Ketterän kehityksen menetelmien erot ja soveltuvuus erilaajuisiin projekteihin

Turun yliopisto
Tietotekniikan laitos
TkK-tutkielma
Tietotekniikka
Maaliskuu 2024
Juuso Korsimo

TURUN YLIOPISTO
Tietotekniikan laitos

Juuso Korsimo: Ketterän kehityksen menetelmien erot ja soveltuvuus erilaajuisiin projekteihin

TkK-tutkielma, 30 s.
Tietotekniikka
Maaliskuu 2024

Ketterä kehitys on tärkeässä roolissa nykypäivän ohjelmistotuotannossa sekä muissa kehitystyötä hyödyntävissä projekteissa. Ketterän kehityksen periaatteesta on luotu kehitysmenetelmiä, jotka laativat niiden käyttäjille yleisohjeet toimintatavoista ja ohjaavat projektin etenemistä omalla tavallaan. Tutkielman tutkimuskohteina ovat kaksi ketterän kehityksen menetelmää: Scrum ja Kanban. Tämän kandidaatintutkielman aiheena on tutkia edellä mainittuja ketterän kehityksen menetelmiä, löytää ja vertailla niiden vahvuuksia ja heikkouksia, tutkia miten ne toimivat erikokoisissa projekteissa ja muodostaa johtopäätöksistä optimaalinen toimintaympäristö molemmille menetelmille.

Tutkimus on suoritettu kirjallisuuskatsauksena. Tuloksien mukaan molemmat menetelmät ovat päteviä kehitysmenetelmiä ohjelmistotuotannon käyttöön. Menetelmien väliset erot perustuvat työnkulun järjestelyyn, järjestettävien kokousten lukumäärään ja kehitystiimien rakenteisiin. Tutkimuksen mukaan Kanban toimii paremmin pienissä, nopean aikataulun projekteissa, kun työryhmässä tapahtuu useita muutoksia tai ryhmä ei pysy vakiona pitkän aikaa. Scrum toimii paremmin 5-10 henkilön tiimeistä koostuvista keskikokoisista projekteista, joissa vaaditaan tiimin laajaa, jopa poikkitieteellistä, osaamista. Tulosten mukaan haasteena molemmille menetelmille on kehittäjien aikaisemman kokemuksen puute ja tietämättömyys ketterien menetelmien käytöstä. Tulosten perusteella menetelmien kouluttamisessa olisi parannettavaa paremman työvalmiuden saavuttamiseksi, mikä tekee tämän tutkimuksen aiheesta tärkeän.

Asiasanat: Ketterä kehitys, Scrum, Kanban, Ohjelmistotuotanto, Ohjelmistokehitys

Sisällys

1 Johdanto

Ohjelmistokehityksen aikana törmätään usein haasteeseen, jossa kehitettävä projekti pitäisi tuottaa asiakkaalle mahdollisimman nopeasti, laadukkaasti ja pienillä kustannuksilla [1]. Kehitystyö vaatii ryhmän innovatiivisia henkilöitä, joiden yhteistyö on saumatonta ja tavoitteellista, mutta myös projektin kehityskulkua ohjaavan menetelmän. Ketterä kehitys on luotu vastaamaan tähän haasteeseen. Kehitystyön ketteryyden avulla työ reagoi ympärillä tapahtuviin muutoksiin, joilla on vaikutus kehitettävän projektin tarpeisiin [2]. Ketterän kehityksen menetelmät pyrkivät optimoimaan kehitystyön edistymistä ja halutun lopputuloksen saavuttamista tarjoamalla rakenteen ryhmän kommunikoinnille, kehityksen työvaiheille sekä työn järjestelmälliselle etenemiselle [2].

Tässä työssä tutkitaan ketterien menetelmien soveltuvuutta erikokoisiin projekteihin. Ketterän kehityksen menetelmiä on paljon ja tässä tutkielmassa menetelmistä on valittu tutkimuskysymyksien mukaiseen tarkasteluun kaksi. Muut menetelmät ovat rajattu tutkimuksen ulkopuolelle. Scrum ja Kanban ovat tutkimuksessa käsiteltävät ketterän kehityksen menetelmät, mutta myös ulkopuolelle rajattuja menetelmiä esitellään lyhyesti. Tutkimuskysymykset ovat:

- TK 1: Mitkä ovat Scrumin ja Kanbanin keskeiset piirteet, vahvuudet, heikkoudet sekä niiden väliset erot?
- TK 2: Miten Scrum ja Kanban toimivat erikokoisissa projekteissa?
- TK 3: Millainen on optimaalinen toimintaympäristö Scrumille ja Kanbanille?

Tiedonhaku on suoritettu Web of Science-, IEEE Xplore-, Google Scholar -tietokannoista, sekä Googlesta. Tiedonhakua on suoritettu käyttämällä hakusanoja: ”(Scrum OR Kanban) AND Software development”, ”(Scrum OR Kanban) AND Software engineering”, ”Agile AND (Scrum OR Kanban)”, ”(Scrum OR Kanban) AND Challenges”, ”(Scrum OR Kanban) AND Large scale”, ”(Scrum OR Kanban) AND Scaling”. Aineistojen kerääminen aloitettiin etsimällä Scrumia ja Kanbania ohjelmistokehityksessä käsitteleviä aineistoja. Hakusanoja tarkennettiin edellä mainittuihin, etsiessä tiettyyn aihealueeseen liittyvää aineistoja. Aineistojen valinnassa otettiin ensiksi huomiota otsikkoon sekä hakusanan palauttaneeseen kohtaan. Otsikon arvioinnin jälkeen aineisto joko eliminoitiin tai valittiin tarkempaan tarkasteluun tiivistelmän ja koko tekstin arviointiin. Menetelmien toimivuuden vertailuun valikoitiin kokeellisina- ja tapaustutkimuksina Scrumin ja Kanbanin vahvuuksia ja heikkouksia tutkivat aineistot. Tutkimustulokset ovat koostettu aineistoissa esiintyvistä, toistuvista asioista.

Tutkielman toisessa luvussa ”Ketterä kehitys” perehdytään lyhyesti ketterän kehityksen historiaan ja esitellään tutkimukseen valitut ketterän kehityksen menetelmät Scrum ja Kanban. Menetelmien esittelyssä syvennytään niiden keskeisimpiin piirteisiin, työvirran etenemisen prosessiin ja käytäntöihin. Kolmannessa luvussa ”Menetelmien vertailu” tarkastellaan edellä mainittujen menetelmien vahvuuksia, heikkouksia, eroja sekä toimivuutta yrityskoon muuttuessa. Neljäs luku ”Analyysi ja pohdinta” sisältää näkökulmia menetelmien toiminnallisuutta edistävälle toiminnalle. Viidennessä luvussa ”Yhteenveto” tulevat esille tutkielman havainnot ja vastaukset tutkielman tutkimuskysymyksiin.

2 Ketterä kehitys

Ketterä liike (*engl.* Agile movement) lähti leviämään 2000-luvulla uutena ohjelmistotuotannon prosessien ohjausmallina vastauksena uusiin tarpeisiin saada kehitysprosessista kevyempi, nopeampi ja muutoksiin reagoiva [2]. Varsinkin 2000-luvun alkupuolella suosioon nousseet internet- ja mobiilisovellukset loivat tarpeen uusille kehitysmenetelmille, jotka kykenevät tarvittaessa mukautumaan suunnitelmien muutokseen kesken kehityksen [2]. Ketteryyden on tarkoitus mahdollistaa nopea reagointi muuttuvaan ympäristöön, käyttäjien tarpeisiin ja tiukempiin aikatauluihin [1]. Perinteiset ohjelmistokehityksen menetelmät, kuten vesiputousmalli, puolestaan seuraavat tiukasti projektille laadittua aikataulua ja sen vaiheita [3]. Näiden perinteisten menetelmien mukaan kaikki tulevat ongelmat ovat ratkaistavissa ja jokainen ongelma on ratkaistavissa optimaalisella ja ennalta odotettavalla ratkaisulla [1].

Joukko ohjelmistokehittäjiä kokoontui vuonna 2001 luomaan yhdessä Ketterän manifestin (*engl.* Manifesto for Agile Software Development), joka sisältää olennaiset arvot ja viiteohjeet ketterää kehitystä toteuttavalle toiminnalle [1]. Ketterän manifestin neljä tärkeintä arvoa ovat: [4]

1. Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja (Individiuals and interactions over processes and tools)

2. Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota (Working software over comprehensive documentation)

3. Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja (Customer collaboration over contract negotiation)

4. Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa (Responding to change over following a plan)

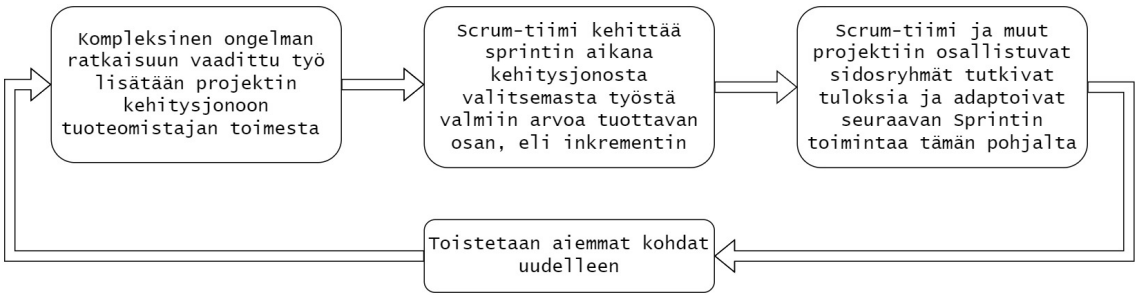
Manifestin neljässä arvossa esitetään kaksi arvoa tuovaa asiaa, joista manifesti erityisesti painottaa ensimmäisenä esitellyn asian olevan arvokkaampi [4]. Ensimmäiset arvot edustavat ketterää mallia ja jälkimmäiset perinteisen ohjelmistokehityksen mallia. Ketteriä ohjelmistokehityksen menetelmiä on monia, mutta yleisimmin ketteriksi menetelmiksi listataan Extreme Programming (XP), Scrum, Feature-Driven Development (FDD), Adaptive Software Development (ASD), Crystal ja Lean Software Development (LD) [5].

Ketterät ohjelmistokehityksen menetelmät vaativat tietynlaisen ympäristön toimiakseen tehokkaasti. Yleensä ketterän kehityksen aiheuttamat haasteet periytyvät seuraavista tekijöistä: organisaatiosta, ihmisistä ja prosessista [5].

2.1 Scrum

Scrum on ketterän ohjelmistokehityksen menetelmä, jonka perustana on kehitystiimin hallinnoiminen toimimaan joustavasti muuttuvassa ympäristössä [2]. Scrumin toiminta pohjautuu iteraatioiden kautta inkrementaaliseen etenemiseen, tiiviiseen kommunikaatioon ja monitieteellisen osaamiseen Scrum-tiimin sisällä [6].

Scrumin teorian kolme kulmakiveä ovat: läpinäkyvyys, tarkastelu ja mukauttaminen [6]. Menetelmän mukaan ihmisistä muodostetaan ryhmiä, jotka sisältävät ryhmänä kaikki työn suorittamiseen vaaditun asiantuntijuuden sekä taidot. Ryhmät jakavat taitoja jäsentensä kesken tai hankkivat vaaditut taidot tarvittaessa muulla keinolla. Läpinäkyvyys Scrumissa tarkoittaa kehitettävän työn tilannekuvan tiedostamista ryhmän sekä muiden sidosryhmien toimesta. Tarkastelun avulla voidaan

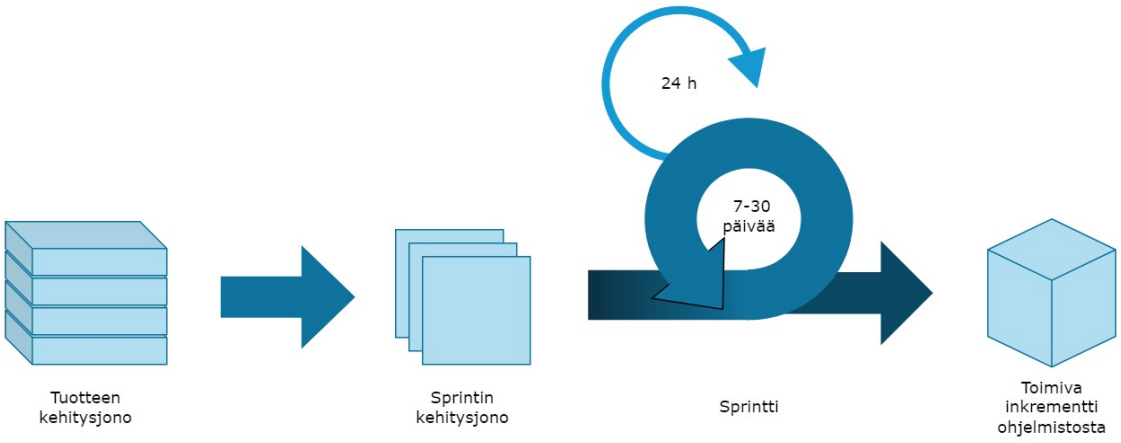


Kuva 2.1: Havainnekaavio Scrumin toiminnan pääpiirteistä [6].

ratkaista kehitystyölle haitallisia ongelmakohtia toimeenpanemalla ryhmän keskustelun ja reflektion pohjalta vaadittuja muutoksia. Mukauttaminen ohjaa ryhmää tekemään tarkastelun perusteella tehtävät muutokset poikkeamien minimoimiseksi ja täten parantaa tiimin sekä menetelmän yhteistä toimivuutta. [6]

Scrumin toiminta yksinkertaistettuna perustuu Scrum Masterin luomaan kehitysympäristöön, jonka toiminnan pääpiirteet ovat esitetty kuvassa 2.1. Taustalla on myös viisi kehittäjien menestyksekkäälle toiminnalle tärkeää arvoa: sitoutuminen, keskittyminen, avoimuus, kunnioitus ja rohkeus. Scrumin rakenne on pääpiirteiltään hyvin yksinkertainen, eikä se aseta tiukkoja rajoitteita Scrum-tiimin toiminnalle. Sääntöjen avulla menetelmä ohjaa tiimiä toteuttamaan Scrumin teoriaa ja hallinnoi käyttäjien yhteistyötä ja kommunikointia. Scrumin prosessit itseohjautuvasti mukautuvat menetelmää käyttävissä tiimeissä heidän tarpeidensa mukaiseksi tehokkuuden edistämiseksi. [6]

Scrum-tiimi on jaoteltu kolmeen päärooliin: 1. tiimiin (*engl.* The Team), 2. tuoteomistajaan (*engl.* Product Owner) ja 3. Scrum Masteriin [7]. Scrum Master vastaa asiakkaan kanssa kommunikoinnista, projektin suunnitelmallisesta ja tehokkaasta etenemisestä ja Scrumin periaatteiden noudattamisesta tiiminsä sisällä [2]. Asiakas, Scrum Master ja muut sidosryhmät valitsevat henkilön tuoteomistajan rooliin. Kokonaisuudessaan Scrum toimii parhaiten pienien tiimien kanssa ja suositus tiimin



Kuva 2.2: Havainnekaavio Scrumin toiminnan prosessista (perustuu kuvaan [8]).

koosta on alle 10 henkeä [2]. Jos tiimin osallistujamäärä on yli 10, tulisi muodostaa useampi Scrum tiimi ja jakaa osallistujat usean tiimin kesken [2].

Scrumissa on kolme vaihetta: ennen kehitystä, kehityksessä ja kehityksen jälkeen [2]. Ensimmäisessä vaiheessa, eli ennen kehitystä, luodaan tuotteen kehitysjono (*engl.* Product Backlog), joka sisältää kaikki valmiille tuotteelle sillä hetkellä tiedossa olevat vaatimukset [2]. Tuotteen kehitysjono järjestellään ja priorisoidaan siten, että eniten arvoa asiakkaalle tuottavat ominaisuudet ja ohjelmiston osat nostetaan prioriteettijonon ensimmäisiksi [7]. Lisäksi tuotteen kehitysjonossa oleville työtehtäville asetetaan tehtäväkohtaiset tavoitteet kyseisen tehtävän valmistumisen ajankohdalle [7]. Projektin aikana tuotteen kehitysjonoa päivitetään vastaamaan ajantasaisia vaatimuksia, muuttuneita prioriteetteja ja aikataulutusta tarkennetaan projektin edetessä [2].

Scrumin kehitysvaihe on iteratiivinen vaihe, joka toteuttaa ketterälle kehitykselle ominaiset piirteet [2]. Scrumin iteraatiot eli Sprintit ovat määrätyn mittaisia jaksoja, joiden kesto on yhdestä viikosta yhteen kuukauteen [2]. Scrumin etenemisen vaiheet ovat havainnollistettu kuvassa 2.2. Sprinttien aikana syntyy aina projektin etenemiseen vaadittu esivaatimus, osa tai ominaisuus valmiista lopputuotteesta [7]. Uuden

Taulukko 2.1: Esimerkki Scrumin Sprintin suunnittelusta [6], [2].

Kysymys	**Vaikutus**
Mikä tästä Sprintistä tekee arvokkaan asiakkaalle	Tuoteomistaja ehdottaa muulle Scrum-tiimille millä asioilla olisi mahdollista tuottaa ohjelmistotuotteen asiakkaalle arvoa. Tämän jälkeen koko Scrum-tiimi päättää yhdessä Sprintin tavoitteen, jonka tavoitteena on kertoa sidosryhmille miksi tämä Sprintti on heille arvokas.
Mitä Sprintin aikana on mahdollista saada valmiiksi	Tuoteomistaja ja kehittäjät valitsevat Sprintin kehitysjonoon työtehtäviä tässä vaiheessa pidetyn keskustelun pohjalta. Tässä vaiheessa Scrum-tiimi myös tarkentaa kehitysjonon kohtia tarpeen vaatiessa selventääkseen tavoitetta ja parantaakseen ennustettavuutta.
Miten Sprinttiin valitut työtehtävät saadaan toteutettua	Kehittäjät suunnittelevat ja toteuttavat työn itsenäisesti. Kehitysjonoon lisättyjen työtehtävien suorittamisen kesto arvioidaan ja merkitään työtehtävän rinnalle. Pitkät työtehtävät pyritään pilkkomaan pienempiin palasiin, joiden kesto olisi enintään yhden työpäivän.

Sprintin alkaessa tiimi kokoontuu kaksivaiheiseen Scrum Masterin pitämään Sprintin suunnittelukokoukseen (*engl.* Sprint Planning) keskustelemaan tulevan Sprintin tavoitteesta, kehitettävistä ominaisuuksista ja niiden toteutuksesta [2], [7]. Kokouksen ensimmäisessä osassa mukana on Scrum Master, asiakas, käyttäjät, johtoryhmä, tuoteomistaja ja Scrum-tiimi [2]. He päättävät, mitä asioita tässä Sprintissä tulisi saada valmiiksi, milloin tuotteen kehitysjonosta valitaan nämä asiat ja siirretään suunniteltavan Sprintin kehitysjonoon (*engl.* Sprint Backlog) [2]. Sprintin kehitysjono koostuu työtehtävistä ja niiden arvioidusta kestosta, jotka tulisi saada tehdyksi Sprintin aikana [2]. Työtehtävien keston arviointi on usein uudelle Scrum-tiimille haastavaa, mutta ajan myötä tiimi oppii tunnistamaan oman kehitystyön nopeuden ja kapasiteetin [6]. Sprintin suunnittelukokouksen etenemisessä usein käytetyt kysymykset ja niiden vaikutukset ovat esitelty taulukossa 2.1.

Sprinttien aikana tiimi kokoontuu päivittäin lyhyisiin, noin 15 minuutin päivittäispalavereihin (*engl.* Daily Scrum), joissa jokainen tiimin jäsen kertoo oman tilanteensa [2]. Päivittäispalaverit järjestetään yksinkertaistamisen ja rytmittämisen

vuoksi Sprintin jokaisena työpäivänä samassa paikassa ja samaan aikaan [6]. Yleensä nämä palaverit järjestetään etäpuheluna ja aamuisin. Päivittäispalaverissa tiimin jokaiselle jäsenelle esitetään samat kolme kysymystä, joihin jokainen vastaa omalla vuorollaan: mitä olet tehnyt viime päivittäispalaverin jälkeen, mitä olet tekemässä tämän ja seuraavan päivittäispalaverin välissä ja onko jotain haasteita, jotka estävät Sprintin tai projektin suunnitelmallisen etenemisen [7]. Näiden palaverien tarkoitus on vähentää muiden kokouksien tarvetta ja sopia tiimin välisistä palavereista ongelmien ratkaisua varten [7]. Sprintin edistymisen seurantaan käytetään edistymiskäyriä, jotka mittaavat jäljellä olevien työtehtävien lukumäärää [9]. Edistymiskäyrän kuvaajassa vaaka-akselilla on aika ja pystyakselilla työtehtävien lukumäärä. Kuvaajaan piirretään ideaalista tilannetta kuvaava käyrä ja Sprintin edetessä reaalitilannetta vastaava käyrä.

Ennen jokaisen Sprintin päättymistä pidetään Sprintin katselmointi (*engl.* Sprint Review) ja Sprintin retrospektiivi (*engl.* Sprint Retrospective). Sprintin katselmoinnissa pääosallisina ovat Scrum Master sekä Scrum-tiimi ja katselmointia seuraavat sivullisina: asiakas, johtoryhmä ja tuoteomistaja. Scrum Master ja Scrum-tiimi esittävät, mitä Sprintin aikana saatiin valmiiksi ja mitä seuraavaksi suoritetaan ja toteutetaan [2]. Tämän kokouksen aikana saattaa tulla esille uusia kehitettäviä ominaisuuksia, joita lisätään tuotteen kehitysjonoon tai jopa muutetaan koko projektin kehityksen suuntaa [2]. Sprintin retrospektiivin on tarkoitus kehittää Scrum-tiimin toimintaa tunnistamalla asioita, jotka hidastivat toimintaa Sprintin aikana ja pohditaan sekä toimeenpannaan tarvittavia muutoksia tehokkuuden parantamiseksi [6].

2.2 Kanban

Kanban on ketterän ohjelmistokehityksen menetelmä, jonka perustana on projektin työtehtävien hallinnointi ja jatkuva kehityksen eteneminen [10]. Kanbanin tavoitteena on saada aikaiseksi vakaa ja tuloksia tuottava tuotanto, joka pystyy reagoimaan

mahdollisesti muuttuviin vaatimuksiin [11], [12]. Keskeinen menetelmän toimivuuden pääpiirre on tasaisen työnkulun saavuttaminen ja sen ylläpitäminen [10].

Kanban-menetelmä on peräisin 1950-luvun Japanista, jossa menetelmää käytettiin autoteollisuudessa työvaiheiden aikataulutuksen hallintaan [11]. Kanban-menetelmän toiminta perustuu visuaaliseen prosessin hallintaan, joka pyrkii vähentämään hidastavia tekijöitä, hukkaa ja odotusaikoja [12]. Tätä menetelmää David J. Anderson on käyttänyt ensimmäinen kerran ohjelmistotuotannossa vuonna 2004 [12] ja laajempaan julkisuuteen menetelmä on tullut hänen vuonna 2010 julkaiseman kirjan: ”Kanban: Successful Evolutionary Change for Your Technology Business” myötä [13]. Andersonin mukaan ketterän Kanbanin viisi pääpiirrettä ja niiden vaikutukset ovat esitelty taulukossa 2.2

Taulukko 2.2: Kanbanin viisi tärkeintä pääpiirrettä [11], [12].

Pääpiirre	**Vaikutus**
Rajoittaa yhtäaikaisen tekemisen määrää (*engl.* Limiting work in progress)	Rinnakkaisten tehtävien määrän rajoittaminen jokaisessa Kanban-taulun sarakkeissa. Johtaa tehtävien nopeampaan valmistumiseen ja saa projektin etenemään tasaisessa tahdissa.
Havainnollistaa työjonoa (*engl.* Visualizing workflow)	Kanban-taululta nähdään yhteistyö, työjono, viivästymiset, mitkä ovat tärkeitä tietoja kaikille projektissa mukana oleville tiimin jäsenille ohjelmistotuotannossa.
Mitata ja hallita etenemistä (*engl.* Measuring and managing the flow)	Tasainen tahti on projektin etenemiselle tärkeää ja se toimii hyvänä vetäjänä tiimin kehitykselle.
Työ prosessien määrittely (*engl.* Making process policies explicit)	Projektin etenemistä varten määritellään eksplisiittiset strategiat, jotka muodostavat raamit itse kehitykselle. Nämä eivät ole kuitenkaan sääntöjä, joita tulisi täysin noudattaa, vaan ainoastaan politiikka, johon tulisi viitata.
Kehityskohteiden tunnistaminen mallien avulla (*engl.* Using models to recognize improvement opportunities)	Käytetään toiminnasta kehitettäviä osa-alueita tunnistavia malleja ja toimitaan niiden ehdottamalla tavalla, jotta kehitystyö olisi tehokkaampaa.

Kanban-taulu ja siihen kiinnitetyt kortit toimivat Kanban-menetelmän havainnoinnin keskeisimpänä asiana, minkä päälle kaikki menetelmän hallinnointi rakentuu. Yksinkertainen Kanban-taulu ja siinä yleisimmin esiintyvät rakenteet ovat havainnollistettu taulukossa 2.3. Kanban-taulu on jaettu useampaan sarakkeeseen, jotka havainnollistavat aina jotakin tiettyä työvaihetta ja määrittelevät sarakekohtaisen rajoituksen korttien yhtäaikaiselle määrälle kyseisessä sarakkeessa [12]. Sarakkeiden rajoitukset määräytyvät tiimin koon mukaan, joka toimii vastapainona projektin valmistumistavoitteille [11]. Taulukossa 2.3 sarakerajoitukset ovat ilmaistu sarakkeen otsikon jälkeen suluissa numeroilla. Taululla esiintyviin sarakkeisiin asetetaan kortteja, jotka edustavat jotain kehitettävän ohjelmistotuotteen ominaisuutta tai muuta työtehtävää [12]. Kortit siirtyvät taulun sarakkeissa vasemmalta oikealle niiden kehitystilanteen edetessä. Taulun vasemmassa reunassa on kehitysjono-sarake, joka sisältää valmiille ohjelmistotuotteelle sidosryhmien vaatimuksien mukaiset työtehtävät kortteina [14]. Prosessin edetessä kortit siirtyvät kehityksen mukana seuraavaan sarakkeeseen, mutta ylittämättä sarakkeiden määrättyä rajoitusta [14]. Kun yksittäinen kortti on käynyt kaikki työvaiheet läpi, se siirtyy taulun oikeassa reunassa olevaan sarakkeeseen, joka sisältää kaikki valmistuneiden työtehtävien kortit [14]. Kanban-taulusta nähdään, missä vaiheessa ohjelmistoprosessia mennään, tällä hetkellä kehityksessä olevat työtehtävät ja tehtävien prioriteetit [11]. Edellä mainittuja kohtia hyväksi käyttäen voidaan tunnistaa esimerkiksi työnkulun ongelmakohdat. Menetelmän työnkulun aktiiviseen hallinnointiin kuuluu sarakerajoitukset, työtehtävien kasaantumisen estäminen työnkulun vaiheissa ja jumittuneiden työtehtävien ratkaiseminen [10].

Käytännössä taululla esitettyjen työtehtävien kesto tulee arvioida ja asettaa näkyviin taululle työtehtävän korttiin. Arvioinnissa tulisi esiintyä työtehtävän suorittamiseen vaadittava aika, eli Kanban-taulun kehitysjonosta valitun työtehtävän kortin siirtymiseen kulunut aika taulun viimeiseen sarakkeeseen. Arviossa ilmenee myös

Taulukko 2.3: Yksinkertainen Kanban-taulu, johon on asetettu sarakekohtaiset rajoitteet työn määrälle. Taulukossa esiintyvät kirjaimet edustavat työtehtäviä. (Muokattu lähteen [12] pohjalta)

Kehitysjono	(2) **Analyysi**	(3) **Kehitys**	(2) **Testaus**	**Valmiina**
G	F	E	A	C
H		D	B	
I		J		

prosenttiosuus, joka ilmaisee kuinka suuri osa tehtävästä saadaan edellä mainitussa ajassa valmiiksi. Arvioinnissa käytetään hyväksi menetelmän luomaa dataa ja kehittäjien kokemuksia edeltävien työtehtävien kestosta. [10]

Kanban ei sisällä vakiintunutta hierarkiaa tai rooleja tiimin jäsenille [13], joten menetelmänä tämä voidaan ottaa käyttöön ilman suurempia muutoksia tiimien rakenteisiin. Kanban ei ole rajoittunut vain tiettyyn alaan, vaan menetelmää on mahdollista soveltaa kaikkiin projekteihin alasta riippumatta [10]. Tässä työssä kuitenkin tutkitaan Kanbanin käytettävyyttä ja ominaisuuksia ohjelmistotuotannon näkökulmista.

Kanbanin hyötyihin kuuluu selkeän kokonaiskuvan luominen Kanban-taulun avulla, työprosessin sekä työkuorman tehokas ja helppo hallinnointi [12]. Ohjelmistokehittäjien kohtaamat ongelmat tulevat esille myöskin Kanban-taululla, sillä jos heidän kehityksen alla oleva työtehtävä ei etene, se ei myöskään etene taululla [3]. Jos Kanban-taululta huomataan jonkin kortin olevan pitkän ajan jumissa samassa vaiheessa, ja täten samalla täyttävän sarakkeen yhtäaikaisten työvaiheiden rajoituksen, voivat muut tiimin jäsenet auttaa ongelman ratkomisessa [3]. Sarakkeiden rajoitusten asettaminen on tästä syystä tärkeää, sillä edellisten korttien tulee siirtyä eteenpäin ennen kuin seuraava kortti voidaan vetää seuraavaan sarakkeeseen, ja näin saadaan aikaan koko projektin tasainen edistys [11].

Menetelmän tehokkaan toiminnan takaamiseksi kehityksen aikana kerätään dataa, jota analysoidaan Kanbanissa neljästä eri näkökulmasta erilaisia mittareita hyö-

dyksi käyttäen. Taulukossa 2.4 esitetään Kanbanin neljä tärkeintä etenemisen mittaria, joiden mukaan on mahdollista arvioida sekä parantaa menetelmän toimintaa ja tehokkuutta. [10]

Taulukko 2.4: Kanbanin etenemisen mittarit [10].

Mittari	**Tarkastelukohde**
Kehityksessä olevat työtehtävät (*engl.* Work in Progress)	Kyseisellä ajanhetkellä kehityksessä olevien työtehtävien lukumäärä.
Suoritusteho (*engl.* Throughput)	Valmiiden työtehtävien lukumäärä suhteutettuna tiettyyn ennalta määrättyyn kuluneen ajan yksikköön.
Työtehtävän ikä (*engl.* Work Item Age)	Aika, joka on kulunut työtehtävän aloituksesta kyseiseen ajanhetkeen.
Jakson kesto (*engl.* Cycle Time)	Aika, joka on kulunut työtehtävän aloituksesta sen valmistumiseen.

Kehityksessä olevien työtehtävien lukumäärän mittarin avulla havainnollistetaan tiimin kuormitusta ja sen avulla voidaan esimerkiksi säätää sarakerajoituksia tarpeen mukaisesti. Suoritustehon mittarilla tiimi reflektoi oman kehityksen kapasiteetin, jonka avulla projektin työmäärän suorittamisen kestoa voidaan ennustaa. Työtehtävien ikää mittaavalla mittarilla voidaan tunnistaa mitkä tehtävät tai työnkulun vaiheet ovat pullonkaulana projektin etenemiselle. Jakson keston mittarin avulla tiimin on mahdollista tuottaa tarkempia arvioita uusien työtehtävien kestoksi [15]. Tätä voidaan käyttää hyödyksi jakamalla työtehtävät kuvitteellisiin pisteisiin niiden laajuuksien mukaan, mitata käytännössä keskiarvo yhden pisteen suorittamiselle vaaditulle ajalle ja täten uusien työtehtävien pisteyttämisen jälkeen voidaan ennustaa työtehtävän jakson kesto [15].

3 Menetelmien vertailu

Tämän luvun kappaleissa vertaillaan edellä esiteltyjen ketterien ohjelmistokehityksen menetelmien vahvuuksia, heikkouksia, eroja ja toimivuutta. Vertailun pääpaino on menetelmien vahvuuksissa ja heikkouksissa, niiden välisissä eroissa sekä toimivuuden tarkastelussa erilaajuisissa projekteissa.

Menetelmien vahvuuksia ja heikkouksia tarkastellaan ketterän kehityksen menetelmien onnistumiselle tärkeiden ominaisuuksien mukaan. Projektin onnistumiselle tärkeitä puitteita ovat: nopea kommunikaatio tiimin sisällä, työkalut ja tekniikat, johtajien tuki ja pienet tiimit [16]. Epäonnistumiselle ominaisia piirteitä ovat: koulutuksen puute, ketterän menetelmän huomiomatta jättäminen, epäsoveliaat teknologiat ja työkalut sekä huono asiakassuhde [16].

3.1 Scrumin vahvuudet ja heikkoudet

Scrum kuuluu suosituimpiin ja eniten käytettyihin ketterän kehityksen menetelmiin ja tästä syystä sitä on tutkittu ja kehitetty paremmaksi vuosien varrella. Kehityksen ja uudistusten tuominen julki näkyy esimerkiksi ”www.scrumguides.org” -sivuston tasaisin väliajoin päivittyvästä Scrumin opaskirjasta.

Ohjelmistokehityksen menetelmänä Scrum on ihanteellinen viiden tai useamman henkilön kehitystiimeille. Scrumissa suositeltu tiimin enimmäiskoko on alle 10 henkilöä, mikä mahdollistaa tiiviin kommunikaation kaikkien jäsenten kesken. Sopivan tiimikoon myötä jäsenien osallistumisprosentti projektiin on huomattu kasvavan [7].

Osallistumisprosentin kasvu johtaa yhtenäisempään sekä tehokkaampaan kehitystyöhön, jonka hallinta ja ohjaaminen on vaivattomampaa. Tiimikoon pieni osallistujamäärä ei kuitenkaan rajoita tiimien kokonaislukumäärää ja rinnakkaisia Scrum-tiimejä voikin olla useita. Tällöin järjestetään Scrum of Scrums -tapaamisia, joihin jokaisesta rinnakkaisesta tiimistä valitut henkilöt osallistuvat kertomaan oman tiiminsä ajankohtaisen tilanteen. Projektin kehitysjonon työtehtävien jakoon tulee kiinnittää huomiota rinnakkaisten Scrum-tiimien kanssa ristikkäisen työn estämiseksi. Scrum of Scrums -palavereissa pyritään tunnistamaan tiimien väliset ristikkäiset työtehtävät ja ratkaisemaan nämä [17].

Scrumin teoria estää muutosten tekemisen Sprintin tavoitteeseen kesken Sprintin, mikä rajoittaa huomattavasti tämän menetelmän nopeutta suurempiin muutoksiin reagoimiseen. Sprintin aikana tapahtuvat muutokset kuormittavat Scrum Masteria, jonka tulee ylläpitää ajankohtaista kokonaiskuvaa kehityksen etenemisestä [18]. Sprinttien aikana tapahtuvat muutokset ovat ristiriidassa Sprintin suunnittelun kanssa, jos suunnittelussa Sprintin kehitysjonoon valittuja työtehtäviä muutetaan useaan kertaan [18]. Tällöin Sprintin suunnittelusta saatava arvo ja tehokkuus laskee.

Scrumin toteuttaminen laajassa projektissa tai kehitysympäristössä on haastavaa ja vaatii perusteellista suunnittelua. Projektit, joihin osallistuu suuri määrä kehittäjiä, vaativat toimivan ja koordinoidun pohjan rinnakkaisten Scrum-tiimien väliselle toiminnalle. Monen tiimin välinen yhteistoiminta lisää sidosryhmille taakkaa osallistua useampiin tapaamisiin, jotka vievät aikaa itse kehitykseltä. Yksittäisen kehittäjän läsnäolon vaikutus laskee suuremmassa tiimikokonaisuudessa verrattuna pienempään kehitystiimiin. Suuren tiimikokonaisuuden tiedon ajankohtainen ylläpito muuttuu laajassa ympäristössä haastavaksi. Se heikentää myös Scrumin teorian läpinäkyvyyden toteutumista, jolloin tehtävän työn näkyminen on heikompaa kaikille projektin työntekijöille ja muille sidosryhmille [6].

Taulukossa 3.1 esitellään tämän tutkimuksen lähteissä todettuja positiivisia vahvuuksia Scrumin käytöstä ja taulukossa 3.2 puolestaan menetelmän heikkouksia.

Taulukko 3.1: Scrumin koetut vahvuudet.

Hyöty	**Kokemus**
Yhteistyö	Tiimien yhteistyö on toimivaa odotetun ja laadukkaan lopputuloksen saavuttamiseksi [9], [13], [14].
Tiimikoko	Scrum-tiimit ovat riittävän pieniä, jotta jokaisen ääni tulee esille, mutta riittävän suuria saamaan tarvittava määrä työtä aikaiseksi [9], [13].
Projektin onnistuminen	Kokonaisuutena projektin kehitys mielletään onnistuneeksi [9], [14].
Työnjako	Scrumissa tietyt työtehtävät ovat valmiiksi kohdennettu sen roolien mukaisille henkilöille [9], [13].
Asiakkaan vaatimusten seuraaminen	Kehitystyö perustuu asiakkaan asettamille vaatimuksille ja täten tekee työn suunnittelusta ja kehittämisestä selkeää [9], [14].

Taulukko 3.2: Scrumin koetut heikkoudet.

Heikkous	**Kokemus**
Kokemuksen puute	Osallisilla ei ole riittävää kokemusta menetelmän sulavaa ja tehokasta toimintaa varten [19].
Usean tiimin välinen toiminta	Scrum tiimien ja projektien väliset riippuvuudet tuottavat haasteellisia tilanteita [19], [20].
Läpinäkyvyyden puute	Scrumin vaatima läpinäkyvyys ei toteudu oletetusti [19].
Aikavyöhykkeiden erot	Kokouksien järjestäminen haastavaa, jos tiimin jäsenet työskentelevät eri aikavyöhykkeissä [19].
Reagointi muutokseen	Sprintin laajuus on määritelty ennen sen alkua ja kesken Sprinttiä laajuus ei voi muuttua. Tästä seuraa hitaampi reagointi muutoksiin muihin menetelmiin verrattuna [9].

3.2 Kanbanin vahvuudet ja heikkoudet

Kanban on otettu ohjelmistotuotannon puitteissa käyttöön Scrumia myöhemmin, mutta se on silti laajalti käytetty menetelmä. Kanban tarjoaa itsenäisesti toimivan ja helppokäyttöisen pohjan projektinhallinnalle, mutta menetelmiä tulee myös soveltaa tiimeissä heidän käyttötarkoitukseensa sopivaksi. Kanbanin teorian mukaan sitä jopa kehotetaan käyttämään yhteistyössä muiden menetelmien kanssa työvirran tehokkuuden parantamiseksi [10].

Kanban on menetelmänä hyvin helposti käyttöönotettava ja ei välttämättä vaadi muutoksia tavallisen ohjelmistokehitystiimin hierarkiaan. Marko Ikosen ja kumppanien tapaustutkimus ”On the Impact of Kanban on Software Project Work” käsittelee Kanban-menetelmän toimintaa ohjelmistokehityksessä [3]. Tutkimuksessa nousee esille Kanban-menetelmän toimeenpanon vaivattomuus. Tiimin jäsenillä ei välttämättä tarvitse olla aikaisempaa kokemusta tai tietoja Kanbanin toiminnan periaatteista, sillä nämä ovat nopeasti ja helposti omaksuttavissa [3]. Menetelmän yksinkertaisen käyttöönoton ja perusteiden sisäistämisen avulla sillä on suurempi todennäköisyys tuottaa kehitettävästä projektista laadukas lopputulos.

Kanbanissa päivittäisiä kokouksia ei järjestetä, mutta kommunikaation koetaan silti toimivan tehokkaasti tätä menetelmää käyttävän tiimin sisällä [10], [11]. Kanban-taululta on mahdollista tunnistaa ongelmista kärsivät työtehtävät, jos niiden kortit eivät etene keskimääräisellä tahdilla. Tällöin tiimin sisällä on mahdollista kysyä apua muilta kehittäjiltä, ratkaista ongelma yhdessä ja täten edistää taas koko projektin kehitystyötä. Ongelman ratkaisussa menetelmää käyttävät kehittäjät ovat löytäneet ongelmakohdat ripeästi ja niiden ratkominen on aloitettu heti niiden ilmentymisen jälkeen [12].

Kommunikaation koetaan toimivan myös kehitystiimin ulkopuolisien osapuolien kanssa hyvin. Tämä ilmenee tutkimuksien tuloksissa kohonneena asiakkas tyytyväi-

syytenä [11], varhaisempana asiakaspalautteena [11] ja kaikkien projektin sidosryhmien välisen kommunikaation parantumisena [11].

Taulukossa 3.3 esitellään tämän tutkimuksen lähteissä todettuja positiivisia vahvuuksia Kanbanin käytöstä ja taulukossa 3.4 puolestaan menetelmän heikkouksia.

Taulukko 3.3: Kanbanin koetut vahvuudet.

Hyöty	**Kokemus**
Kommunikaatio	Kommunikaatio hoituu tehokkaasti, sitä tapahtuu riittävästi [3], [12] ja myös kaikkien sidosryhmien välillä [11].
Jaksojen kestot	Työtehtävät suoritetaan valmiiksi ohjelmistotuotteeksi tehokkaasti [11], [13], [15].
Tiimin mielipiteet ja tyytyväisyys	Työssä mukana olevat kehittäjät kokevat menetelmän intuitiiviseksi [12], [14] ja heidän motivaation on tutkittu kasvavan Kanbania käyttäessä [11].
Ongelmien ratkaisu ja palaute	Kehitystyön aikana kohdattavat ongelmat löytyvät nopeasti ja niitä voidaan aloittaa ratkaisemaan välittömästi [3], [12], [14]. Asiakaspalaute on myös koettu tulevan aikaisemmassa vaiheessa perille kehitystiimille Kanbanin avulla [11].
Visualisointi	Eteneminen ja tilannekuva on havainnoitavissa Kanban-taululta [3], [14].

Taulukko 3.4: Kanbanin koetut heikkoudet.

Heikkous	**Kokemus**
Ei sisällä yksin kaikkea mitä projektinhallintaan tarvitaan	Kanban tarvitsee muita ketterän kehityksen menetelmiä rinnalle tuekseen [11], [12].
Nykyinen kehitykulttuuri	Nykyisiä organisaation rakenteita ja kehitysmenetelmiä ei haluta muuttaa [11].
Sarakerajoitusten määrittäminen	Aluksi sarakerajoituksia on vaikea määrittää ja niiden säätäminen heikentää kehitystyön tehokkuutta ja vie aikaa tiimiltä [11], [12], [20].
Tavoiteen näkyvyys	Projektin tavoite ei tule vahvasti esille Kanbania käyttäessä [14].
Kokemuksen puute	Menetelmää ei tunneta, käyttökokemuksen ja koulutuksen puute [11].

3.3 Scrumin ja Kanbanin erot ketterän kehityksen menetelminä

Kaikki ketterän kehityksen menetelmät seuraavat pääpiirteiltään ketterän kehityksen manifestin määrittelemiä ohjeita [2], mutta menetelmien välillä on tästä huolimatta huomattavia eroavaisuuksia. Tässä kappaleessa perehdytään Scrum ja Kanban-menetelmien välisiin eroihin ohjelmistotuotannossa.

Scrum laatii tiimin jäsenille roolit ja viitteet, miten tiimin jäsenten tulisi toimia projektin aikana [2]. Kanban toimii tuotantoketjun visualisoijana ja optimoijana, jonka pohjalta projektissa työskentelevän tiimin on helpompi saada kokonaiskuva projektin vaiheista ja niiden etenemisestä [12]. Scrum ja Kanban lähestyvät ketterän kehityksen raameja eri näkökulmista. Scrum perustuu enemmän tiimin hallintaan, kun Kanban taas työnkulun hallintaan.

Vertailtavien menetelmien välillä yksi suurimpia eroja on projektiin osallistuvan tiimin henkilömäärä ja siihen liittyvät rajoitteet. Scrumin periaatteet vaativat noin 4-10 henkilön kompaktin ja tiiviin työtiimin [2]. Pieni tiimikoko mahdollistaa kattavan kommunikaation jokaisen tiimin jäsenen välillä ja kokonaisuutena pienen tiimin hallinnoiminen on tehokkaampaa. Laajat Scrumia hyödyntävät projektit sisältävät useampia Scrum-tiimejä ja tiimit kommunikoivat vain kehitysvaiheen pääpiirteet toistensa välillä niin sanotuissa Scrum of Scrums -tapaamisissa [17]. Kanban-menetelmässä ei ole yleisiä rajoitteita tiimin henkilömäärään liittyen, ja tämän takia se on menetelmänä vaivattomasti käyttöönotettavissa mihin vain projektiin tiimikoosta välittämättä.

Valitut menetelmät lähestyvät ohjelmistoprojektin etenemistä erilaisilla ratkaisuilla. Scrumissa tuotteen kehitysjonosta valitaan Sprinttien suunnitteluvaiheessa kehittäjien sekä tuoteomistajan ohjaamana Sprintin kehitysjonoon työtehtäviä, jotka ovat mahdollista toteuttaa valmiiksi kyseisen Sprintin aikana [6]. Suunnittelu-

vaiheen jälkeen kehittäjät tulevat itse valitsemaan Sprintin kehitysjonosta itselleen työtehtäviä. Sprintit jaksottavat Scrumia käyttävän projektin etenemisen inkrementteihin, joiden kestot ovat ennalta määrättyjä ja yleensä kesto on viikon ja kuukauden väliltä [2]. Kanbanissa projektin kehityksen eteneminen on jatkuvaa. Tämä tarkoittaa uuden työtehtävän aloittamista sen ollessa mahdollista [10]. Kanbania käyttäessä projektissa työskentelevillä kehittäjillä voi olla vain ennalta määrätty lukumäärä työtehtäviä kehityksessä [12]. Työtehtävän valmistuttua kehittäjä valitsee itselleen uuden työtehtävän kehitysjonosta ja näin muodostuu Kanbanin jatkuva eteneminen [3].

Scrumissa Sprinttien kesto on ennalta määrätty ja tiimi määrittää Sprintin työjonon laajuuden itse perustuen aikaisempiin kokemuksiin tiimin tehokkuudesta [7]. Samanaikaisesti kehityksessä olevien työtehtävien lukumäärää Scrum ei rajoita, toisin kuin Kanban. Kanbanissa sarakerajoitukset määrittävän aktiivisessa kehityksessä olevien työtehtävien lukumäärän työprosessin eri vaiheissa [10]. Sarakerajoitusten tarkoitus on keskittää kehitys tiettyihin työtehtäviin, tunnistaa työnkulussa tapahtuvat pullonkaulat ja luoda kehityksen etenemisestä jatkuvaa [10].

Sprintin suunnittelussa määritellään Sprintin tavoite, jonka kehittäjät sitoutuvat toteuttamaan Sprintin päätyttyä. Sprintin tavoite tulee aina saavuttaa tai vaihtoehtoisesti hylätä tuoteomistajan käskystä, mikäli Sprintin aikana tapahtuneiden muutoksien myötä tavoite on muuttunut tarpeettomaksi [6]. Kanbanissa tehtävien prioriteetteja voidaan vaihtaa milloin vain ja tämä tarjoaa nopean reagoinnin kehityksen aikana ilmeneviin muutoksiin. Nopea reagointi voi kuitenkin myös aiheuttaa liiallista kehitettävän kohteen vaihtelua, jolla on haitallinen vaikutus työtehtävän läpimenoaikaan [11].

Tutkimuksessa vertailtavien menetelmien tiimirakenteissa on huomattavissa merkittäviä eroja. Scrumin tiimissä tulee olla Scrum Master, tuoteomistaja ja kehittäjät. Näillä rooleilla ovat Scrumissa määritellyt työtehtävät ja vastuullisuudet. Kan-

banissa ei ole erityisesti määrättyjä rooleja, eikä niille ole menetelmää hyödyntäessä välttämätöntä tarvetta. Scrum johdattelee ja ohjaa roolien avulla menetelmää käyttävää tiimiä toimimaan roolien periaatteiden mukaisesti, kun Kanban taas antaa vapauden toteuttaa roolit tiimin itse parhaaksi kokemalla tavalla. Scrumissa rooleille jaetut vastuullisuudet auttavat tiimiä tehtävänjaossa, sillä tietyt työtehtävät ohjautuvat itsessään asiasta vastaavalle henkilölle. Toisaalta roolit täytyy nimittää tiimin jäsenille ennen kuin työ voidaan aloittaa. Kanbanin vapaus roolien suhteen helpottaa tiimin muodostamista, mutta hankaloittaa tehtävänjakoa hierarkiarakenteen puuttumisen myötä. Roolien puutteen takia Kanbanissa jokaisen tiimin jäsenellä on vastuu Kanban-taulun tietojen ylläpitämisessä ja päivittämisessä. [6], [10]

Scrumissa läpinäkyvyys mahdollistaa työn etenemisen seuraamisen siihen osallistuvien kehittäjille sekä muille sidosryhmille [6]. Läpinäkyvyyden toteuttaminen Scrumissa on koko Scrum-tiimin vastuulla, minkä myötä on mahdollista huomata merkittäviä vaihteluita projektien tai tiimien välillä [19]. Scrumissa työn vahva läpinäkyvyys parantaa työnlaatua, epäkohtien huomaamista sekä mahdollistaa sidosryhmien palautteen antamisen [19]. Toisaalta heikko läpinäkyvyys taas voi aiheuttaa sidosryhmien tai Scrum-tiimin työn tilannekuvan pois ajan tasalta. Muita tilannekuvaa havainnollistavia rakenteita Scrumissa ovat tuotteen kehitysjono, Sprintin kehitysjono ja edistymiskäyrät. Scrumissa kehitysjonoista ja edistymiskäyristä muodostuva tilannekuva kertoo yleensä projektin etenemisestä Sprintin aikana [3]. Kanban-menetelmän avulla havainnollistettavana ei ole vain pieni osa projektia, vaan kaikki vaiheet ovat esillä Kanban-taululla. Tämän mahdollistaa yksittäisten työtehtävien etenemisen seuraamisen, mutta myös projektin etenemisen kokonaisuutena [3].

Molemmat menetelmät pyrkivät optimoimaan toimintaansa mahdollisimman tehokkaan työvirran saavuttamiseksi. Scrumissa menetelmän optimoiminen tapahtuu Sprinttien jälkeen järjestettävässä Sprintin retrospektiivissä [6]. Scrum-tiimi kokoontuu yhteen keskustelemaan kehitysmenetelmään liittyvien optimointien tarpeista.

Kanbanissa jatkuva työvirran kehitys on yksi menetelmän pääperiaatteista. Kanbanissa voidaan järjestää säännöllisin väliajoin myös koko projektin kesken olevia kokouksia, mutta tämä ei ole aina välttämätöntä [10]. Ohjeen mukaan muutoksia voidaan ottaa käyttöön ilman virallisia tapaamisia [10].

Scrumissa kehitettävä projekti etenee iteratiivisten Sprinttien avulla luoden aina uuden inkrementin ohjelmistoa Sprintin päätyttyä [6]. Sprintin lopputuloksena tulee olla jotain konkreettinen, käytettävä ja kohti projektin tavoitetta vievä osa [6]. Kanbanissa työtehtävä on valmis, kun se on kulkenut Kanban-taulun kaikki sarakkeet läpi. Kanbanissa työtehtävien suorittamiseen kulunutta aikaa mitataan prosessin läpimenoaikana ja pyynnöstä tuotteen toimittamiseen kulunutta aikaa läpimenoaikana [10]. Kanbanissa prosessin läpimenoaika ja läpimenoaika vaihtelevat työtehtävän laajuuden ja mahdollisten ongelmien kohtaamisen myötä, kun Scrumissa valmis inkrementti syntyy aina tasaisin väliajoin Sprintin päättyessä.

Taulukossa 3.5 esitetään kootusti Scrumin ja Kanbanin erojen pääpiirteet ja selitetään molempien menetelmien kohdalla miten eroava piirre on siinä toteutettu.

3.4 Scrumin ja Kanbanin toimivuus erilaajuisissa projekteissa

Ketterän kehityksen menetelmien laajan käytön mukaan niiden tulee sopeutua erilaisiin käyttökohteisiin ja projektiympäristöihin. Laajan projektin kehitystyöhön osallistuu määrällisesti enemmän kehittäjiä sekä muita sidoshenkilöitä verrattuna pienempään projektiin. Tässä kappaleessa tutkitaan Scrumin ja Kanbanin toimivuuden välisiä eroja projektikoon muuttuessa.

Scrumin tiimissä tulisi olla vähintään viisi henkeä [2], joka hieman rajoittaa sen käytettävyyttä pienimmissä projekteissa ja yrityksissä. Kanbanissa ei ole rajoitusta

Taulukko 3.5: Scrumin ja Kanbanin eroavaisuuksien pääpiirteet.

Eroavaisuus	**Scrum**	**Kanban**
Kehitystyön eteneminen	Etenee inkrementaalisissa sykleissä, eli Sprinteissä. Sprintin aikana suoritetaan ennalta valittu määrä työtehtäviä ja niiden julkaisu tapahtuu Sprintin päätyttyä.	Eteneminen on jatkuvaa ja aina työtehtävän valmistuessa siirrytään uuteen.
Tiimikoko	4-10 henkeä, ennalta määritellyt roolit: Scrum Master, tuoteomistaja ja kehittäjät.	Ei rajoitusta tiimin koolle, eikä ennalta määriteltyjä rooleja.
Yhtäaikaisten kehityksessä olevien työtehtävien rajoittaminen	Sprinttiin valitaan tietty määrä työtehtäviä, mutta yhtäaikaisten työtehtävien kehittämiselle ei rajoitusta.	Rajoitetaan yhtäaikaisessa kehityksessä olevia työtehtäviä menetelmän tarvitsevan sujuvan työnkulun takaamiseksi.
Reagointi muutokseen	Ei muutosta tavoitteeseen kesken Sprintin [6].	Muutosten ilmentyessä voidaan muuttaa kehityksen suuntaa heti [10].
Havainnollistavuus	Edistymiskäyrät havainnollistavat Sprintin keskeneräisten työtehtävien lukumäärää [6].	Projektin eteneminen esitetään Kanban-taululla.

tiimin vähimmäiselle koolle [10], minkä takia menetelmä vaikuttaa toimivan paremmin projekteissa, joissa henkilöstöä on vähemmän kuin viisi henkilöä.

Pienten yritysten piirissä Scrumin haasteeksi voi koitua tiimin jäsenten heikko sitoutuminen projektiin osallistumiseen. Yritystoiminnan hallinta vaatii myös työvoimaa ja pienissä yrityksissä työskentelevät työntekijät voivat joutua hoitamaan usean roolin työtä yhtäaikaisesti. Tämän takia joidenkin jäsenien voi olla haastavaa olla paikalla kaikissa kokouksissa ja antaa täyttä sitoutumista projektin kehitykselle. [21]

Suurempien projektien kommunikaation sekä läpinäkyvyyden on koettu heikkenevän Scrumia käyttäessä. Scrum of Scrums -tapaamisissa tiimien osallistujat eivät ole varmoja mitä kaikkea tulisi kertoa, sillä muut tiimit eivät työskentele samojen

työtehtävien piireissä. Eriävien työtehtävien takia tapaamisissa esiintyvät asiat saattavat koskea vain pientä osaa osallistuvista tiimeistä. Scrum of Scrums -tapaamisten osallistuminen tulisi rajata 3–5 tiimille, jotka työskentelevät samojen työtehtävien kanssa tapaamisten tehokkuuden parantamiseksi. [17]

Kanbanin haasteisiin kuuluu myös tiimien maantieteellisen sijainnin ja aikavyöhykkeiden erot. Näiden haasteiden aiheuttamia haittavaikutuksia pyritään minimoimaan esimerkiksi hyödyntämällä fyysisen Kanban-taulun sijaan elektronisia vastikkeita reaaliaikaisen, sijainnista riippumattoman, päivittymisen vuoksi. [20]

4 Analyysi ja pohdinta

Tämä luku sisältää edeltäviin kappaleisiin pohjautuvaa analysointia ja pohdintaa ketterän kehityksen menetelmien käytettävyyden parantamisesta. Menetelmien haittapuolina esiintyi aikaisemman kokemuksen sekä koulutuksen puute (katso luvut: 3.2, 3.4). Tässä luvussa pohditaan ja ehdotetaan tapoja parantaa tietotekniikan alan opiskelijoiden tuntemusta ketterän kehityksen menetelmien käytöstä osana tutkinto opintojaan.

Tietotekniikan opinnoissa opiskelijat tutustuvat ketterän kehityksen menetelmien laajasta kirjosta lähinnä vain Scrumiin. Useamman ketterän menetelmän tuntemus ja aiempi kokemus niiden käytöstä parantaisi opiskelijan valmiutta työelämään siirtyessä. Opintojen aikana ketterän kehityksen menetelmiä voitaisiin hyödyntää ryhmätöiden suorittamiseen sekä järjestää kurssi, jonka aikana opiskelijat tutustuisivat tarkemmin näihin menetelmiin ja hyödyntäisivät niitä käytännön sovelluksissa. Ketterien menetelmien hyödyntäminen ryhmätöissä parantaisi osallisten ymmärrystä työn etenemisen vaiheista, samalla tehostaen työn suorittamista minimoimalla ryhmän jäsenten ohjaamisen puutteen, päällekkäisen työn tekemisen ja ryhmän jäsenten tyhjäkäynnin.

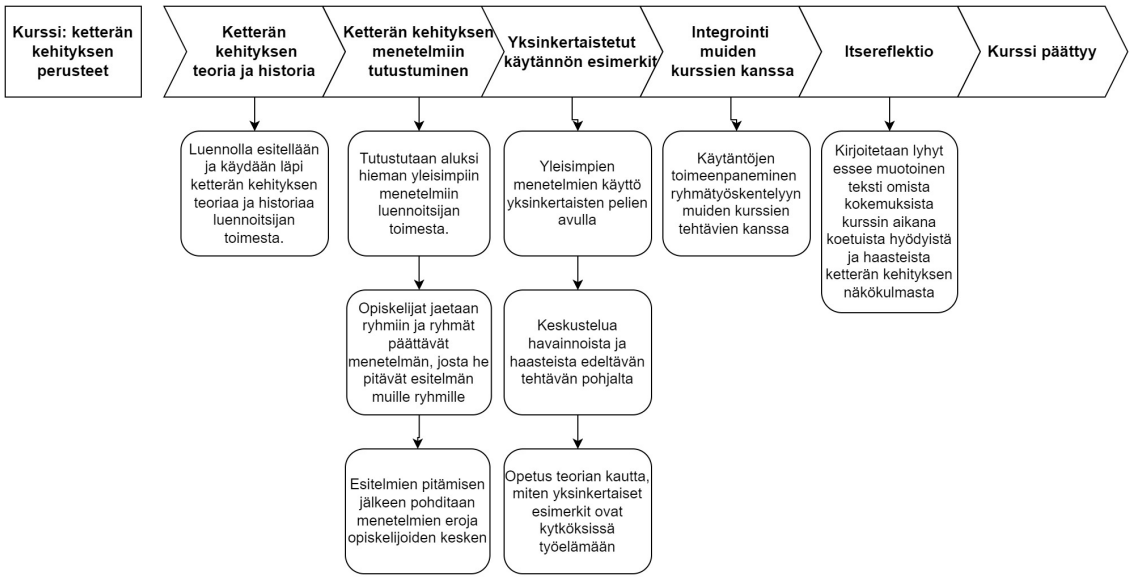
Tässä tutkimuksessa tutkituista menetelmistä Kanban nousee nopeammin käyttöönotettavammaksi, sillä se ei vaadi toimiakseen ryhmälle hierarkiaa tai tiettyjen roolien määrittämistä ja se on sovellettavissa kaikentyyppisiin projekteihin. Opiskelun aikana ryhmätöissä Kanban voitaisiin ottaa käyttöön pitämällä ryhmän jä-

senien kesken palaveri työn suunnittelua varten, jossa käydään läpi työn haluttua sisältöä, ominaisuuksia ja rakennetta. Keskustelun pohjalta lisätään haluttuun lopputulokseen johtavat työtehtävät Kanban-taululle ja suunnitellaan niiden suorittamiseen vaaditut sarakkeet ja niiden rajoitukset. Työtehtävät tulee vielä järjestää tärkeysjärjestykseen ja tämän jälkeen opiskelijat voivat alkaa suorittaa kehitettävää työtään päivittäen edistystä Kanban-taululle. Essee muotoisen tutkimustyön työtehtävät voisivat esimerkiksi olla työssä esiintyvät aiheet ja asiat, jotka halutaan tuoda esille, sarakkeet voisivat taas olla tiedonhaku, suunnittelu, kirjoittaminen ja oikoluku. Virtuaalisen Kanban-taulun käyttö mahdollistaisi ryhmän koordinaation, vaikka sen jäsenet tekisivät työtä itsenäisesti tapaamatta toisiaan.

Scrumin hyödyntäminen opinnoissa olisi työläämpää ja hankalaa kurssien sekä opiskelijoiden vaihdellessa. Scrumin toimiakseen tehokkaasti se vaatii ryhmän, jonka osalliset pysyvät pääosin samana pitkän aikaa. Tämän menetelmän ryhmähierarkia, roolit ja niiden vastuullisuudet vaativat ryhmän jäseniltä niihin perehtymistä. Kurssien verrattain lyhyen keston ja epäsäännöllisten aikataulujen takia Scrumin iteratiivinen rakenne on haasteellinen ajoittaa töiden suorittamista varten. Scrumin käytettävyyden mahdollistamiseksi vaadittaisiin huomattavia muutoksia opintojen rakenteeseen, joten menetelmä ei ole sopiva opintojen aikana suoritettavien töiden suorittamiseen. Joidenkin ohjelmistotuotantoon pohjautuvien kurssien aikataulutusta voisi esimerkiksi muuttaa Scrumille sopivaksi, joka mahdollistaisi opiskelijoille suorittaa käytännön sovelluksia menetelmän toiminnalle.

Ketterien menetelmien kouluttamista varten voitaisiin luoda kurssi, jossa opiskelijat pääsisivät tutustumaan useampaan ketterään menetelmään niin teoriassa, kuin myös käytännössä. Tämän kaltaisen kurssin avulla opiskelijoille voitaisiin tarjota kattavampi koulutus ketterien menetelmien toimivuudesta ja niiden käytön merkityksestä kehitystyössä.

Kurssin rakennetta havainnollistetaan kuvassa 4.1. Tämän pohjalta kurssi ete-



Kuva 4.1: Kurssin toteutuksen rakenne ja vaiheiden pääpiirteet.

nisi ensin ketterän kehityksen historian ja yleisen teorian pohjalta opiskelijoiden omien esitelmien valmistamiseen ketterän kehityksen menetelmistä. Pienryhmissä opiskelijat valitsevat ryhmälleen yhden ketterän menetelmän, josta luodaan ja pidetään lyhyt esitelmä muille opiskelijoille. Esitelmässä tulisi esille valitun menetelmän pääpiirteet, työn etenemisen vaiheet ja käytön laajuus. Näin jokainen opiskelija tutustuisi tarkemmin oman ryhmän menetelmään, mutta myös pystyy vertailemaan menetelmiä muiden ryhmien esitysten pohjilta.

Käytännön sovelluksissa käytettäisiin tällä hetkellä työelämässä yleisimmin käytettyjä menetelmiä, kuten Scrumia ja Kanbania. Käytännön tutustuminen aloitetaan hyödyntämällä menetelmiä ryhmissä yksinkertaisiin peleihin. Opiskelijat keskustelevat ja analysoivat pelien välissä omista havainnoista käytetystä menetelmästä ja projektin etenemisestä, jotta praktiikat jäävät paremmin mieleen. Perusteiden oppimisen jälkeen opiskeluperiodissa muiden kurssien ryhmätyöt integroitaisiin tämän kurssin kanssa ja työt suoritettaisiin käyttämällä jotakin ketterän kehityksen menetelmää avuksi.

Kaikki ketterän kehityksen menetelmät pohjautuvat ketterän kehityksen mani-

festissa ilmentyviin periaatteisiin, joten sisäistämällä yhden menetelmän toiminta-periaatteen, on helppoa mukautua käyttämään myös muita menetelmiä. Ketterän kehityksen perusteiden kattava opettaminen kehittäisi opiskelijoiden valmiutta niin työelämään, sekä ohjaamaan opiskelun aikana tapahtuvien töiden tehokkaampaa suorittamista.

5 Yhteenveto

Ketterä kehitys on luotu tarpeesta, ratkaisemaan ohjelmistotuotannon kohtaamia haasteita ja positiivisten tuloksien myötä se on noussut hyvin olennaiseksi osaksi nykyajan ohjelmistotuotannon työkulkua. Ketterän kehityksen periaatteisiin perustuvia menetelmiä on monia, joista jokainen lähestyy kehitystyön suorittamista omalla tavallaan. Tässä tutkielmassa tutkittiin ketterän kehityksen menetelmistä Scrumia ja Kanbania. Tutkimuksessa pyrittiin tunnistamaan näiden menetelmien vahvuuksia, heikkouksia ja eroavaisuuksia.

Scrum ja Kanban kuuluvat käytetyimpiin ketterän kehityksen menetelmiin. Scrum on menetelmistä suositumpi ja sen toimintaa onkin laajalti tutkittu jo aikaisemmin. Kanban on menetelmistä nuorempi ja se tarjoaa erilaisen lähestymisen ketterään kehitykseen. Kumpikin menetelmistä on käytettävissä projekteissa yksinään ja niiden toimintaan kuuluu käytettyjen menetelmien sopeuttaminen erilaisille tiimeille ja projekteille kehityksen aikana. Tällä tavoin menetelmät tehostavat ja sopeuttavat toimintansa käyttötarkoitukseen sopivaksi.

Scrumin ja Kanbanin väliset erot liittyvät pääosin kehitystyön etenemiseen, tiimikokoihin, yhtäaikaisten kehityksessä olevien työtehtävien rajoittaminen, reagointi muutokseen, havainnollistavuus ja valmiiden työtehtävien julkaisu. Scrum on tutkimuksen perusteella suunnittelua ja järjestelyjä vaativa ketterän kehityksen menetelmä, jonka toiminta perustuu inkrementaaliseen, ennalta määrätyn kestoisiin ajanjaksoihin eli Sprintteihin. Vahvuudet ovat sopiva tiimikoko, toimiva yhteistyö

ja selkeä työnjako. Heikkouksia ovat kokemuksen puute menetelmän käytöstä, haasteellinen kommunikointi usean tiimin välillä ja hidas reagointi muutokseen. Kanban on tutkimuksen perusteella kevyt ja nopeasti käyttöönotettava kehitystyötä ohjaava ketterän kehityksen menetelmä. Kanbanin toiminta perustuu Kanban-tauluun, jonka sarakkeet edustavat kehityksen vaiheita ja työtehtävät ovat taulun sarakkeissa eteneviä kortteja. Kanbanin koetut vahvuudet ovat tilannekuvan havainnollistaminen, ongelmien tehokas ratkaisu ja jaksojen kestot. Heikkouksia ovat menetelmän ohjatun toiminnan puute, nykyisen kehityskulttuurin käyttämiä menetelmiä haastava muuttaa ja kehittäjien kokemuksen puute Kanban-menetelmästä.

Tutkimuksen perusteella Kanban toimii pienimmissä projekteissa, joissa osallistujia on 1-4, Scrumia paremmin. Syynä tähän on Scrumin toiminnan vaatimat roolien toteutumiset, jolloin osallistujia pitäisi olla enemmän. Keskisuurissa projekteissa kummatkin menetelmät ovat toimivia. Suuriin projekteihin menetelmiä soveltaessa kummatkin kohtaava haasteita suuren tiimin ja muiden osallisten tilannekuvan ylläpitämisessä.

Optimaalinen toimintaympäristö Scrumille on tutkimuksen perusteella keskikokoinen ohjelmistoprojekti, johon osallistuvat Scrum-tiimit ovat alle 10 henkisiä ja työskentelevät mieluiten samassa paikassa tai aikavyöhykkeessä. Menetelmänä Scrum toimii parhaiten projekteissa, jotka vaativat tiivistä yhteistyötä ja innovaatiota. Tehokkaan toiminnan takaamiseksi, Scrum-tiimin henkilöt tulisivat pysyä samoina ja sitoutuneena projektiin. Scrumin käyttöönotto vaatii perusteellista suunnittelua sekä roolien nimittämistä jäsenille.

Optimaalinen toimintaympäristö Kanbanille on tutkimuksen perusteella pienet ja keskikokoiset ohjelmistoprojektit. Kehitykseen osallistuvan tiimin kokoa ei ole selkeästi määrätty, mutta suuri tiimi heikentää sen toimintatehokkuutta kaikkien jäsenten välisen kommunikoinnin vaikeutuessa. Kanban on menetelmänä nopeas-

ti toteutettavissa sen yksinkertaisen toimintaperiaatteen ja vakiintuneiden roolien puutteen takia.

Ketterän kehityksen menetelmät ovat aiheena hyvin laaja ja tutkimusta voitaisiin lähestyä monista näkökulmista. Aihealueen rajaamisen seurauksena tässä tutkielmassa käsiteltiin vain kahta ketterän kehityksen menetelmää. Tämän tutkielman aiheesta voisi suorittaa jatkotutkimusta esimerkiksi opiskelijoille suunnatulla kyselyllä. Kyselyn avulla kartoitetaan opiskelijoiden tämänhetkistä tieto- ja taitotasoa ketteristä menetelmistä ja niiden toimeenpanemisesta kehitystyössä. Jatkotutkimusta voisi myös suorittaa järjestämällä edellisessä luvussa esitelty kurssi ja tutkia opiskelijoiden tietoja ja taitoja kurssia ennen ja sen jälkeen.

Lähdeluettelo

[1] T. Dybå ja T. Dingsøyr, "Empirical studies of agile software development: A systematic review", *Information and Software Technology*, vol. 50, nro 9, s. 833–859, 2008, ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2008.01.006. url: https://www.sciencedirect.com/science/article/pii/S0950584908000256.

[2] P. Abrahamsson, O. Salo, J. Ronkainen ja J. Warsta, *Agile software development methods: Review and analysis* (VTT Publications 478), English. Finland: VTT Technical Research Centre of Finland, 2002, ISBN: 951-38-6009-4.

[3] M. Ikonen, E. Pirinen, F. Fagerholm, P. Kettunen ja P. Abrahamsson, "On the Impact of Kanban on Software Project Work: An Empirical Case Study Investigation", teoksessa *2011 16th IEEE International Conference on Engineering of Complex Computer Systems*, 2011, s. 305–314. DOI: 10.1109/ICECCS.2011.37.

[4] *Agile Manifesto*, https://agilemanifesto.org/. (viitattu 29. 02. 2024).

[5] T. Chow ja D.-B. Cao, "A survey study of critical success factors in agile software projects", *Journal of Systems and Software*, vol. 81, nro 6, s. 961–971, 2008, Agile Product Line Engineering, ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2007.08.020. url: https://www.sciencedirect.com/science/article/pii/S0164121207002208.

[6] J. Sutherland ja K. Schwaber, *Scrum Guide*, https://scrumguides.org/, 2020.

[7] K. Schwaber, *Agile project management with Scrum*. Microsoft press, 2004.

[8] Wikimedia Commons, *Scrum project management method*, [Viitattu; marraskuu 12, 2023], 2009. url: https://en.m.wikipedia.org/wiki/File:Scrum_process.svg.

[9] A. Srivastava, S. Bhardwaj ja S. Saraswat, "SCRUM model for agile methodology", teoksessa *2017 International Conference on Computing, Communication and Automation (ICCCA)*, 2017, s. 864–869. DOI: 10.1109/CCAA.2017.8229928.

[10] D. Vacanti ja J. Coleman, *Kanban Guide*, https://kanbanguides.org//, 2020.

[11] M. O. Ahmad, J. Markkula ja M. Oivo, "Kanban in software development: A systematic literature review", teoksessa *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, 2013, s. 9–16. DOI: 10.1109/SEAA.2013.28.

[12] H. Alaidaros, M. Omar ja R. Romli, "The State of the Art of Agile Kanban Method: Challenges and Opportunities", *Independent Journal of Management & Production*, vol. 12, joulukuu 2021. DOI: 10.14807/ijmp.v12i8.1482.

[13] M. Alqudah ja R. Razali, "An empirical study of Scrumban formation based on the selection of scrum and Kanban practices", *Int. J. Adv. Sci. Eng. Inf. Technol*, vol. 8, nro 6, s. 2315–2322, 2018.

[14] H. Lei, F. Ganjeizadeh, P. K. Jayachandran ja P. Ozcan, "A statistical analysis of the effects of Scrum and Kanban on software development projects", *Robotics and Computer-Integrated Manufacturing*, vol. 43, s. 59–67, 2017, Special Issue: Extended Papers Selected from FAIM 2014, ISSN: 0736-5845. DOI: https://

doi.org/10.1016/j.rcim.2015.12.001. url: https://www.sciencedirect.com/science/article/pii/S0736584515301599.

[15] R. Polk, "Agile and Kanban in Coordination", teoksessa *2011 Agile Conference*, 2011, s. 263–268. DOI: 10.1109/AGILE.2011.10.

[16] M. Hamdani ja W. H. Butt, "Success and Failure Factors in Agile Development", teoksessa *PROCEEDINGS 2017 INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE AND COMPUTATIONAL INTELLIGENCE (CSCI)*, H. Arabnia, L. Deligiannidis, F. Tinetti, Q. Tran ja M. Yang, toim., International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, DEC 14-16, 2017, American Council Sci & Educ, 2017, s. 981–986, ISBN: 978-1-5386-2652-8. DOI: 10.1109/CSCI.2017.169.

[17] M. Paasivaara, C. Lassenius ja V. T. Heikkila, "Inter-team Coordination in Large-Scale Globally Distributed Scrum: Do Scrum-of-Scrums Really Work?", teoksessa *PROCEEDINGS OF THE ACM-IEEE INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT (ESEM'12)*, sarja International Symposium on Empirical Software Engineering and Measurement, 6th ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Lund, SWEDEN, SEP 19-20, 2012, IEEE; Assoc Comp Machinery (ACM); ACM Special Interest Grp Software Engn (SIGSOFT); IEEE Comp Soc (CS), 2012, s. 235–238, ISBN: 978-1-4503-1056-7.

[18] J. O. Birkeland, "From a Timebox Tangle to a More Flexible Flow", teoksessa *AGILE PROCESSES IN SOFTWARE ENGINEERING AND EXTREME PROGRAMMING*, A. Sillitti, A. Martin, X. Wang ja E. Whitworth, toim., sarja Lecture Notes in Business Information Processing, 11th International

Conference on Agile Software Development (XP2010), Trondhim, NORWAY, JUN 01-04, 2010, vol. 48, 2010, s. 325–334, ISBN: 978-3-642-13053-3.

[19] R. Budiman, T. Raharjo ja A. Suhanto, ”Scrum Project Management Challenges and Solutions: Systematic Literature Review”, teoksessa *2022 IEEE 8th International Conference on Computing, Engineering and Design (ICCED)*, 2022, s. 1–6. DOI: 10.1109/ICCED56140.2022.10010471.

[20] N. Tripathi, P. Rodríguez, M. O. Ahmad ja M. Oivo, ”Scaling Kanban for Software Development in a Multisite Organization: Challenges and Potential Solutions”, teoksessa *Agile Processes in Software Engineering and Extreme Programming*, C. Lassenius, T. Dingsøyr ja M. Paasivaara, toim., Cham: Springer International Publishing, 2015, s. 178–190, ISBN: 978-3-319-18612-2. DOI: 10.1007/978-3-319-18612-2_15.

[21] B. L. Romano ja A. Delgado Da Silva, ”Project Management Using the Scrum Agile Method: A Case Study within a Small Enterprise”, teoksessa *2015 12th International Conference on Information Technology - New Generations*, 2015, s. 774–776. DOI: 10.1109/ITNG.2015.139.