti, kustannustehokkaasti sekä pienemmällä riskillä. Yritysten on silti punnittava tarkkaan myös näiden menetelmien haasteita, sillä usein resurssit ovat rajalliset, kilpailu kovaa ja liiketoimintaympäristö alati muuttuvaa.

Tässä tutkielmassa pyrittiin vastaamaan siihen, miten kustannustehokkaat ja ketterät ohjelmistokehitysmenetelmät tukevat kehitystyötä uusissa ohjelmistoyrityksissä ja mitkä ovat niihin liittyvät keskeisimmät haasteet. Tutkimus tehtiin kirjallisuuskatsauksena. Työn tutkimuskysymykset ja -vastaukset ovat:

- TK1: Mitkä ovat keskeisimmät menetelmät ja käytännöt, joilla uudet ohjelmistoyritykset voivat toteuttaa kustannustehokasta ja ketterää ohjelmistokehitystä?

Kustannustehokkaita ja ketteriä menelmiä on monia. Tutkielmassa käsiteltiin laajuussyistä näistä neljää: Scrumia, Lean-ohjelmistokehitystä, Kanbania ja MVP:tä. Lisäksi käsiteltiin myös näiden menetelmien samanaikaista käyttöä.

- TK2: Miten nämä ketterät ohjelmistokehitysmenetelmät tukevat uusien ohjelmistoyritysten toimintaa ja kehitysprojekteja?

Tutkitut menetelmät tukevat ohjelmistokehityksen projektinhallintaa siten, että ne mahdollistavat projektille selkeän rakenteen (Scrum) ja tehokkaamman työs-

kentelyn (Lean-ohjelmistokehitys), auttavat projektin visualisoinnissa (Kanban) sekä tuotteen nopeammassa markkinoille tuomisessa (MVP). Tämä kaikki puolestaan mahdollistaa sen, että kustannukset alenevat, toimitusajat lyhynevät, virheiden määrä vähenee ja asiakastyytyväisyys kasvaa.

- TK3: Mitkä ovat kustannustehokkaan ja ketterän ohjelmistokehityksen keskeisimmät haasteet uusissa ohjelmistoyrityksissä?

Kustannustehokkaan ja ketterän ohjelmistokehityksen keskeisimpiä haasteita uusissa ohjelmistoyrityksissä ovat menetelmien mahdollinen väärinymmärrys, heikko viestintä ja liian nopea käyttöönotto. Scrum voi osoittautua liian raskaaksi pienelle tiimille. Lean-ohjelmistokehityksen periaatteiden soveltaminen tietotyöhön ja ohjelmistoihin voi olla haasteellista niiden aineettoman luonteen vuoksi. Kanbanin vaatima avoimuus ja yhteistyö sekä MVP:n liian suppea tai laaja toteutus voivat myös osoittautua haasteellisiksi.

Kustannustehokkuus kiinnostaa monia yrityksiä, ja ketterät menetelmät ovat aiheena hyvin laaja. Jatkotutkimuksessa voitaisiin toteuttaa samanlainen kirjallisuuskatsaus esimerkiksi muista ketteristä ja kustannustehokaista menetelmistä. Näitä voisivat olla esimerkiksi XP (Extreme Programming) [2], joka on ohjelmistokehitysmenetelmä, tai SAFe (Scaled agile framework) [12], joka on joukko organisaatioja työnkulun malleja. XP:n tarkoituksena on parantaa ohjelmistojen laatua ja vastata muuttuviin asiakasvaatimuksiin [2]. SAFe:n tarkoituksena on ohjata yrityksiä skaalautumaan Lean- ja ketterien periaatteiden mukaisesti [12]. Näitä sekä jo tutkielmassa tutkittuja menetelmiä ja niiden toimivuutta voitaisiin mitata esimerkiksi kyselytutkimuksella niitä hyödyntävissä yrityksissä.

Yhteenvetona voidaan todeta, että kustannustehokkaat ja ketterät ohjelmistokehitysmenetelmät ovat keskeisessä roolissa uusien ohjelmistoyritysten menestyksessä.

Ne mahdollistavat nopean reagoinnin markkinoiden muutoksiin, tehokkaan resurssien käytön sekä asiakaslähtöisen tuotekehityksen. Ketteryys ja kustannustehokkuus tukevat innovointia ja riskienhallintaa. Hyödyntämällä oikeita menetelmiä ja työkaluja yritykset voivat kehittää kilpailukykyisiä tuotteita nopeasti ja joustavasti, mikä vahvistaa niiden asemaa markkinoilla sekä tukee niiden pitkäaikaista kasvua.

Lähdeluettelo

[1] M. Choras, T. Springer, R. Kozik et al., "Measuring and improving agile processes in a small-size software development company", *IEEE Access*, vol. 8, s. 78 452–78 466, 2020, ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2990117.

[2] M. Poppendieck ja M. A. Cusumano, "Lean software development: A tutorial", *IEEE Software*, vol. 29, s. 26–32, 5 2012, ISSN: 07407459. DOI: 10.1109/MS.2012.107.

[3] V. Lenarduzzi ja D. Taibi, "MVP Explained: A Systematic Mapping Study on the Definitions of Minimal Viable Product", teoksessa *Proceedings - 42nd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2016*, Institute of Electrical ja Electronics Engineers Inc., lokakuu 2016, s. 112–119, ISBN: 9781509028191. DOI: 10.1109/SEAA.2016.56.

[4] M. Kuhrmann, P. Tell, R. Hebig et al., "What Makes Agile Software Development Agile?", *IEEE Transactions on Software Engineering*, vol. 48, s. 3523–3539, 9 syyskuu 2022, ISSN: 19393520. DOI: 10.1109/TSE.2021.3099532.

[5] M. Beedle ja muut, *Manifesto for Agile Software Development*, https://agilemanifesto.org/, [Viitattu 07-11-2024], 2001.

[6] R. Razdan ja S. Kambalimath, "Super lean software startup engineering management", *2019 IEEE Technology and Engineering Management Conference, TEMSCON 2019*, kesäkuu 2019. DOI: 10.1109/TEMSCON.2019.8813609.

[7] K. Schwaber ja J. Sutherland, *Scrum Guide | Scrum Guides — scrumguides.org*, https://scrumguides.org/scrum-guide.html, [Viitattu 05-11-2024].

[8] L. Rising ja N. Janoff, "The Scrum software development process for small teams", *IEEE Software*, vol. 17, nro 4, s. 26–32, 2000. DOI: 10.1109/52.854065.

[9] "A statistical analysis of the effects of Scrum and Kanban on software development projects", *Robotics and Computer-Integrated Manufacturing*, vol. 43, s. 59–67, helmikuu 2017, ISSN: 07365845. DOI: 10.1016/j.rcim.2015.12.001.

[10] C. Pereira, A. Santos, L. MacHado ja L. Zaina, "How developers feel about tools: An investigation on software startup professionals experience with virtual kanban boards", teoksessa *Proceedings - 15th International Conference on Cooperative and Human Aspects of Software Engineering, CHASE 2022*, Institute of Electrical ja Electronics Engineers Inc., 2022, s. 1–10, ISBN: 9781450393423. DOI: 10.1145/3528579.3529172.

[11] E. A. P. S. Kom ja S. T. E. Suryani, "Designing cost measurement system in a small scrum based software company using activity based costing model (case study: ABC Company)", *2019 International Conference on Information and Communications Technology, ICOIACT 2019*, s. 943–947, heinäkuu 2019. DOI: 10.1109/ICOIACT46704.2019.8938480.

[12] M. Tanaka ja M. Aoyama, "A distributed large-scale agile software development for multiple products and its practical evaluation", *2021 IEEE/ACIS 19th International Conference on Software Engineering Research, Management and Applications, SERA 2021*, s. 66–72, 2021. DOI: 10.1109/SERA51205.2021.9509268.

[13] N. Nikitina, M. Kajko-Mattsson ja M. Stråle, ”From scrum to scrumban: A case study of a process transition”, teoksessa *2012 International Conference on Software and System Process (ICSSP)*, 2012, s. 140–149. DOI: 10.1109/ICSSP.2012.6225959.

[14] C. Ebert, P. Abrahamsson ja N. Oza, ”Lean software development”, *IEEE Software*, vol. 29, s. 22–25, 5 2012, ISSN: 07407459. DOI: 10.1109/MS.2012.116.

[15] X. Wang, K. Conboy ja O. Cawley, ”“Leagile” software development: An experience report analysis of the application of lean approaches in agile software development”, *Journal of Systems and Software*, vol. 85, s. 1287–1299, 6 kesäkuu 2012, ISSN: 0164-1212. DOI: 10.1016/J.JSS.2012.01.061.

[16] F. Dobrigkeit, D. D. Paula ja N. Carroll, ”InnoDev Workshop: A One Day Introduction to Combining Design Thinking, Lean Startup and Agile Software Development”, *2020 IEEE 32nd Conference on Software Engineering Education and Training, CSEE and T 2020*, s. 189–198, marraskuu 2020. DOI: 10.1109/CSEET49119.2020.9206184.

[17] Wikipedia, *BENlabs*, https://en.wikipedia.org/wiki/BENlabs, [Viitattu 16-12-2024], 2024.

[18] D. Vacanti ja J. Coleman, *Kanban Guide — kanbanguides.org*, https://kanbanguides.org/english/, [Viitattu 19-11-2024], 2020.

[19] M. O. Ahmad, D. Dennehy, K. Conboy ja M. Oivo, ”Kanban in software engineering: A systematic mapping study”, *Journal of Systems and Software*, vol. 137, s. 96–113, maaliskuu 2018, ISSN: 01641212. DOI: 10.1016/j.jss.2017.11.045.

[20] P. S. M. dos Santos, A. C. Beltrão, B. P. de Souza ja G. H. Travassos, ”On the benefits and challenges of using kanban in software engineering: a structured

synthesis study", *Journal of Software Engineering Research and Development*, vol. 6, 1 joulukuu 2018. DOI: 10.1186/s40411-018-0057-1.

[21] J. Melegati, R. Chanin, A. Sales, R. Prikladnicki ja X. Wang, "MVP and experimentation in software startups: A qualitative survey", *Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020*, s. 322–325, elokuu 2020. DOI: 10.1109/SEAA51224.2020.00060.

[22] "A Systematic Mapping Study on the Use of Software Engineering Practices to Develop MVPs", *Proceedings - 2021 47th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2021*, s. 62–69, syyskuu 2021. DOI: 10.1109/SEAA53835.2021.00017.

[23] E. Klotins, M. Unterkalmsteiner ja T. Gorschek, "Software engineering anti-patterns in start-ups", *IEEE Software*, vol. 36, s. 118–126, 2 maaliskuu 2019, ISSN: 19374194. DOI: 10.1109/MS.2018.227105530.