Keskeiset epäonnistumistekijät ketterän kehityksen ohjelmistoprojekteissa

Turun yliopisto
Tietotekniikan laitos
TkK-tutkielma
Tietotekniikka
Toukokuu 2025
Anna Parkkinen

Turun yliopiston laatujärjestelmän mukaisesti tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -järjestelmällä.

TURUN YLIOPISTO
Tietotekniikan laitos

Anna Parkkinen: Keskeiset epäonnistumistekijät ketterän kehityksen ohjelmistoprojekteissa

TkK-tutkielma, 24 s.
Tietotekniikka
Toukokuu 2025

Tässä kandidaatintutkielmassa tarkastellaan ohjelmistoprojektien epäonnistumistekijöitä sekä keinoja näiden minimoimiseksi ketterän kehityksen viitekehyksessä. Tutkimuksen tavoitteena on tunnistaa keskeiset haasteet ja esitellä ratkaisuja, joilla ohjelmistoprojektien onnistumisprosenttia voitaisiin parantaa. Kirjallisuuskatsaus osoittaa, että epäonnistumiseen vaikuttavat monet eri tekijät. Nämä epäonnistumistekijät voidaan jakaa neljään eri kategoriaan. Katerogiat ovat organisatoriset haasteet, ihmisiin liittyvät haasteet, prosessiongelmat ja tekniset haasteet. Näitä ongelmia ei voida tarkastella irrallisina, vaan ne kytkeytyvät toisiinsa ja voivat aiheuttaa ketjureaktioita, jotka heikentävät projektin etenemistä ja lopputulosta.
Onnistumista tukevat selkeät roolit, tuki sprinttityön yläpuolella, avoin ja jatkuva kommunikaatio sekä panostaminen projektijohtamisen ja -osaamisen koulutukseen. Johto nousee keskeiseen rooliin paitsi suunnan näyttäjänä myös mahdollistajana, jonka tehtävänä on tukea tiimiä estämättä sen itseohjautuvuutta, joka on tärkeää ketterässä kehityksessä. Avainasemassa on jatkuva oppiminen, reflektointi ja kehittämismyönteinen ilmapiiri.

Asiasanat: Ohjelmistoprojekti, ketterä kehitys, ketterät menetelmät, onnistumistekijät, epäonnistumistekijät, projektinhallinta, johtaminen

Sisällys

Kuvat

Taulukot

1 Johdanto

Ohjelmistoprojektien epäonnistumisprosentti on globaalisti erittäin korkea. Vuonna 2023 ohjelmistoprojektien epäonnistumisprosentiksi todettiin noin 70 prosenttia [1]. Projektinhallintaan ja siihen valittavaan menetelmään vaikuttaa nykymaailmassa nopeasti muuttavat markkinat ja jatkuvasti kehittyvät uudet teknologiat [2]. Suurin ero, mitä ketterät menettelyt tuovat ohjelmistoprojekteihin vesiputousmallin tilalle on joustavuus. Tämä näkyy projektin vaiheissa kommunikaationa, jolla ennaltaehkäistään ja minimoidaan riskejä. [3] Tästä huolimatta epäonnistumisprosentti on yhä lähellä seitsemääkymmentä. Tässä tutkielmassa keskitytään ketteriä menettelyjä noudattaviin ohjelmistoprojekteihin, koska ketteryys vastaa jo moneen vesiputousmallin ongelmaan. Vesiputousmalli on ketterää kehitystä edeltävä projektinhallintamenetelmä, mikä on perinteinen lienaarinen lähestymistapa ohjelmistokehitykseen. Ketterät menetelmät onkin nykyään suosittuja projektinhallinnassa.

Liiketoiminnan näkökulmasta IT-projekteihin sidotaan paljon pääomaa. Projektin epäonnistuminen johtaa siihen käytetyn pääoman menettämiseen. Ohjelmistoprojektit yksinään kustantavat 50-80 miljardia dollaria vuosittain [4]. Jo pelkästään vuonna 2017 Suomen valtio satsasi 90 miljoonaa euroa sote-uudistuksen IT-kehitykseen [5]. Näiden valtavien summien lisäksi IT-projektit ylittävät budjettinsa keskimäärin noin neljänneksellä. Tilanne on kuitenkin vielä huolestuttavampi, koska noin joka kuudes projekti ylittää suunnitellun budjetin jopa yli kolminkertaisesti. Samalla näiden projektien aikataulut venyvät lähes kaksinkertaisiksi alkuperäiseen

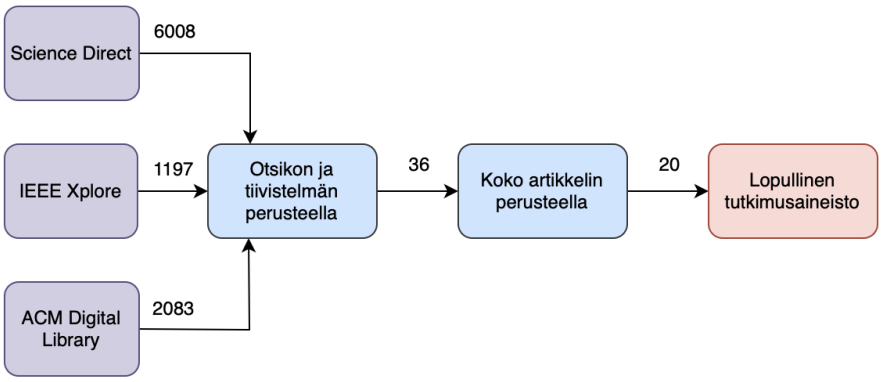
suunnitelmaan verrattuna. [1] Onnistuneista projekteista saatu hyöty taloudellisesti on kuitenkin näitä rahasummia suurempi. [4]

Jotta voidaan ymmärtää, miksi projektit epäonnistuvat, on tärkeää tehdä jälkikäteisarviointeja. Ilman niitä ei todennäköisesti tunnista epäonnistumisten syitä. [6] Virheiden analysointi johtaa riskienhallintaan ja näin kehittymiseen. Ongelmien ymmärrys johtaa projektien resurssien tehokkaampaan käyttöön, asiakkaiden tyytyväisyyteen sekä IT-alan ja tiimien kehittymiseen. Analysoimalla epäonnistuneita projekteja voidaan kehittää ohjelmistokehitysprosessia ja tehdä parannuksia. Tämä auttaa myös tunnistamaan kriittiset päätöksentekokohdat jo ennen projektin aloitusta sekä sen edetessä. [6] Tutkielman tutkimuskysymykset ovat:

- TK1: Mitkä ovat ketteriä käytäntöjä noudattavien ohjelmistoprojektien tyypillisimpiä epäonnistumistekijöitä?
- TK2: Miten näitä tunnistettuja epäonnistumistekijöitä on mahdollista hallita?

Tutkielma toteutettiin kirjallisuuskatsauksena. Tietokannat, joissa haku toteutettiin oli ScienceDirect, ACM Digital Library ja IEEE Xplore. Nämä tietokannat valikoituivat, koska ne tuottivat osuvimmat tulokset hakulauseelle ja sanoille. Haussa käytetty hakulause oli (Agile software project challenges) OR (Software project fail* OR 'Waterfall vs agile' OR 'Save software project'). Haku tehtiin englanniksi ja suodatettiin Kuvan 1.1 mukaisesti.

Ensimmäisessä haussa löydettiin yhteensä 9288 artikkelia, joiden otsikot ja tiivistelmät käytiin läpi. Tässä vaiheessa karsittiin pois kaikki artikkelit, jotka eivät liittyneet tutkimuskysymyksiin. Artikkeleiden rajauksessa karsittiin pois myös ne, jotka eivät koskeneet ohjelmistoprojekteja ja niissä kohdattuja ongelmia. Aineistoon rajattiin mukaan artikkelit, jotka on julkaistu 2010-luvulla ja sen jälkeen. Kriteerinä oli myös, että ne keskittyvät ketteriin menetelmiin.



Kuva 1.1: Aineistonhakuprosessi

Ensimmäisen karsinnan jälkeen jäljelle jäänet 28 artikkelia luettiin kokonaisuudessaan. Sieltä karsiintui tutkielman käyttöön 16 artikkelia, jotka tukivat tutkimuksen suuntaa parhaiten. Tutkimusta täydennettiin sivustoilla, jotta tutkimus on ajankohtainen. Esimerkiksi ohjelmistoprojektien epäonnistumisprosentti haluttiin ajankohtaisesti tutkielmaan. Mukaan otettiin myös yksi artikkeli, joka on julkaistu ennen 2010-lukua.

Tutkielma on jaettu johdannon lisäksi neljään lukuun. Luvussa kaksi käsitellään ketterän ohjelmistokehityksen projektinhallinan menetelmiä. Luku kolme esittelee kirjallisuuskatsauksena analysoidut artikkelit, jotka käsittelevät epäonnistuneita ohjelmistoprojekteja. Luvussa neljä näitä epäonnistumistekijöitä käsitellään niitä ehkäisevän toiminnan näkökulmasta. Viides luku vielä erittelee projektin rooleja sekä kriitiisiä kohtia, missä pitäisi parantaa. Viimeisessä luvussa, yhteenvedossa käydään läpi tutkielman näkökulmia, ja selvitettyjä haasteita sekä niiden mahdollisia ratkaisuja.

2 Ketterät menetelmät projektinhallinnassa

Ketterät menetelmät ovat alun perin kehitetty ohjelmistoprojekteihin, mutta niitä sovelletaan nykyään laajasti myös muilla aloilla [7]. Niiden suosio perustuu havaintoihin ketteryyden ja projektin onnistumisen välisestä yhteydestä [8]. Ketterissä menetelmissä korostuu ajatus, että vähemmän suunnittelua voi olla tehokkaampaa. Ne eroavat perinteisistä malleista, kuten vesiputousmallista, painottamalla jatkuvaa suunnittelua, joustavaa laajuutta ja aktiivista asiakasvuorovaikutusta. Kehitys tapahtuu iteratiivisesti ja inkrementaalisesti: ohjelmistoa kehitetään jaksoissa lisäämällä ominaisuuksia yksi kerrallaan. Syklit kestävät tyypillisesti alle kuukauden. [7], [8]

Ketterät menetelmät soveltuvat erityisesti nopeasti muuttuvissa ympäristöissä [9]. Niissä korostuvat joustavuus ja tiivis yhteistyö tuoteomistajan kanssa, jotta kehitys vastaa käyttäjien tarpeita. Tärkeä periaate on kyky reagoida muutoksiin sen sijaan, että pitäydytään alkuperäisessä suunnitelmassa. [10] Menetelmissä arvostetaan vuorovaikutusta enemmän kuin prosesseja tai työkaluja. Muutosta pidetään mahdollisuutena, ja kehitystyötä tehdään yhteistyöhön kannustavassa ilmapiirissä. [10]

Keskeisiä viitekehyksiä ovat Scrum, Kanban ja CI/CD (jatkuva integraatio ja toimitus). Scrum käyttää lyhyitä sprinttejä, joissa tiimi pyrkii ennalta asetettuihin

tavoitteisiin, korostaen tarkastelua ja itseohjautuvuutta. Kanban perustuu visuaaliseen työnkulkuun ja rajoittaa keskeneräisiä tehtäviä, mikä auttaa hahmottamaan etenemistä ja kehittämään prosesseja. CI/CD puolestaan mahdollistaa nopeat ja toistuvat ohjelmistojulkaisut. [9], [10] Päivittäiset palaverit ovat olennainen osa ketteryyttä, koska niissä tiimi käy läpi edistymistä, päivän tehtäviä ja haasteita [10].

2.1 Ketteryyden hyödyt

Ketteryys nähdään toimivana ohjelmistoprojektinhallinnan työkaluna projektin laadun, laajuuden, ajan ja kustannusten näkökulmasta. Chow ja Cao [11] toteuttivat tutkimuksen, jossa on mukana dataa 109:stä eri ketterästä ohjelmistoprojektista, ja data on kerätty 25:stä eri maasta. Tutkimuksessa heidän löytämänsä kolme kriittistä onnistumistekijää ovat toimitusstrategia, ketterät ohjelmistosuunnittelutekniikat ja tiimin kyvykkyys. Artikkelissa painotetaan projektijohtajan tekemien valintojen tärkeyttä. Projektijohtajan täytyy keskittyä näihin kolmeen elementtiin, jotta taataan onnistunut projekti ja prosessi. [11]

Näiden elementien lisäksi asiakkaan osallistuminen ja psykologinen turvallisuus vaikuttavat merkittävästi projektiin. Psykologinen turvallisuus ilmenee tiimin dynamiikassa, jolloin jäsenet tuntevat olonsa turvalliseksi. Tämä johtaa avoimeen ja rohkeaan ideointiin sekä kommunikaatioon. On todettu, että 46 prosenttia projektin onnistumistekijöistä liittyy tiimin kyvykkyyksiin ja asiakkaan rooliin [8]. Ketterissä menetelmissä viestintä on keskeistä, ja dokumentaatio pyritään pitämään kevyenä ja helposti ymmärrettävänä. Ketteryys pyrkii tehostamaan kehitysprosessia vähentämällä turhia ja epäsäännöllisiä tehottomuuksia sekä nopeuttamalla toimituksia. [8]

Ketteryys muistuttaa Lean-ajattelua, jossa pyritään poistamaan hukkaa ja keskittymään arvoa tuottaviin toimintoihin. Tämän saavuttaminen vaatii asiakaslähtöisyyttä ja sitoutumista jatkuvaan yhteistyöhön. [8] Puhdas ketteryys edellyttää, että

asiakas on valmis tiiviiseen ja toistuvaan yhteistyöhön, hyväksyy projektin lopputulokseen liittyvän epävarmuuden ja panostaa koulutukseen sekä alkuinvestointeihin.

Tutkimukset osoittavat, että ketterät menetelmät voivat parantaa erityisesti asiakastyytyväisyyttä, mutta puhtaan ketteryyden saavuttaminen on työlästä ja haastavaa. Menetelmien mukauttaminen ja järkevä soveltaminen ovat tärkeitä. On olennaista kehittää aidosti ketterää toimintaa sen sijaan, että ketteryys nähdään itseisarvona tehokkuuden kustannuksella. [8]

2.2 Ketterien menetelmien haasteet ja toimivuuden mittaaminen

Ketterän projektinhallinnan haasteena on, että lyhyen aikajänteen suunnittelu voi johtaa epäselviin toimenpiteisiin ja heikompaan sitoutumiseen. Muutospyyntöjen joustavuus voi ohittaa alkuperäisen suunnitelman johdonmukaisuuden, mikä aiheuttaa hallitsemattomia suunnan- ja laajuuden muutoksia. Iteratiivinen ja testivetoinen lähestymistapa voi vaikeuttaa projektin kokonaisvaltaista hallintaa ja ennustettavuutta, erityisesti verrattuna lineaariseen vesiputousmalliin. Vaikka ketterät menetelmät reagoivat nopeasti muuttuviin asiakasvaatimuksiin, ne voivat lisätä epävarmuutta ja hallinnan haasteita. [2]

Ketteryyden vahvuutena sekä heikkoutena on sen iteratiivinen työskentelytapa, mikä voi olla ristiriidassa perinteisten yrityksien kulttuurin kanssa. Tämä voi ilmetä erityisesti suunnittelussa, raportoinnissa, hierarkkisissa rakenteissa ja johtamisessa, mikä voi vaikeuttaa sen käyttöönottoa. Lisäksi projektin onnistuminen riippuu vahvasti tiimin taidoista ja itseohjautuvuudesta. [2]

Projektin koko vaikuttaa ketterien menetelmien hallintaan. Suuremmissa projekteissa korostuvat viestinnän, joustavuuden ja koordinoinnin haasteet. Ketteryyden skaalaaminen edellyttää usein organisaatiorakenteiden muutoksia ja huomioimista

moniin toimialakohtaisiin sääntelyihin. [12]. Mahdollisesti aikaavievä käyttöönotto-prosessi näkyy projektin viivästymisenä.

Ketteriä menetelmiä esitellessä etenkin suuremmille organisaatioille, hybridilähestymistapa on hyvä aloitus menetelmien omaksumiseen ja toimintatapojen integrointiin. Hybridilähestymistapa yhdistää perinteisen ja ketterän mallin vahvuudet. Tällöin laaja kokonaiskuva rakennetaan perinteisellä tavalla, mutta osaprojektit hallinnoidaan ketterästi hyödyntäen tiheää viestintää ja lyhyitä palautesyklejä. Menetelmän valinta on usein haasteellinen käytännön toimijoille. [2]

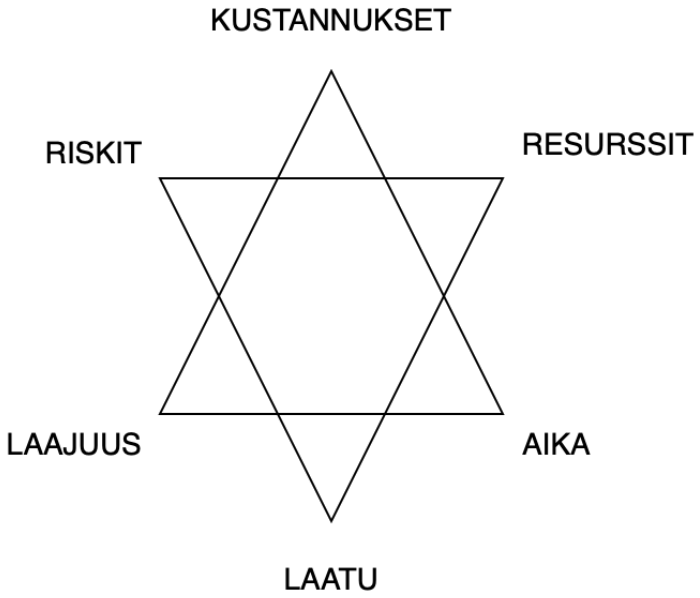
Vaikka ohjelmistot ovat keskeisessä roolissa nykypäivänä, niiden kehitysprosessit eivät ole täydellisiä. Projekteja viivästyy, epäonnistuu tai hylätään, ja onnistuneessakin toteutuksessa tarvitaan jatkuvaa ylläpitoa ja virheiden korjauksia. [11]

Ketterät menetelmät voivat tarjota etuja perinteisiin menetelmiin nähden, mutta eivät takaa automaattista menestystä. Projektien epäonnistuminen on edelleen yleistä, ja epäonnistumiset voivat tulla kalliiksi. Siksi projektipäälliköiden on tärkeää ymmärtää, mitkä tekijät vaikuttavat onnistumiseen. [8]

Perinteisesti projektien onnistumista mitataan kustannusten, aikataulun, laadun ja sidosryhmien odotusten perusteella [13]. Tämä auttaa seuraamaan projektin etenemistä ja tekemään tarvittavat muutokset ajoissa. Barros et al. [8] esittelevät rautakolmion (Kuva 2.1) mittaristona, joka koostuu kustannuksista, ajasta, laajuudesta ja laadusta. Kuitenkin esimerkiksi sidosryhmien tyytyväisyys, mitä ei mitatata rautakolmiossa, on noussut tärkeäksi onnistumisen mittariksi erityisesti ketterissä projekteissa.

Kuva 2.1: Rautakolmio

Nykyään suosittu mittari on projektinhallinnan tähti (Kuva 2.2), joka laajentaa rautakolmiota lisäämällä resurssit ja riskit. Tämä korostaa projektien moniulotteisuutta ja sitä, miten eri rajoitteiden yhteensovittaminen voi olla haastavaa. [14]



Kuva 2.2: Projektinhallinnan tähti

3 Yleisimmät ohjelmistoprojektien epäonnistumistekijät

Tutkimukset ohjelmistoprojektien epäonnistumisista ja ongelmista perustuvat usein aiemmista projekteista saatuihin oppeihin. Vaikka projektit voivat erota toisistaan, niissä ilmenevät haasteet ovat usein riittävän samankaltaisia, jotta niistä voidaan tehdä yleistyksiä. [11] Ohjelmistoprojektien peruuntumiset johtuvat usein projektin aikana tehdyistä virheistä, ja tällaisilla peruuntumisilla on suuri taloudellinen vaikutus [6].

Erityisesti ohjelmistokehitysprojekteja tarkasteltaessa on tunnistettu kymmenen epäonnistumisen merkkiä. Näistä vähintään seitsemän voidaan havaita jo ennen kuin projektin suunnittelua on edes aloitettu tai ensimmäistäkään koodiriviä kirjoitettu. Nämä ovat: 1) projektipäälliköt eivät ymmärrä käyttäjien tarpeita, 2) projektin laajuutta ei ole määritelty, 3) hankkeen muutoksia hallitaan huonosti, 4) valittu teknologia muuttuu, 5) liiketoiminnan tarpeet muuttuvat, 6) määräajat ovat epärealistisia, 7) käyttäjät ovat vastahakoisia, 8) sponsorointi on menetetty, 9) tiimistä puuttuu kyvykkyyksiä ja 10) johdon kyvykkyys. [15]

Monet ohjelmistoprojektit epäonnistuvat jo ennen varsinaisen työn aloittamista, sillä kriittiset virheet tehdään usein projektin suunnittelu- ja tarjousvaiheessa. Yleisiä ongelmia ovat esimerkiksi epärealistiset aikataulut, puutteellinen ymmärrys vaatimuksista, väärät henkilöstövalinnat ja lupaukset toteuttaa teknisesti liian vaa-

tivia ratkaisuja ilman riittävää osaamista. Usein projektin kohtalo sinetöidään sopimusvaiheessa, kun toimittaja ja asiakas eivät tunnista riskejä tai tekevät päätöksiä liiketoiminnallisista syistä teknisten realiteettien sijaan. Esimerkiksi liian tiukka aikataulu tai osaamattoman tiimin valinta voi johtaa peruuttamattomiin ongelmiin, jotka käyvät ilmi vasta projektin edetessä. Vaikka projektitiimi ja johto tekisivät kaikkensa, kriittinen alkuvaiheen virhe voi tehdä projektin pelastamisesta mahdotonta. [6]

3.1 Haasteiden kategorisointi kokonaisuuksiksi

Useat tutkimukset ovat käsitelleet organisaatioiden siirtymistä ketteriin menetelmiin ja niihin liittyviä haasteita. Ne ovat tuoneet esiin tyypillisiä virheitä ja väärinkäsityksiä ketterien projektien toteutuksessa. Ketterien hankkeiden johtaminen on erityinen haaste, mutta ongelmia ilmenee myös laajemmin, esimerkiksi ihmisiin, prosesseihin ja teknologiaan liittyvillä alueilla. [11]

Chow ja Cao [11] ovat luokitelleet nämä epäonnistumiset ja ongelmat neljään pääkategoriaan, jotka ovat 1) organisatoriset haasteet, 2) ihmisiin liittyvät haasteet, 3) prosessiongelmat ja 4) tekniset haasteet. Organisatoriset haasteet liittyvät yrityksen rakenteeseen ja toimintatapoihin ja taas ihmisiin liittyvät esimerkiksi tiimin dynamiikkaan ja osaamiseen. Prosessiongelmat näkyvät esimerkiksi puutteellisena suunnitteluna tai epäselvinä toimintamalleina, kun taas tekniset haasteet näkyvät teknologisina rajoituksina ja haasteina. Näiden neljän osa-alueen ymmärtäminen auttaa tunnistamaan ja ehkäisemään mahdollisia ongelmia jo varhaisessa vaiheessa. [11]

Taulukko 3.1: Epäonnistumistekijät

Ongelma	Holtsnider, 2010	Ahonen, 2010	Lehtinen, 2014	Mohagheghi, 2017	Wu, 2024	Conboy, 2019	Damasiotis, 2017
Organisatoriset haasteet (1)							
Johdon osallistumisen/-taidon puute			X	X	X	X	X
Organisaation/yrityksen menettelyt			X	X		X	X
Ongelma rahoituksessa	X		X				X
Ihmisiin liittyvät haasteet (2)							
Tiimin kyvykkyyksien puute		X	X			X	
Huono suhde/kommunikaatio sidosryhmien/asiakkaan kanssa		X	X	X		X	
Ongelma tiimin/projektin koossa			X				
Tiimin kommunikaation puute			X	X		X	
Prosessiin liittyvät haasteet (3)							
Huono prosessin seuranta/raportointi			X				
Epäonnistunut suunnittelu ja rajaus	X	X	X	X	X	X	X
Tekniset haasteet (4)							
Väärät teknologiat/työkalut		X	X	X		X	X

Edellä esitetty Taulukko 3.1 havainnollistaa ohjelmistoprojektien yleisimpiä epäonnistumistekijöitä. Nämä ongelmat on luokiteltu edellä mainittuihin neljään pääkategoriaan. Taulukko 3.1 listaa seitsemän eri artikkelin käsittelemiä ongelmia. Taulukko mahdollistaa ongelmien yleisyyden hahmoittamisen. Se näyttää, miten eri vuosina julkaistujen ja eri näkökulista käsittelevien artikkelien ohjelmistoprojektien ongelmat ovat samankaltaisia. Lähteiden perusteella Taulukko 3.1 valittiin yleisimpiä ongelmia. Näiden lisäksi on iso määrä muitakin haasteita.

Yleisimmiksi tekijöiksi valikoitui ketteriä menettelyjä noudattavien ohjelmistoprojektien toteuttamisessa haasteet kyvykkyyksissä, kommunikaatiossa, teknologioissa ja yleisesti suunnittelussa. Taulukko 3.1 havainnollistaa, miten jokaisessa artikkelissa projektin suunnittelu oli epäonnistumistekijänä jollain tavalla. Huono suunnittelu näkyy esimerkiksi viivästyksinä ja budjettiongelmina. Puutteellinen riskienhallinta voi aiheuttaa ennakoimattomia haasteita. Lisäksi epärealistiset vaati-

mukset ja huono priorisointi voivat kuormittaa tiimiä ja heikentää lopputuloksen laatua. Ilman selkeää suunnitelmaa projekti voi ajautua hallitsemattomaksi ja epäonnistua. [4], [12], [16]–[19]

Toiseksi eniten esiintyvä ongelma oli johdon osallistumisen tai taidon puute. Keskeisiä ongelmia ovat epäselvä visio ja strategia, huono päätöksenteko sekä heikko kommunikaatio. [16], [17] Nämä näkyvät Taulukossa 3.1. Tällaiset tekijät voivat hidastaa projektia ja aiheuttaa epävarmuutta. Lisäksi resurssien ja budjetin hallinnan puutteet voivat johtaa resurssipulaan ja kustannusten hallitsemattomaan kasvuun. [18], [19]

Jos johto ei tue tiimiä sekä motivoi sitä, työntekijöiden suorituskyky voi heikentyä. Muutosvastarinta voi kasvaa ilman johdon tukea, etenkin suuremmissa perinteisissä organisaatioissa [12]. Hyvä johtajuus on ratkaisevan tärkeää projektin selkeän suunnan, riittävien resurssien ja tehokkaan yhteistyön varmistamiseksi. [12], [16]

Kommunikaation tärkeys niin tiimin kuin sidosryhmien kanssa nousi esille lähes jokaisessa tarkastellussa artikkelissa. [6], [12], [16], [17] Etenkin ketteriä menetelmiä noudattavissa ohjelmistoprojekteissa lyhyiden syklien tarvitsema kommunikaatio on avainasemassa.

Sidosryhmien merkitys on yhä tärkeämpää. Tiivis kommunikaatio ja yhteistyö heidän kanssaan johtaa usein projektin menestykseen. [12] Suunnitteluun ja projektiin valmistautumiseen kuuluu myös tiimin koon, kyvykkyyksien ja heidän käyttämien teknologioiden sekä työkalujen valitseminen, joita saatetaan pitää itsestäänselvyyksinä. Kuitenkin nämä seikat nousivat usein esiin projektien epäonnistumistekijöinä. [6], [16], [19]

3.2 Epäonnistumistekijöiden analysoinnin tavoitteet

Kuten todettu, monet hankkeet peruuntuvat ulkoisten tekijöiden, kuten liiketoimintaympäristön muutosten takia. Kuitenkin useat niistä epäonnistuvat sisäisten virheiden takia jo ennen toteutusta tai sen aikana. Tällaiset tapaukset ovat erityisen kiinnostavia, sillä niiden analysointi auttaa tunnistamaan yleisiä virheitä ja ehkäisemään niitä tulevaisuudessa. Ymmärtämällä peruuntumisen taustat on mahdollista parantaa projektinhallintaa ja vähentää turhaa resurssien haaskausta. [6] Analyysi yleisimmistä epäonnistumistekijöistä johtaa useampaan onnistuneeseen projektiin.

Zianan ja Charlesin [20] tutkimuksen päätavoitteena on asettaa etusijalle aiemmin tunnistetut riskitekijät ja arvioida niiden merkitystä ohjelmistohankkeissa. Tämä tutkimus, ja sen päämäärä tukee hyvin tämän tutkielman tavoitteita. Zianan ja Charlesin [20] kokeelliseen tutkimukseen osallistui kolme eri ryhmää ohjelmistoharjoittelijoita, jotka tunnistivat yhteensä 435 riskiä hanke-erittelyiden pohjalta. Riskitekijät jaettiin eri luokkiin, kuten 'ylilyönnit', 'alilyönnit', 'virheet', 'käsitteelliset ongelmat', 'muutokset', 'erot', 'vaikeudet', 'riippuvuudet', 'ristiriidat', 'ongelmat' ja 'haasteet'. Kokeneet asiantuntijat arvioivat näitä riskitekijöitä pisteytyksen ja kriittisyyden perusteella. [20]

Tutkimuksessa analysoitiin erityisesti 16 satunnaisesti valittua riskitekijää, jotka liittyivät 'vaikeuksiin', 'muutoksiin', 'virheisiin' ja 'haasteisiin'. Näiden riskitekijöiden ymmärtäminen ja hallinta on olennaista ohjelmistohankkeiden onnistumisen kannalta. Esimerkiksi suurten hankkeiden epäonnistumiseen vaikuttavat usein 'vaikeudet', kun taas 'muutokset' voivat joko edistää tai haitata hankkeen etenemistä. 'Virheitä' syntyy, kun oikeat käsitteet toteutetaan väärin, ja 'haasteet' tuovat mukanaan uusia olosuhteita, jotka voivat aiheuttaa ristiriitoja tai toisaalta vauhdittaa kehitystä.

Ziana ja Charles [20] tukivat väitettä, että riskitekijöiden analysointi ja priorisointi ovat välttämättömiä, jotta voidaan toteuttaa tehokkaita lieventämistoimia. Ilman huolellista arviointia ja ennakoivaa riskienhallintaa ohjelmistohankkeet voivat kohdata vakavia ongelmia, jotka vaarantavat niiden onnistumisen. Siksi riskien systemaattinen tutkiminen on keskeinen osa projektinhallintaa ja auttaa parantamaan hankkeiden laatua ja luotettavuutta. [20]

4 Ohjelmistoprojektien onnistumiseen vaikuttavat tekijät

Virheet voidaan välttää ymmärtämällä, mikä meni pieleen ja miten se olisi voitu välttää. Näihin kysymyksiin ei kuitenkaan ole yleisesti saatavilla hyviä vastauksia peruuntuneiden ohjelmistohankkeiden osalta. Saatavuuden puutteeseen on ainakin kaksi syytä: se, että vain pieni osa hankkeista käy läpi epäonnistumisanalyysin, ja se, että peruuntuneita ohjelmistoprojekteja koskevaa tietoa ei ole jaettu julkisuuteen. [6]

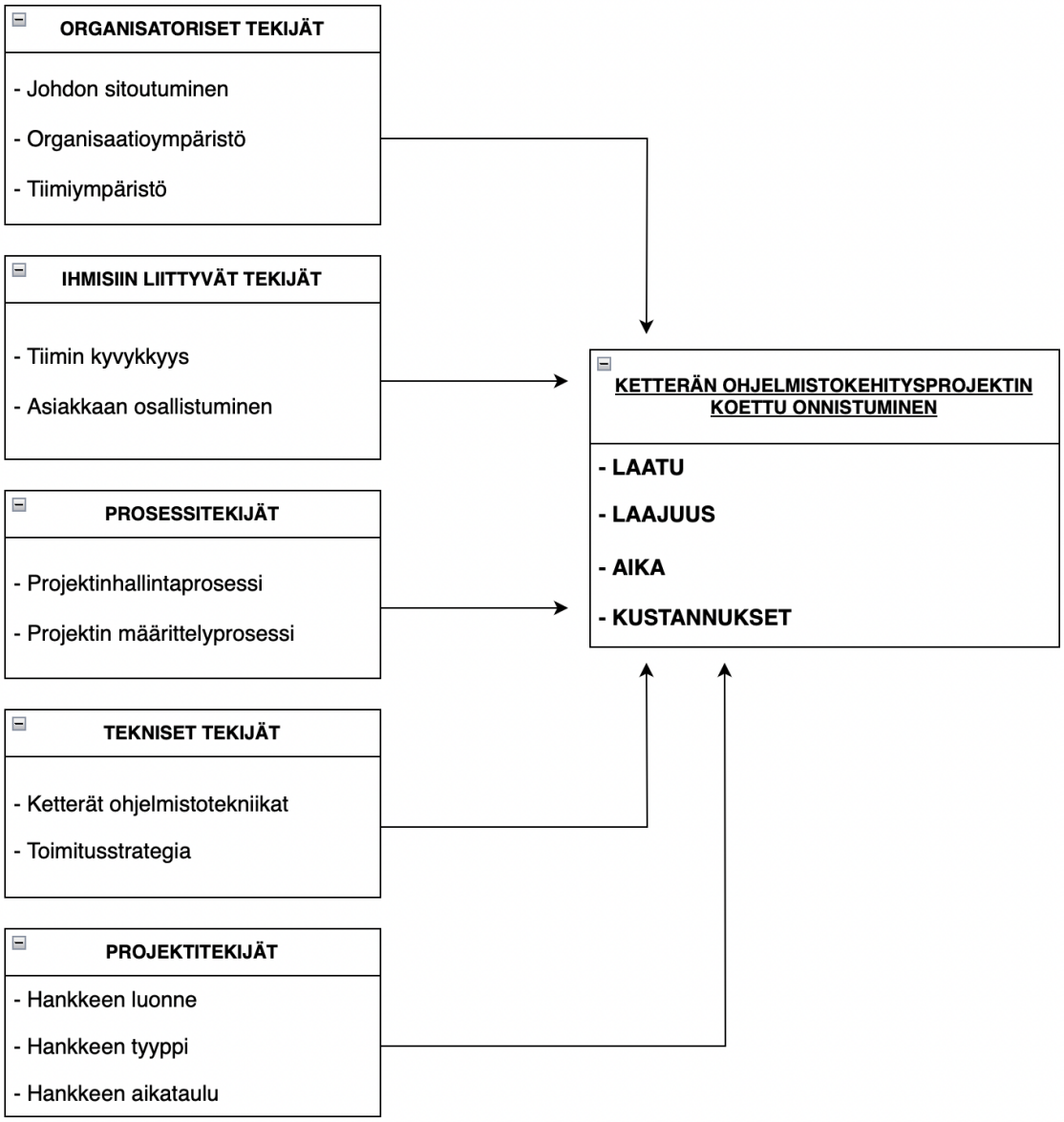
Epäonnistumista ei jaeta tietoa, kuin vain suuremmista toimijoista. Isompienkin toimijoiden epäonnistumiset jaetaan vain sen takia, että ne ovat niin julkisessa asemassa, jolloin epäonnistumisia ei voi olla käsittelemättä. Näiden toimijoiden on useasti pakko kertoa epäonnistumisesta heidän laajojen sidosryhmien osallisuuden takia. Osallisuus näkyy esimerkiksi rahallisesti, jolloin projektissa menetetty pääoma vaikuttaa suoraan myös heihin. Epäonnistumisten analysoinnin puute ja jakaminen vaikeuttaa huomattavasti ohjelmistoteknisten projektien tuntemuksen yleistä kehittämistä.

4.1 Onnistumistekijöiden kategorisointi

Taulukossa 3.1 (s.11) esitetyt neljä epäonnistumistekijöiden kategoriaa toimii päinvastaisesti onnistumistekijöinä, kun niiden epäonnistuminen pyritään estämään.

Chown ja Caon [11] tunnistamat ja määrittelemät kategoriat eli puutteet ovat organisatoriset haasteet, ihmisiin liittyvät haasteet, prosessiongelmat ja tekniset haasteet. Listatut puutteet ja ongelmat huomioimalla minimoidaan epäonnistumistekijöitä, mikä johtaa ymmärrykseen, jolloin onnistumismahdollisuus kasvaa.

Kuvassa 4.1 Chow ja Cao [11] ovat listanneet onnistumistekijöitä. Kuva havainnollistaa, miten onnistumistekijät ovat lähes samat kuin epäonnistumistekijät, mutta vain vastakohtina. Chow ja Cao ovat esittäneet vielä viidennen onnistumistekijän, joka on projektitekijät. Projektitekijöitä voi olla projektin luonne, tyyli ja aikataulu. Kuva 4.1 kuvaa, miten näiden kaikkien viiden kategorian haasteet huomioiden ketterän ohjelmistoprojektin saa onnistumaan. Onnistumismittareina pidetään laatua, laajuutta, aikaa ja kustannusta, mikä on aiemmin tutkielmassa (Kuva 2.1) käsitelty rautakolmio. Projektitekijöiden lisäksi onnistumistekijöitä Kuvassa 4.1 on aikaisemman tiedon mukaan listattu organisatoriset tekijät, ihmisiin liittyvät tekijät, prosessitekijät ja tekniset tekijät. [11]



Kuva 4.1: Onnistumistekijät. Kuva muokattu lähteestä [11]

Organisatorisiin onnistumistekijöihin luetellaan johdon sitoutuminen, organisatorinen sekä tiimin ympäristö. Ihmisiin liittyvät onnistumistekijät ovat tiimin kyvykkyys ja asiakkaan sitoutuminen. Prosessin puolestaan ovat projektinhallinta- ja määrittelyprosessi. Viimeisimmäksi vielä todettiin onnistumiseen liittyvän tekniset tekijät eli ketterät ohjelmistotekniikat ja toimitusstrategia. [11]

4.2 Epäonnistumistekijöistä ratkaisu onnistumiseen

Taulukko 3.1 (s. 12) havainnollistaa, miten huono projektin suunnittelu on yleisin epäonnistumistekijä. Yksi aikaisin suunnittelun vaihe on riskienhallinta, jolla tutkitaan onko kannattavaa aloittaa projekti ja sen laajempi suunnittelu. Riskianalyysi ja -suunnittelu ovat kaksi tärkeintä prosessia ohjelmistoprojektin riskienhallinnassa. Riskianalyysissä tutkitaan riskitekijöiden ja projektin lopputuloksen välisiä suhteita, mikä on ratkaisevan tärkeää tehokkaan riskienhallinnan kannalta. Ohjelmistoprojektien onnistuminen riippuu vahvasti riskienhallinnasta, sillä muuttuvat vaatimukset ja monimutkaiset riskitekijät voivat merkittävästi vaikuttaa hankkeiden lopputulokseen.

Riskien tehokas hallinta edellyttää sekä kattavaa analyysiä että huolellista suunnittelua, mutta perinteinen subjektiivinen harkinta ei aina riitä monimutkaisten riskien hallintaan. [21] Tämän vuoksi Hu et al. [21] tutkimuksessa esitetään integroitu lähestymistapa ohjelmistoprojektien riskien älykkääseen hallintaan, joka auttaa minimoimaan riskien vaikutuksia ja parantamaan projektin lopputuloksen ennakoitavuutta. Ehdotettu kehys koostuu kahdesta keskeisestä moduulista: riskianalyysimoduulista ja riskisuunnittelumoduulista. Riskianalyysimoduuli ennustaa projektin onnistumisen todennäköisyyttä, kun taas riskisuunnittelumoduuli laatii analyysin perusteella optimaalisen riskienhallintastrategian, joka minimoi kustannukset ja parantaa projektin onnistumismahdollisuuksia. Jotta riskien analysointi ja suunnittelu voidaan toteuttaa tehokkaasti.[21]

Hu et al. [21] tutkimuksessa ehdottavat uutta monista moniin -toimintatietämyksen löytämismenetelmää. Sen avulla voidaan tunnistaa parhaat mahdolliset toimenpiteet riskien lieventämiseksi, mikä on hyödyllistä myös muille ohjelmistohankkeille. Kehyksen tavoitteena on tarjota projektin sidosryhmille älykäs päätöksenteon tukiväline, jonka avulla riskienhallinta voidaan integroida saumattomasti osaksi projektinhallintaa. Kokonaisuudessaan riskienhallinnan mer-

kitys ohjelmistoprojekteissa on kriittinen, sillä hyvin suunniteltu ja toteutettu riskienhallinta voi merkittävästi vähentää epäonnistumisen mahdollisuutta ja parantaa projektin ennakoitavuutta. [21]

Kommunikaatio on yksi yleisimmistä syistä ohjelmistoprojektien epäonnistumiseen. Puutteellinen tai epäselvä viestintä voi johtaa väärinkäsityksiin, motivaation laskuun ja projektin etenemisen hidastumiseen. Erityisesti kokouksissa, jotka ovat keskeinen väline tiedon jakamiseen ja ongelmien käsittelyyn, huono vuorovaikutus voi heikentää tiimin tehokkuutta ja projektin onnistumista. Tätä ongelmaa voidaan kuitenkin minimoida kehittämällä ja hyödyntämällä työkaluja, jotka auttavat arvioimaan ja parantamaan kokouksissa tapahtuvaa vuorovaikutusta. Työkalut voivat auttaa tiimiä saamaan selkeämmän kuvan siitä, miten hyvin he kommunikoivat kokouksissa ja missä asioissa on parantamisen varaa. [22]

Klunderin et al. [22] tapaustutkimus osoittaa, että kokoukset ovat tehokkaita tiedon jakamisen ja ongelmien käsittelyn välineitä, mutta myös, että ongelmien esittäminen on positiivisessa yhteydessä ratkaisujen löytämiseen. Tämä viittaa siihen, että ongelmista puhuminen on tärkeä askel kohti ratkaisujen löytymistä, ja että vuorovaikutus kokouksissa voi olla avain ongelmien tehokkaaseen ratkaisemiseen. [22]

Kommunikaation optimointia voi yhä tutkia enemmän ja parantaa tutkimalla tätä työkalusta saatua dataa. Käyttäytymisen ja työkalun esille tuomien kommunikaation ongelmat analysoimalla voidaan löytää toistuvia kaavoja. Nämä kaavat tiedostomalla kommunikaatiota saadaan tehostettua ja minimoitua epäonnistunutta kommunikaatiota. Onnistunut kommunikaatio johtaa suuremmalla todennäköisyydellä onnistuneihin projekteihin.

5 Pohdinta ja analyysi

Lesum et al. [23] tutkimuksessa, jossa kerättiin dataa 92 ohjelmistoyrityksestä Bangladeshissa, tarkasteltiin ohjelmistoprojektien tulosta heikentäviä vaikutuksia ja riskimuuttujien roolia. Tulokset osoittivat, että hyvä projektin johtaminen vahvistaa projektinhallinnan vaikutusta projektin onnistumiseen. [23] Tämä korostaa johdon merkitystä ohjelmistoprojektien kriittisenä onnistumistekijänä. Vastaavasti tämän tutkielman aiemmissa osissa on esitetty, että johdon rooli sekä tiimin osaaminen ja sidosryhmien osallistuminen ovat keskeisiä tekijöitä ohjelmistoprojektien onnistumisen kannalta (Taulukko 3.1).

Ketterässä kehityksessä näihin haasteisiin vastataan selkeillä rooleilla ja jatkuvalla vuorovaikutuksella. Tuoteomistaja, Scrum Master ja kehitystiimi muodostavat kokonaisuuden, jossa tuoteomistaja vastaa liiketoimintatavoitteista ja priorisoinnista. Scrum Master tukee tiimiä esteiden poistamisessa ja kehitystiimi toteuttaa ohjelmistoa. Käytännössä roolit voivat kuitenkin jäädä epäselviksi. Tuoteomistajalta saattaa puuttua päätösvalta, Scrum Masterin rooli typistyy kokousten vetämiseen, ja johdolla ei ole näkyvyyttä sprinttitason työhön. Tämä heikentää ohjausta ja päätöksentekoa.

Vaikka tiimi olisi kokeneista asiantuntijoista koostuva, käytännön haasteet voivat estää onnistumisen. Teknologian ja liiketoiminnan välinen kuilu aiheuttaa ristiriitoja: johto ei aina ymmärrä teknisiä rajoitteita, eikä kehittäjillä ole näkyvyyttä

liiketoimintatavoitteisiin. Tämä voi johtaa epärealistisiin aikatauluihin, huonoihin teknologiavalintoihin ja laskevaan motivaation.

Projektin eri vaiheissa tarvitaan selkeämpää vastuunjakoa ja yhteistyötä. Alussa johdon ja tuoteomistajan on määriteltävä tarpeet ja resursointi realistisesti. Kehitysvaiheessa Scrum Masterin on poistettava esteet, ja johdon tuettava strategisesti ilman mikromanageerausta. Johdon on varmistettava, että tiimillä on selkeä suunta, tarvittavat resurssit ja osaaminen.

Sprinttien yläpuolella johdon tuki konkretisoituu tavoitteiden kirkastamisena, jolloin liiketoiminnan painopisteet näkyvät backlogissa. Ilman tätä backlogiin voi päätyä teknisesti järkeviä mutta liiketoiminnallisesti vähäarvoisia tehtäviä. Toinen keskeinen tukimuoto on resursoinnin ennakointi. Kapasiteettivajeet, osaamispuutteet ja epäselvät roolit rajoittavat tiimin toimintaa. Johto voi vastata tilanteeseen lisäämällä osaamista, tuomalla ulkopuolista asiantuntemusta tai varaamalla lisäresursseja edellyttäen jatkuvaa vuoropuhelua tiimin kanssa.

Johto vastaa myös toimivan viestintäympäristön ja palautekanavien rakentamisesta. Vaikka Scrum sisältää sisäisiä palautesilmukoita, johdon tulisi hyödyntää niitä ja keskustella säännöllisesti tuoteomistajan ja Scrum Masterin kanssa. Osallistuminen katselmointeihin osoittaa arvostusta ja vahvistaa yhteyttä liiketoimintaan. Keskeistä on myös muutosten ja odotusten hallinta. Projektit epäonnistuvat usein, kun tavoitteet muuttuvat jatkuvasti tai odotukset ovat epärealistisia. Johdon tehtävänä on suojata tiimiä tältä paineelta ja neuvotella raamit sidosryhmien kanssa.

Jatkuvan oppimisen kulttuuri on toinen tapa tukea kehitystä. Koulutuksiin panostaminen, kokeilujen salliminen ja virheistä oppiminen ilman syyllistämistä tukevat tiimin kasvua. Johdon tulee näyttää esimerkkiä ja edistää avoimuutta ja kehittämistä.

Myös johdon on kehityttävä. Sertifikaatit, kuten PMP tai CSPO, syventävät ymmärrystä eri rooleista ja tukevat parempaa päätöksentekoa. Sertifikaattien mukana

tuoma asiantunteva yhteinen kieli tiimin kanssa mahdollistaa sujuvamman yhteistyön ja tehokkaamman johtamisen.

Projektityöskentelyä ja etenkin johtamista tulisi sisällyttää jo perusopetukseen. Koska asiantuntijatyö on yhä enemmän projektiluonteista, olisi hyödyllistä tuoda projektiosaamista nykyisiin kursseihin tai luoda erillisiä koulutuksia. Sertifikaatit lisäisivät myös nuorten houkuttelevuutta työmarkkinoilla.

Koulutuksen tavoitteena ei ole pelkkä tiedon jakaminen, vaan taitojen ja ymmärryksen kehittäminen. Se auttaa nuoria hahmottamaan roolinsa tiimissä, toimimaan rakentavasti ja valmistautumaan tulevaisuuden johtotehtäviin.

6 Yhteenveto

Tämä tutkimus käsittelee ketterän kehityksen menetelmiä menettelevien ohjelmistoprojektien epäonnistumistekijöitä ja miten niitä voidaan minimoida. Tämä aihe on edelleen merkittävä tutkimuksen kohde, koska epäonnistumisprosentti ohjelmistoprojekteissa on yhä 70 prosenttia.

(TK1) Tyypillisimpiä epäonnistumistekijöitä ketteriä käytäntöjä noudattavissa ohjelmistoprojekteissa ovat johdon sitoutumattomuus, kehitystiimin osaamispuutteet sekä puutteellinen tai epäselvä kommunikaatio sidosryhmien kanssa. Nämä kaikki ongelmat näkyivät myös epäonnistuneena suunnitteluna ja rajauksena. Nämä tekijät eivät ole yksittäisiä ongelmia, vaan kytkeytyvät usein toisiinsa ja muodostavat ketjureaktioita, jotka heikentävät projektin etenemistä ja lopputulosta.

(TK2) Tutkimus toi esiin useita hallintakeinoja, kuten roolien selkeyttäminen (Scrum-viitekehyksen mukaisesti), johdon strategisen tuen konkretisoiminen sprinttitason työn yläpuolella, kommunikoinnin parantaminen teknisen puolen ja liiketoiminnan välillä sekä ketterän toimintatavan kouluttaminen koko organisaatiolle. Tutkimus painottaa erityisesti johdon toiminnan avainasemaa sekä koulutuksen ja jatkuvan tiimin ja sen yksilöiden kehittymistä.

Vaikka ketterät menetelmät on suunniteltu lisäämään ohjelmistoprojektien onnistumismahdollisuuksia, ne eivät yksin riitä. Ilman osaavaa johtoa, selkeitä rooleja, tehokasta viestintää ja jatkuvaa vuorovaikutusta projektit voivat ajautua epäonnis-

tumiseen. Onnistumisen avainasemassa on jatkuva kehittyminen ja oppiminen niin yksilönä kuin tiiminäkin.

Tätä tutkimusta voitaisiin jatkaa keräämällä laajempaa analyysia johonkin tietokantaan. Voitaisiin tutkia esimerkiksi prosenttiosuuksia eri epäonnistumistekijöiden yleisyydestä. Tutkimuksissa voitaisiin perehtyä myös tarkemmin eri ketterän kehityksen menetelmiin, jolloin saataisiin tieto, mitkä niistä olisivat toimivimpia. Tutkimukset kartoittaisi epäonnistumistekijöiden yleisyyksiä ja painoarvoja erilaisissa projekteissa. Laajempi analyysi epäonnistumistekijöistä laajentaisi samalla epäonnistumisen minimointitapoja. Analyysi yhä edesauttaisi ohjelmistoprojektien onnistumisprosenttia. Laajan tallennetun analyysin avuin pystyttäisiin mahdollisesti muodostaa ohjelmistoprojekteja tukeva tietokanta ja sille käyttöliittymä.

Jatkotutkimuksissa voitaisiin myös käyttää hyväksi erilaisten tiimien haastatteluja. Selvittää epäonnistuneiden ohjelmistoprojektien johdon, kehittäjien ja asiakkaan näkökulmia. Projektin osalliset osaisivat kuvailla, mikä epäonnistui omalla osa-alueellaan. Silloin saataisiin moniulotteisempi kuva siitä, missä ja miksi ongelmia syntyy. Vaikka monet tutkielmassa käytetyt lähteet listasivat ongelmia jokaiselta osa-alueelta, niin voi olla että artikkelin luojat olivat itse mukana vain yhdellä osa-alueella, jolloin heillä ei välttämättä ole todenmukainen kokonaiskuva.

Tutkielman teemaa voisi vielä yksityiskohtaistaa tutkimalla aihetta erilaisten ohjelmistoprojektein näkökulmasta. Esimerkiksi eri toimialojen ja eri laajuudeltaan olevia projekteja. Mielenkiintoinen näkökulma jatkotutkimuksiin olisi myös tekoälyn sekä muiden kehittyvien teknologioiden vaikutus ohjelmistoprojektien onnistumiseen.

Lähdeluettelo

[1] P. Nieminen. ”3 TAPAA TUHOTA LUPAAVINKIN PROJEKTI| AVE Group”. (2023), url: https://humanpotential.fi/artikkelit/3-tapaa-tuhota-lupaavinkin-projekti/ (viitattu 08. 04. 2025).

[2] T. Thesing, C. Feldmann ja M. Burchardt, ”Agile versus Waterfall Project Management: Decision Model for Selecting the Appropriate Approach to a Project”, *Procedia Computer Science*, vol. 181, s. 746–756, tammikuu 2021, ISSN: 1877-0509. DOI: 10.1016/J.PROCS.2021.01.227.

[3] D. Ly, M. Overeem, S. Brinkkemper ja F. Dalpiaz, ”The Power of Words in Agile vs. Waterfall Development: Written Communication in Hybrid Software Teams”, *Journal of Systems and Software*, vol. 219, s. 112 243, tammikuu 2025, ISSN: 0164-1212. DOI: 10.1016/J.JSS.2024.112243.

[4] B. Holtsnider, T. Wheeler, G. Stragand ja J. Gee, ”The Problem: Why Software Projects Fail”, *Agile Development and Business Goals*, s. 11–29, tammikuu 2010. DOI: 10.1016/B978-0-12-381520-0.00002-3.

[5] V. Miettinen. ”Suomi tekee huikeita valintoja – "Julkista rahaa jaetaan näennäisedistyksellisiin projekteihin Tekniikka&Talous”. (2017), url: https://www.tekniikkatalous.fi/uutiset/suomi-tekee-huikeita-valintoja-julkista-rahaa-jaetaan-naennaisedistyksellisiin-projekteihin/cf532420-0370-3b31-828b-742ae9621d1c (viitattu 10. 04. 2025).

[6] J. Ahonen ja P. Savolainen, ”Software engineering projects may fail before they are started: Post-mortem analysis of five cancelled projects”, *Journal of Systems and Software*, vol. 83, s. 2175–2187, 11 marraskuu 2010, ISSN: 0164-1212. DOI: 10.1016/J.JSS.2010.06.023.

[7] P. Serrador ja J. K. Pinto, ”Does Agile work? — A quantitative analysis of agile project success”, *International Journal of Project Management*, vol. 33, s. 1040–1051, 5 heinäkuu 2015, ISSN: 0263-7863. DOI: 10.1016/J.IJPROMAN.2015.01.006.

[8] L. Barros, C. Tam ja J. Varajão, ”Agile software development projects–Unveiling the human-related critical success factors”, *Information and Software Technology*, vol. 170, s. 107 432, kesäkuu 2024, ISSN: 0950-5849. DOI: 10.1016/J.INFSOF.2024.107432.

[9] 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, helmikuu 2017, ISSN: 0736-5845. DOI: 10.1016/J.RCIM.2015.12.001.

[10] Skillwell. ”Ketterä kehitys lyhyesti| Skillwell”. (2024), url: https://skillwell.fi/fi/blogi/kettera-kehitys-lyhyesti (viitattu 12. 04. 2025).

[11] T. Chow ja D. B. Cao, ”A survey study of critical success factors in agile software projects”, *Journal of Systems and Software*, vol. 81, s. 961–971, 6 kesäkuu 2008, ISSN: 0164-1212. DOI: 10.1016/J.JSS.2007.08.020.

[12] K. Conboy ja N. Carroll, ”Implementing Large-Scale Agile Frameworks: Challenges and Recommendations”, *IEEE Software*, vol. 36, s. 44–50, 2 maaliskuu 2019, ISSN: 19374194. DOI: 10.1109/MS.2018.2884865.

[13] M. F. Santos, R. A. Filipe ja J. C. Cunha, "From Traditional to Agile Methodologies in Software Project Management Education: A Case Study", *Proceedings of the 2024 16th International Conference on Education Technology and Computers*, s. 491–497, syyskuu 2024. DOI: 10.1145/3702163.3702460.

[14] L. LaPrad. "The Triple Constraints of Project Management | TeamGantt". (2023), url: https://www.teamgantt.com/blog/triple-constraint-project-management.

[15] J. S. Reel, "Critical success factors in software projects", *IEEE Software*, vol. 16, s. 18–23, 3 toukokuu 1999, ISSN: 07407459. DOI: 10.1109/52.765782.

[16] T. O. Lehtinen, M. V. Mäntylä, J. Vanhanen, J. Itkonen ja C. Lassenius, "Perceived causes of software project failures – An analysis of their relationships", *Information and Software Technology*, vol. 56, s. 623–643, 6 kesäkuu 2014, ISSN: 0950-5849. DOI: 10.1016/J.INFSOF.2014.01.015.

[17] P. Mohagheghi ja M. Jorgensen, "What contributes to the success of IT projects? Success factors, challenges and lessons learned from an empirical study of software projects in the norwegian public sector", *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering Companion, ICSE-C 2017*, s. 371–373, kesäkuu 2017. DOI: 10.1109/ICSE-C.2017.146.

[18] Y. Wu, C. Liu, Z. Xu et al., "The Software Genome Project: Unraveling Software Through Genetic Principles", *Proceedings - 2024 39th ACM/IEEE International Conference on Automated Software Engineering, ASE 2024*, s. 2319–2323, lokakuu 2024. DOI: 10.1145/3691620.3695307.

[19] V. Damasiotis, P. Fitsilis, P. Considine ja J. O'Kane, "Analysis of software project complexity factors", *ACM International Conference Proceeding Series*,

s. 54–58, tammikuu 2017. DOI: 10.1145/3034950.3034989. url: https://dl.acm.org/doi/10.1145/3034950.3034989.

[20] M. A. Ziana ja J. Charles, "Prioritization of Risks in Agile Software Projects Through an Analytic Hierarchy Process Approach", *Procedia Computer Science*, vol. 233, s. 713–722, tammikuu 2024, ISSN: 1877-0509. DOI: 10.1016/J.PROCS.2024.03.260.

[21] Y. Hu, J. Du, X. Zhang et al., "An integrative framework for intelligent software project risk planning", *Decision Support Systems*, vol. 55, s. 927–937, 4 marraskuu 2013, ISSN: 0167-9236. DOI: 10.1016/J.DSS.2012.12.029.

[22] J. Klünder, N. Prenner, A. K. Windmann et al., "Do You Just Discuss or Do You Solve?: Meeting Analysis in a Software Project at Early Stages", *Proceedings - 2020 IEEE/ACM 42nd International Conference on Software Engineering Workshops, ICSEW 2020*, vol. 20, s. 557–562, kesäkuu 2020. DOI: 10.1145/3387940.3391468.

[23] S. A. Lesum, S. R. Akthar, M. R. Islam, F. Sadia ja M. Hasan, "Project Governance to Improve the Performance of Software Projects by Mitigating the Software Risk Factors: The Moderating Role of Project Leadership", *Procedia Computer Science*, vol. 239, s. 1863–1870, tammikuu 2024, ISSN: 1877-0509. DOI: 10.1016/J.PROCS.2024.06.368.