DevOpsin ja CI/CD:n käyttö, haasteet ja vaikutus ohjelmistoprojektin onnistumiseen ja laatuun

Turun yliopisto
Tietotekniikan laitos
LuK -tutkielma
Tietojenkäsittelytiede
Kesäkuu 2025
Antti Huotari

TURUN YLIOPISTO
Tietotekniikan laitos

Antti Huotari: DevOpsin ja CI/CD:n käyttö, haasteet ja vaikutus ohjelmistoprojektin onnistumiseen ja laatuun

LuK -tutkielma, 29 s.
Tietojenkäsittelytiede
Kesäkuu 2025

Viime vuosina suuren suosion saanut DevOps ja sen ketterän kehityksen käytänteet ovat tärkeä osa nykypäivän nopeatahtista ohjelmistokehitystä. DevOps on jatkettu muoto ketterästä kehityksestä (Agile), joka keskittyy ohjelmistoprojektin kehitys- ja tuotantotiimin yhdistämiseen saavuttaakseen lyhyet julkaisuajat ja tehokkaan kommunikaation. DevOpsin käyttöönotto ei kuitenkaan ole aina onnistunut ja DevOps voi aiheuttaa haasteita niin organisaation, kuin ohjelmiston tasolla. Kandidaatintutkielma suoritettiin kirjallisuuskatsauksena ja siinä tarkastellaan DevOpsin ja Jatkuvan Integroinnin ja Toimituksen (CI/CD) vaikutuksia ohjelmistoprojektin onnistumiseen ja laatuun. Pyritään tuomaan esille haasteita, mitä DevOpsin käyttöönoton aikana saattaa ilmetä ja miten näitä haasteita voidaan ratkaista ja ennaltaehkäistä. Tutkielmassa havaittiin, että DevOpsin käytänteet ja etenkin CI/CD vaikuttavat positiivisesti niin ohjelmistoprojektin onnistumiseen, kuin myös ohjelmistotuotteen laatuun. Haasteita DevOpsissa löytyi erityisesti kulttuurissa, kommunikaatiossa ja DevOpsin käytänteiden puutteellisessa tuntemuksessa. Haasteisiin löytyi ratkaisuja kommunikaation, kultturin ja yhteistyöllisen tiimityön edistämiseksi, kuten johdon sitoutuminen ja DevOps-käytänteiden koulutus. DevOpsin tarkkoja vaikutuksia ohjelmistoprojektin onnistumiseen ja laadun osa-alueisiin on esitetty varsin pinnallisesti ja vertailu havaittin haastavaksi yhteisten standardien puuttumisen vuoksi.

Asiasanat: DevOps, CI/CD, ohjelmistotuotanto, ohjelmistokehitys, critical success factors, ISO 25010, CAMS

Sisällys

1 Johdanto

Viimeisenä kahtena vuosikymmenenä ohjelmistot ovat kasvaneet välttämättömäksi osaksi teollisuutta ja ohjelmistoista on tullut arkipäiväistä niin yrityksien toiminnassa, kuin myös kuluttajilla. Tämä on osoittanut, että ohjelmistoista riippuvaisia tuotteita ja palveluita tarvitaan, jotka ovat johdonmukaisesti luotettavia, hyödyllisiä ja turvallisia käytön aikana. [1], [2] Näiden ohjelmistojen varma ja kustannustehokas suunnittelu, järjestäminen ja hallinnointi vaatii selkeän standardikehyksen, jota ohjelmistokehitysprosessi noudattaa [1]. Ohjelmistoprojektin kehitys- ja julkaisunopeus ovat ohjelmiston laadun kanssa avainasemassa menestyksessä ohjelmistoteollisuudessa, joten monia erilaisia menetelmiä on kehitetty tulosten maksimointiin [2].

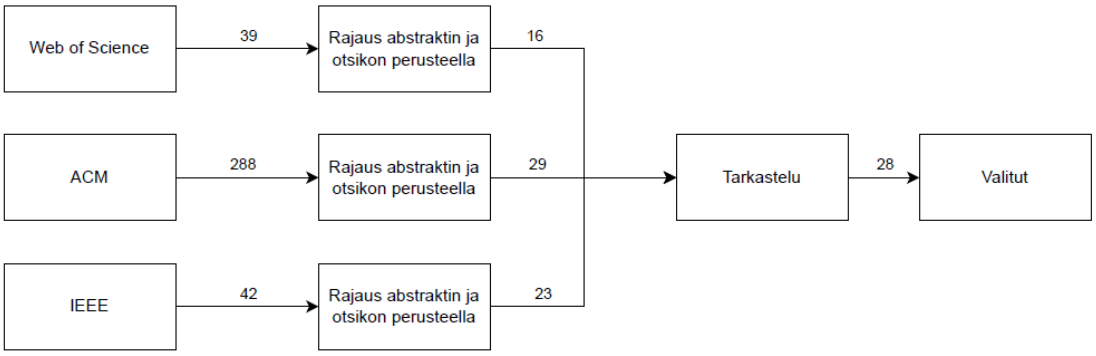
Nykypäivänä yksi suosituimmista menetelmistä on ketterä kehitys (Agile), jonka tärkeimmät ominaisuudet ohjelmistotuotannossa ovat mukautuvuus, yhteistyö ja työn järjestelmällinen eteneminen [3]. DevOps on ketterästä kehityksestä jatkettu ohjelmistotuotannon menetelmä, joka tuo ohjelmistoprojektin kehitystiimin ja tuotannon yhteen [1], [2]. Tässä tutkielmassa tutkitaan kirjallisuuskatsauksen avulla DevOpsia ja sen parhaita käytänteitä ja tarkastellaan sen vaikutusta ohjelmistoprojektin onnistumiseen ja laatuun, haasteet huomioonottaen.

Tutkielman tutkimuskysymykset ovat:

- TK1: Miten DevOps ja CI/CD vaikuttavat ohjelmistoprojektin onnistumiseen ja laatuun?

- TK2: Mitkä ovat DevOps-käyttöönoton yleisimmät haasteet ja miten niitä voidaan ratkaista, tai ennaltäehkäistä?

Tiedonhakuun käytettiin Web of Science-, IEEE Xplore-, ACM Digital library- ja Volter - tietokantoja. Tiedonhaku suoritettiin hakulauseella: "("DevOps"OR "CI/CD") AND ("impact"OR "effect"OR "influence") AND ("software development"OR "software production") AND ("success"OR "quality")". Tässä tiedonhaussa keskityttiin löytämään artikkeleita, jotka käsittelivät DevOpsia tai CI/CD:tä aihetasolla, tutkivat niiden käytön merkitystä tai haasteita ohjelmistokehityksessä. Tämän tiedonhakuprosessin tarkemmat luvut löytyvät Kuvasta 1.1. Saaduista hakutuloksista karsittiin otsikon ja abstraktin perusteella aiheeseen sopivimmat artikkelit lukua varten, jonka jälkeen ne valittiin varsinaisesti käytettäväksi tämän tutkielman lähteenä.



Kuva 1.1: Tiedonhakuprosessi

Lukua 3 varten suoritettiin täydentävä tiedonhaku DevOpsin kriittisistä menestystekijöistä ja DevOpsin haasteista. Käytetty hakulause kriittisiä menestystekijöitä koskevan kirjallisuuden haussa: "("DevOps"AND ("critical success factors"OR "CSF"))". DevOpsin haasteita koskevan kirjallisuuden haussa: "("DevOps"AND ("challenges"OR "issues"OR "complications"))".

Tutkielman toisessa luvussa perehdytään DevOpsin taustoihin ja historiaan. Luvussa esitellään DevOpsin parhaita käytäntöjä ja perehdytään niihin lyhyesti. Kol-

mannessa luvussa tarkastellaan DevOpsin ja Jatkuvan Integraation ja Jatkuvan Toimituksen (CI/CD) vaikutusta ohjelmistoprojektin onnistumiseen. Neljäs luku sisältää katsauksen mahdollisista haasteista, joita ohjelmistoprojekti voi kohdata DevOpsin käyttöönotossa. Luvussa myös käsitellään näiden haasteiden mahdollisia ennaltaehkäiseviä ratkaisuja. Viides luku sisältää pohdintaa ja erilaisia näkökulmia aiheesta. Kuudennessa luvussa tuodaan yhteen tutkielman havainnot ja vastataan tutkielman tutkimuskysymyksiin.

2 DevOpsin taustat ja menetelmät

Ohjelmistokehittäjät ovat ohjelmistotuotannon ensimmäisistä vuosista lähtien yrittäneet etsiä parhaita tapoja kehittää ja toimittaa ohjelmistoja. Nykyajan yritykset myös etsivät tapoja kehittää ja toteuttaa korkealaatuisia ohjelmistoja täyttääkseen asiakkaiden ja markkinoiden vaatimukset. [4] Optimaalisen toiminnan varmistamiseksi, näistä malleista kaikki noudattaa mallille ja projektille ominaista runkoa, joka koostuu vaiheista [1]. Tässä luvussa tarkastellaan DevOpsin taustoja, menetelmiä ja käytänteitä. Esitellään myös työkaluja, joita voidaan käyttää DevOpsin eri vaiheissa.

2.1 Ketterä kehitys (Agile)

Ohjelmistokehityksellä on aina jonkinlainen elämänkaari riippumatta käytetystä mallista. Näitä malleja ovat mm. Big-Bang, V-, Iterative, Spiral ja Vesiputous -malli. [1] Viime vuosina ohjelmistokehityksessä käytetyt mallit ovat muuttuneet perinteisestä vesiputousmallista ketterän kehityksen käytänteisiin. [4] Ketterä kehitysmalli (Agile) esiteltiin vuonna 2001 ja sen jälkeen yritykset alkoivat pian käyttää ketteriä käytänteitä noudattavia kehitysmenetelmiä, kuten Scrum ja Kanban. [1] Ketterä kehitysmalli puoltaa pieniä julkaisuiteraatioita, joihin sisältyy asiakasarviointeja. Se olettaa, että tiimi voi julkaista ohjelmistoa usein tuotantoympäristöä muistuttavassa ympäristössä. Kuitenkin, varhainen ketterän kehityksen kirjallisuus painottaa vain vähän käyttöönottoon liittyviä käytäntöjä. Esimerkiksi Extreme Programming (XP) -käytännöissä ei keskitytä käyttöönottoon. Sama vähäinen huomio käyttöönottoon

on havaittavissa myös perinteisissä ohjelmistotekniikan prosesseissa ja kirjallisuudessa. [5], [6] Tämän seurauksena siirtyminen tuotantoon on usein stressaava prosessi organisaatioissa, sisältäen manuaalisia, virhealttiita toimintoja ja viime hetken korjauksia [5].

Agile-tiimi koostuu keskenään yhteistyötä tekevistä kehitys-, testi- ja liiketaloustiimeistä, joka edistää nopeaa ja ympäristöönsä mukautuvaa ohjelmistokehitystä. Agile kuitenkin keskittyy pääosin vain itse ohjelmiston kehitysosioon ja sen julkaiseminen jää erillisen tuotantotiimin tehtäväksi. Tämä on noussut ongelmaksi etenkin julkaisuvaiheessa. [7] Kehitystiimi suunnittelee, mallintaa ja rakentaa ohjelmiston alusta lähtien, ja erillinen tuotantotiimi suorittaa saadun tuotteen testauksen ja käyttöönoton. Tuotantotiimi antaa palautetta havaituista bugeista ja tarvittavista korjauksista. Tuotantotiimin työnteon aikana kehitystiimi oli kuitenkin toimettomana odottaessaan palautetta tuotoksesta. Joissakin tapauksissa kehitystiimi oli jo siirtynyt seuraavaan projektiin samalla, kun tuotantotiimi jatkoi palautteen antamista edellisestä koodista. [8] Tämä johtaa selvästi pitkiin odotusaikoihin, julkaisuprosessin hidastumiseen ja heikkoon kommunikaatioon, joka voi aiheuttaa virheitä ja väärinkäsityksiä [7].

2.2 DevOpsin päätavoitteet ja menetelmät

DevOps on jatkettu muoto ketterästä kehityksestä, joka tuli käsitteenä ensin esille vuonna 2008 ja pian tämän jälkeen sen suosio levisi laajasti ohjelmistokehitykseen ympäri maailman. [2] Termi "DevOps"viittaa joukkoon käytänteitä, jotka edistävät yhteistyötä ja viestintää kehitystiimin (Development) ja tuotantotiimin (Operations) välillä. DevOps-mallin käytön tavoitteena on ohjelmistojen ja palvelujen toimitus nopeasti ja johdonmukaisesti, samalla säilyttäen laadukkuuden. Tämä näkyy projektissa Jatkuvan Toimituksen ja Käyttöönoton tuottamisena, jotta ohjelmisto saadaan toimitettua nopeasti lyhyillä kehityssykleillä. [1], [9]

DevOpsin päätavoitteet organisaatiossa ovat: [1]

- Tuottaa mittavia liiketoimintahyötyjä toimittamalla luotettavia ja ensiluokkaisia palveluita.
- Ketteryys ja yksinkertaisuuden korostaminen ovat keskeistä kaikilla osa-alueilla: prosesseissa, teknologiassa ja ihmisissä.
- Kehittämällä keskinäisen luottamuksen ja vastuunoton kulttuuria kannustaen innovaatioon ja edistäen yhteistyötä, voi muurit kehitystiimin ja tuotantotiimin välillä murtaa.
- Dynaamisen vaatimustenmukaisuuden hallinta käyttöoikeuksia ja tiedonjakamista koskevien lakien muuttuessa.

DevOpsin pohjimmainen idea on yhdistää kaikki tekijät, jotka ovat tekemisissä ohjelmiston tuotannossa. DevOps tarjoaa ketterään kehitykseen täydentävää joukkoa käytäntöjä, jotka mahdollistavat ohjelmiston iteratiivisen toimituksen lyhyissä sykleissä tehokkaasti. [5], [7] Ohjelmistoprojektin kehitys- ja tuotantotiimi yhdistetään yhdeksi tiimiksi, jossa kehittäjät toimivat yhteistyössä koko ohjelmiston kehityksen elämänkaaren (SDLC) laajuisesti. [8] Tyypillinen DevOps-tiimi koostuu yleensä DevOps-johtajasta tai -arkkitehdista, sekä useista eri kokemustasoilla (junior, keskitaso ja senior) olevista kehittäjistä, jotka jakavat samankaltaiset taidot [9]. DevOps-ohjelmistoprojektissa yhteen tuodut tiimit toimivat tietyssä roolissa ja vastuudet on selvästi jaettu tietyille henkilöille tiimin sisällä. Ohjelmiston kehitysvaiheessa tiimi tekee yhteistyötä asiakkaan kanssa projektin suunnittelussa, jonka jälkeen kehittäjät aloittavat ohjelmiston teknisen toteutuksen. [10]

DevOpsille ei kuitenkaan ole yhtä yleistä määritelmää, vaan sen käyttö muovautuu käyttöympäristönsä mukaan [11]. DevOps on ennen kaikkea kulttuurillinen muutos organisaatiossa ja kehityksessä käytetyt työkalut ovat vain vahvistamassa tehokkuutta. [9], [12]

DevOps voidaan nähdä konseptuaalisena viitekehyksenä, joka perustuu tiettyihin kyvykkyyksiin ja keskittyy CAMS-akronyymiin (Culture, Automation, Measurement, and Sharing). [11] Jotta DevOpsin käyttöönotto olisi kestävää, organisaatioiden on muutettava kulttuuriaan, omaksuttava automaatio, otettava käyttöön mittaamiskäytännöt ja edistettävä tiedon ja osaamisen jakamista.

Kulttuurilla on keskeinen rooli DevOpsin käyttöönotossa, sillä organisaatioiden on asetettava etusijalle korkealaatuisen palvelun tarjoaminen ja kehitettävä DevOps-ajattelutapa. Tämä sisältää jatkuvan oppimisen, kokeilun, tuotelähtöisen asenteen, insinöörikulttuurin, sekä laadun korostamisen. Automaatio on yksi näkyvimmistä osista DevOpsissa. Rutiininomaiset tehtävät, kuten testien ajamiset tehdään automaattisesti, joka säästää tiimin aikaa ja antaa mahdollisuuden keskittyä itse kehitystehtäviin. Automaatio johtaa ennustettaviin ja standardoituihin tuloksiin, joka parantaa ohjelmiston toimituksen tehokkuutta, nopeutta ja luotettavuutta ja tuo täten enemmän arvoa projektille.

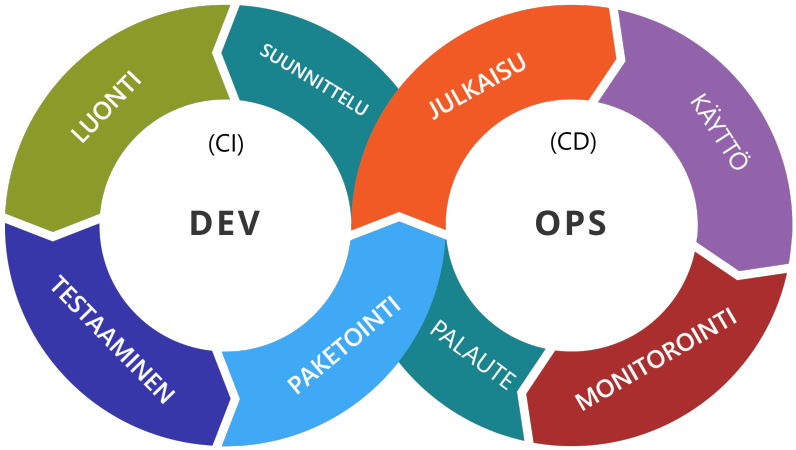
Tulosten mittaaminen on tehokas tapa jatkuvaa toiminnan kehittämistä varten. Jokapäiväisten toimintojen tehokkuuden mittaamisen ja monitoroinnin avulla helpotetaan dataan pohjautuvaa päätöksentekoa. Tämä myös helpottaa palautteen kiertoa tuotanto- ja kehitystiimin välillä, josta päästään tiedon jakamisen tärkeyteen organisaatiossa. Jakaminen on myös tärkeä osa DevOps-prosessia, sillä se edistää tiimien välistä yhteistyötä. DevOpsissa korostetaan usein epäonnistumista saadakseen nopeaa palautetta ja nopeuttaakseen toimitusta. Onnistumisten ja epäonnistumisten jakaminen organisaatiossa on edellytys onnistuneelle DevOps-käyttöönotolle. [13], [14]

2.3 Jatkuva Integraatio (CI) ja Jatkuva Toimitus (CD)

Jatkuva Integraatio (Continuous Integration), Jatkuva Toimitus (Continuous Delivery), ja Jatkuva Käyttöönotto (Continuous Deployment), eli CI/CD toimivat DevOpsin perustana ja ovat tulleet olennaiseksi osaksi ohjelmistokehitystä [10], [13]. CI/CD:n tavoitteena on automatisoida ja virtaviivaistaa ohjelmistopäivitysten ja muutosten toimitus loppukäyttäjälle [15]. Jatkuva Integraatio (CI) on kehityskäytäntö, joka kattaa ohjelmiston muutosten rakennus- ja testausprosessin automatisoinnin. Tämä mahdollistaa virheiden havaitsemisen varhaisessa vaiheessa, parantaa koodin laatua ja edistää tehokasta rinnakkaista kehitystä. CI eroaa perinteisestä manuaalisesta testauksesta, joka on usein hitaampaa ja virhealttiimpaa. Automaattinen testaus CI:n yhteydessä vähentää ihmimillisten virheiden riskiä ja parantaa ohjelmiston toimitusvarmuutta. [10], [13]

Jatkuva Toimitus ja Käyttöönotto (CD) viittaa prosessin seuraavaan vaiheeseen, jossa testattu ja hyväksytty koodi siirretään automaattisesti erilaisiin ympäristöihin, kuten kehitys- (DEV), laadunvarmistus- (QA), ja esituotantoympäristöihin (STAGE). Tavoitteena on varmistaa, että muutokset toimivat luotettavasti ja soveltuvat tuotantoon. CD automatisoi tämän prosessin vähentämällä huomattavasti manuaalisiin tuotantokeinoihin vaadittua aikaa ja työtä. Ohjelmisto voidaan julkaista nopeasti ja toistettavasti ilman manuaalisia toimenpiteitä, mikä vähentää käyttöönottoon liittyviä riskejä ja lyhentää julkaisusyklejä. [13] Kuvassa 2.1 esitetään DevOpsin-prosessin kulku.

DevOps-prosessin tekninen ydinelementti on automaatio. Jokaisen koodimuutoksen tallennuksen jälkeen muutokset viedään versionhallintaan ja automaatiojärjestelmä käynnistää koodin kokoamisen ja testauksen. Samalla suoritetaan muun muassa yksikkötestaus, koodin tarkistus ja validointi suoraan kooditasolla. Testit



Kuva 2.1: DevOps prosessi (perustuu kuvaan [16]).

ajetaan muutosten jälkeen automaattisesti ja hyväksyttyjen testien jälkeen muutokset julkaistaan automaattisesti. Jos testit epäonnistuvat, ei muutoksia lähetetä julkaistavaksi. Nämä vaiheet kuuluvat Jatkuvaan Integraatioon (CI). [10]

Jatkuva Toimitus ja Käyttöönotto (CD) laajentaa tätä prosessia viemällä valmiit ja testatut ohjelmistokomponentit automaattisesti testaus- ja tuotantoympäristöihin. Onnistuneen integraation jälkeen koodi siirretään testipalvelimille käyttäjän hyväksyntätestausta (UAT) varten osana Jatkuvan Toimituksen prosessia. [10], [13] Tämän jälkeen ohjelmiston toimivuutta monitoroidaan asiakkaan kanssa. Julkaisun jälkeinen jatkuva monitorointi on olennaista sovelluksen suorituskyvyn ja vakauden ylläpitämiseksi. Näin mahdollisiin ongelmiin voidaan reagoida nopeasti ja ohjelmiston laatu voidaan varmistaa myös käyttöönoton jälkeen. [13]

Onnistuneen DevOps -kulttuurin ja CI/CD:n käyttöönotto organisaatiossa vaatii monia erilaisia työkaluja, jotka ovat kohdennettuja tiettyihin vaiheisiin Jatkuvassa Integraatiossa, Toimituksessa ja Käyttöönotossa.

Versionhallintajärjestelmä, kuten Git [1] mahdollistaa useiden käyttäjien koodin jakamisen tiimissä. Kehitysympäristö (IDE) vaihtelee organisaatioittain ja projektin mukaan, mutta suosittuja vaihtoehtoja ovat mm. Microsoft Code [2] ja JetBrainsin

[1] `https://git-scm.com/`

[2] `https://code.visualstudio.com/`

IDE:t [3].

Koodimuutosten kokoamiseen suosittuja työkaluja ovat mm. Maven [4] ja Gradle [5]. Tämä edistää käyttövalmiiden ohjelmistokomponenttien luomista. Niillä voidaan automatisoida useita vaiheita, kuten kokoaminen (build), testaaminen, julkaisu, ja käyttöönotto ilman tarvetta manuaaliselle väliintulolle. [17]

Testaamiseen suosittuja työkaluja ovat JUnit [6] ja Se [7]. Niitä käytetään testitapausten ajamiseen, mukaan lukien yksikkö- ja integrointitestit. [13]

Konfiguraatiotyökaluja, joita käytetään tehtäviin, kuten palvelinten päivittäminen, sovellusten asentaminen useille palvelimille samanaikaisesti ovat mm. Puppet [8] ja Ansible [9].

Monitorointiin käytettyjen työkalujen, kuten Nagios [10] ja Splunk [11] avulla voidaan suorittaa proaktiivista valvontaa, joka havaitsee viat varhain estääkseen palvelinvauriot, tai käyttökatkokset. [13]

[3] https://www.jetbrains.com/ides/
[4] https://maven.apache.org/
[5] https://gradle.org/
[6] https://junit.org/junit5/
[7] https://www.selenium.dev/
[8] https://www.puppet.com/
[9] https://docs.ansible.com/
[10] https://www.nagios.org/
[11] https://www.splunk.com/

3 DevOpsin ja CI/CD:n vaikutukset ohjelmistoprojektin onnistumiseen ja laatuun

Viime vuosien aikana DevOps on saavuttanut huomattavaa suosiota, mikä on osittain selitetty sillä, että sen avulla organisaatiot voivat kehittää ja parantaa tuotteitaan nopeammassa tahdissa verrattuna perinteisiin ohjelmistokehitysmenetelmiin. Tämän vuoksi on tärkeää tarkastella tätä lähestymistapaa ja selvittää miten DevOpsin käyttöönotto vaikuttaa projektin onnistumiseen ja laatuun, ja mitkä tekijät ovat mahdollisten vaikutusten takana. [2]

Tässä luvussa tarkastellaan, miten luvussa 2 esiteltyjen DevOpsin ja Jatkuvan Integraation, Toimituksen ja Käyttöönoton (CI/CD) käyttöönotto vaikuttaa ohjelmistoprojektin onnistumiseen ja lopputuotteen laatuun. Tutkitaan DevOpsin onnistumista ja luotettavuutta ohjelmistotuotannon kriittisten onnistumistekijöiden kautta ja esitellään menetelmiä, miten mitata onnistumista ja ohjelmiston laatua. Tässä luvussa vastataan TK1: "Miten DevOps ja CI/CD vaikuttavat ohjelmistoprojektin onnistumiseen ja laatuun?".

3.1 Kriittiset menestystekijät ja DevOps-projektin onnistumisen määrittely

Ohjelmistokehitysprojektien onnistumisen taustalla vaikuttaa joukko tekijöitä, joita kirjallisuudessa kuvataan kriittisiksi menestystekijöiksi (critical success factors). Nämä tekijät muodostavat olennaisen perustan projektien suunnittelulle, toteutukselle ja lopulliselle onnistumiselle. Kriittiset menestystekijät kattavat ominaisuuksia, olosuhteita ja muuttujia, jotka asianmukaisesti hallittuna, ylläpidettynä ja tuettuna voivat merkittävästi vaikuttaa koko organisaation menestykseen. [18]–[20] DevOpsin kriittiset menestystekijät voidaan jakaa Azadin (2022) mukaan kolmeen osaan: teknisiin (Technical), organisaation (Organizational), ja sosiaalis-kulttuurisiin (Social and cultural) tekijöihin. [9] Tekijä voidaan tulkita kriittiseksi, jos sillä on merkittävä vaikutus jatkuviin käytäntöihin ja se edistää prosessin kokonaisvaltaista onnistumista. [18], [21] Taulukossa 3.1 on koottu yhteen kirjallisuuskatsauksessa löydettyjä kriittisiä menestystekijöitä ja havainnollistetaan löydettyjen tekijöiden esiintymistä kirjallisuuskatsauksessa. Tekijät ovat valikoituneet lähteistä, jotka tutkivat DevOpsin, Ketterän Kehityksen, tai jatkuvia menetelmiä käyttävien ohjelmistoprojektien onnistumistekijöitä. Tekijät ovat valittu tähän tutkielmaan käytettäväksi jatkuvan toistuvuuden perusteella kirjallisuuskatsauksessa käytetyistä lähteistä.

Kirjallisuuskatsauksessa löydetyistä kriittisistä menestystekijöistä Taulukossa 3.1 voidaan tulkita, että tiimityö ja yhteistyö organisaation sisällä, johdon sitoutuminen, kommunikaatio, osaaminen ja koulutus, ja asiakkaan osallistaminen ja tyytyväisyys ovat hyvin keskeisiä tekijöitä ohjelmistoprojektin onnistumisessa. Tarkastellaan lyhyesti näitä havaittuja kriittisiä menestystekijöitä. Tekijät Kulttuurillinen muutos (T5), Automaatio (T6) ja Metriikat ja seuranta (T8), jotka ovat käsitelty jo luvussa 2, jätetään tässä osassa käsittelemättä.

Jayakodyn ja Wijayanayaken (2023) tutkimuksen mukaan kollaboratiivisen kult-

Taulukko 3.1: Tekijöiden esiintyminen kirjallisuuskatsauksessa

Lähde	T1: Tiimityö ja yhteistyö	T2: Johdon sitoutuminen	T3: Moniosaavat tiimit	T4: Kommunikaatio	T5: Kulttuurillinen muutos	T6: Automaatio	T7: Osaaminen ja koulutus	T8: Metriikat ja seuranta	T9: Turvallisuus	T10: Asiakkaan osallistaminen
Azad (2022) [9]	x	x	x	x	x	x			x	
Azad et al. (2024) [18]	x	x		x	x	x		x	x	
Niazi et al. (2006) [22]	x	x		x			x			x
Jayakody et al. (2023) [23]	x	x	x	x	x	x	x	x	x	x
Rafi et al. (2022) [24]	x	x	x	x	x	x	x	x		x
Aleu & Aken (2016) [25]	x	x	x	x	x	x	x	x		x
Chow & Cao (2008) [26]	x	x		x			x			x
Sudhakar (2012) [27]	x	x		x			x	x		x

tuurin rakentaminen on DevOpsin käyttöönoton kriittisin menestystekijä. Tähän voidaan sisältää Taulukossa 3.1 mainitut tekijät Tiimityö ja yhteistyö (T1), Johdon sitoutuminen (T2), Kommunikaatio (T4) ja Kulttuurillinen muutos (T5). [23] Läheinen, säännöllinen ja yhteistyöhön perustuva tiimityö on keskiössä ohjelmistokehityksessä erityisesti ketterissä menetelmissä. Agile-liike ohjelmistokehityksessä korostaa vahvojen ihmissuhteiden rakentamista ja yhteisöllisyyden vaalimista kehittäjien keskuudessa. Sen sijaan, että kehitys perustuisi pelkästään institutionalisoituihin prosesseihin ja työkaluihin, ketterät käytännöt asettavat etusijalle läheisten tiimien muodostamisen ja yhteistyöhön perustuvan työympäristön luomisen. [18]

Johdon sitoutuminen ja tuki on laajasti kirjallisuudessa esiintynyt menestystekijä DevOpsin käyttöönotossa ja on esiintynyt DevOpsin kriittisten menestystekijöiden tapaustutkimuksissa mainituimpana tekijänä [23], [27]. DevOps-projektissa johtamisen ja järjestyksen puute voi johtaa sekavaan tiimityöhön. On havaittu, että DevOps-ammattilaiset ovat valmiita vastaanottamaan ohjeita esihenkilöiltään ja

johdon, tai tiiminvetäjien antamien ohjeiden saaminen on keskeistä kehitysprosessin sujuvan etenemisen kannalta. [9] Toisaalta Chowin ja Caon (2008) mukaan ohjelmistoprojektin menestyksessä johdon rooli on enemmän suuntaa antava ja kannustava. [26].

Kommunikaation tärkeys ei ole ainoastaan DevOpsin, tai muiden ketterien menetelmien tärkeä menestystekijä, vaan yleisesti hyvä kommmunikaatio organisaation sisällä ja ulkopuolella asiakkaan kanssa on elintärkeää projektin onnistumisen ja laadun kannalta. Selkeä ja jatkuva kommunikaatio on keskeistä varmistamisessa, että jokainen yksilö organisaatiossa ymmärtää sen tavoitteet ja päämäärän DevOpsin käyttöönotossa ja työskentelee kohti samoja päämääriä. [18], [22]

Tiimin osaaminen, motivaatio ja DevOps-koulutus nousi hyvin yleiseksi menestystekijäksi kirjallisuuskatsauksessa. Tiimissä täytyy olla niin teknistä osaamista, kuin myös vuorovaikutustaitoja, kuten kommunikaatio-, tiimityö-, motivaatio-, ja järjestelmällisyystaitoja. Näillä vuorovaikutustaidoilla tiimin jäsenet täydentävät DevOps-osaamista ja teknisiä taitoja. Näiden lisäksi kattava koulutus organisaation DevOps-työskentelytapoihin auttaa saavuttamaan menestyksekkään DevOps-tiimin. [23]

Tietoturva ja turvallisuus DevOps-projektin tuotoksessa on otettava huomioon jo projektin alusta lähtien. Ohjelmiston turvallisuudessa on tavoite ymmärtää etukäteen mahdollisten hyökkääjien motiivit ja suunnitella, kuinka käsitellä mahdolliset uhkatilanteet. Kun turvallisuusuhat otetaan huomioon kehitysprosessin aikana, vaikeutuu hyökkääjien tavat aiheuttaa uhkia. [18], [23]

Moniosaavat tiimit tulivat kirjallisuuskatsauksessa tekijänä esille erityisesti suoraan DevOpsia koskevissa artikkeleissa. Moniosaavat tiimit sisältävät laajan kattauksen toiminnallisia rooleja ja osaamista ja mm. Akbarin et al. (2020) tutkimuksessa tuli esille, että moniosaavat ja DevOps-osaamista sisältävät tiimit vaikuttavat organisaation onnistumiseen. [9], [12], [25]

Viimeisenä, asiakkaan osallistaminen projektiin on tärkeää räätälöinnin kannalta ja erityisesti muutosten kannalta, johon DevOps on sopiva menetelmä. Asiakkaan tyytyväisyys määrittää viimeistään, miten onnistunut ohjelmistoprojekti on. Kirjallisuuskatsauksessa tuli myös esille, että CAMS-periaatteita käytetään yleisesti DevOpsin menestystekijöinä. [23]

Tutkijoiden keskuudessa ei ole yhtä yhteistä kehystä määrittelemään ohjelmistoprojektin onnistumista [28]. Nykyaikaisessa (2020-2025) kirjallisuudessa on myös varsin heikosti esitetty laajaa onnistumisen kriteerien kehystä, joten tätä lukua varten etsittiin kirjallisuuskatsauksessa löytyneitä onnistumisen kriteerejä ja koottiin ne yhteen kirjallisuudessa esiintymisen perusteella. Taulukossa 3.2 esitetään kirjallisuuskatsauksen perusteella valitut kriteerit ohjelmistoprojektin onnistumisen mittaamiseen ja näiden valittujen kriteerien todettu täyttyminen DevOpsin vaikutuksia tutkivissa lähteissä.

Freferin et al. (2018) [28] suorittamassa tutkimuksessa kerättiin ohjelmistoprojektin onnistumisen mittaamisen kriteerejä eri lähteistä. Näistä havaituista kriteereistä otettiin Taulukkoon 3.2 mukaan ne, jotka mainittiin tutkimuksen yhdeksästä lähteestä vähintään neljä kertaa. Tarkastellaan lyhyesti näitä kriteerejä ja mitä niiden täyttyminen ohjelmistoprojektissa tarkoittaa.

Aikataulussa pysyminen ja projektiin käytetyn ajan minimointi on huomattu olevan yksi näkyvimmistä kriteereistä ohjelmistoprojektin onnistumiselle ja organisaation näkökulmasta yksi tärkeimmistä vaikutuksista on julkaisuajan lyheneminen. [29] Projektille määrätyssä budjetissa pysyminen on selkeästi keskeinen kriteeri ohjelmistoprojektin liiketoiminnalliselle kannattavuudelle. DevOps muovautuu tehokkaasti muutoksiin, joten kustannuksien nousua projektin aikana voidaan hallita DevOpsilla. [13], [29] Asiakkaan tyytyväisyys on ohjelmistoprojektille tärkeä tavoite ja määrittelee lopuksi, kuinka onnistunut projekti on. Ohjelmistoon ja sen takana olevaan prosessiin tyytyväinen asiakas voi tuoda yritykselle uusia asiakkaita ja vah-

Taulukko 3.2: Havaitut onnistumisen kriteerit ja niiden toteutuminen DevOpsissa

Lähde	K1: Aika	K2: Budjetissa pysyminen	K3: Asiakastyytyväisyys	K4: Ohjelmiston laatu	K5: Tavoitteiden saavuttaminen	K6: Tietoturva
Chatterjee & Mittal (2024) [13]	x	x	x	x		
Offerman et al. (2022) [29]	x	x	x	x	x	
Mishra & Otawi (2020) [2]	x		x	x		x
Mowad et al. (2022) [10]	x	x	x	x		x
Frefer et al. (2018) [28]	x	x	x	x	x	x
Azad (2024) [9]	x		x	x		
El Aouni et al. (2025) [3]	x	x		x	x	

ventaa suhteita nykyisten asiakkaiden kanssa. [10] Havaitut onnistumisen kriteerit täydentävät toisiaan ja ovat osittain toisistaan riippuvaisia, sillä yksittäisen kriteerin täyttyminen vaatii usein muiden kriteerien toteutumista.

3.2 DevOpsin ja CI/CD:n vaikutus ohjelmiston laatuun

DevOpsin vaikutusta ohjelmiston laatuun voidaan jäsentää tarkemmin ISO 25010 -standardin [30] määrittelemien ohjelmistotuotteen laadun ominaisuuksien kautta. Standardi jakaa laadun kahdeksaan pääominaisuuteen, jotka luetellaan Taulukossa 3.3. Näiden ominaisuuksien avulla tutkittiin, miten DevOpsilla on havaittu kirjallisuudessa olevan niihin vaikutusta.

Mainittakoon, että DevOpsin spesifeistä vaikutuksista ohjelmistotuotteiden laa-

tuun löytyy kirjallisuutta varsin vähän etenkin vapaasti saatavana, josta pohdintaa lisää luvussa 5. Mishran ja Otaiwin (2020) suorittamassa tutkimuksessa tutkittiin kirjallisuuskatsauksen avulla DevOpsin vaikutuksia ohjelmiston laatuun. Tutkimuksessa todettiin DevOpsin vaikuttavan positiivisesti ohjelmistoprojektin tuotteen laatuun melkein jokaisen ISO25010 -standardin laadun ominaisuuksien mukaisesti. [2] Saleemin et al. (2023) suorittamassa tutkimuksessa käytettiin ISO25010 -standardia määrittelemään kyselyssä ohjelmistoprojektin tuotteen laatua, mutta tässä tapauksessa DevOpsin vaikutusta ohjelmiston laatuun ei tarkasteltu spesifimmin. Tämä johtuu siitä, että yritykset eivät useimmiten salli ohjelmistotuotteensa laadun mittaamista. Laadun mittaamiseen vaaditaan suoraa pääsyä lähdekoodiin ja yritykset eivät halua, että heidän tuotteensa leimataan huonolaatuiseksi mahdollisten puutteiden vuoksi. Tutkimuksessa kuitenkin todettiin, että DevOpsin käyttöönotolla oli positiivinen vaikutus ohjelmiston laatuun. [31]

Taulukko 3.3: ISO 25010 ohjelmistotuotteen laadun ominaisuudet [30]

Nro	**Laadun ominaisuus**	**Kuvaus**
O1	Tarkoituksenmukaisuus	Kuinka hyvin tuote täyttää tarpeet määritellyissä käyttöolosuhteissa.
O2	Tehokkuus	Kuinka hyvin tuote käyttää resursseja ja suorittaa toimintonsa määritellyissä aikarajoissa.
O3	Yhteensopivuus	Tuotteen kyky vaihtaa tietoa muiden järjestelmien kanssa yhteisessä ympäristössä.
O4	Käytettävyys	Tuotteen kyky mahdollistaa vuorovaikutus käyttäjän kanssa eri käyttöyhteyksissä.
O5	Luotettavuus	Tuotteen kyky suorittaa toiminnot määritellyissä olosuhteissa tietyssä ajassa.
O6	Tietoturva	Tuotteen kyky suojata dataa ja estää haitallisten toimijoiden pääsy tietoihin.
O7	Ylläpidettävyys	Tehokkuuden ja toimivuuden aste, jolla tuotetta voidaan muokata, korjata ja sopeuttaa ympäristön ja vaatimusten muutoksiin.
O8	Joustavuus	Tuotteen kyky sopeutua vaatimusten, käyttöyhteyksien tai ympäristön muutoksiin.

Taulukosta 3.4 voidaan tulkita, että DevOpsin vaikutus ohjelmistoprojektin tuotteen laatuun on ollut pääosin positiivinen. Negatiivisia vaikutuksia raportoitiin vain yhdessä lähteessä, eikä yksikään lähde maininnut, ettei DevOpsilla olisi lainkaan vaikutusta ohjelmiston laatuun. Näin ollen voidaan todeta, että DevOpsin käyttöönotolla on kirjallisuuden perusteella lähes poikkeuksetta ainakin jonkinlainen vaikutus ohjelmiston laatuun, useimmiten parantava.

Taulukko 3.4: DevOpsin vaikutus ohjelmiston laatuun eri lähteissä

Lähde	**Positiivinen**	**Negatiivinen**	**Ei vaikutusta**
Saleem et al. (2023) [31]	x		
Chatterjee & Mittal (2024) [13]	x		
Mowad et al. (2022) [10]	x		
Offerman et al. (2022) [29]	x	x	
Mishra & Otawi (2020) [2]	x		
El Aouni et al. (2025) [3]	x		
Andrade et al. (2023) [32]	x		
Dileepkumar & Mathew (2023) [15]	x		

Kirjallisuudessa todettiin, että CAMS -viitekehystä noudattaen saadaan parannettua ohjelmiston laatua huomattavasti. [2], [13], [31], [33] Näistä ominaisuuksista suurin vaikutus ohjelmiston laatuun Saleem et al. (2023) suorittaman tutkimuksen mukaan on automaatiolla. [31] Dileepkumarin ja Mathew:n (2023) tutkimuksen mukaan automaatiolla, etenkin CI/CD:n näkökulmasta on huomattu olevan merkittäviä positiivisia vaikutuksia ohjelmiston laatuun. Tutkimuksessa tuli esille, että erään yrityksen ohjelmistosta löydettyjen virheiden määrä väheni peräti 60 % CI/CD:n käyttöönoton jälkeen. [15] Saleemin et al. tutkimuksessa kuitenkin todettiin, että jatkuvalla integraatiolla ei ollut huomattavaa vaikutusta ohjelmiston laatuun. Koska DevOps-käytännöt ovat kuitenkin yleisesti yhdistetty ohjelmiston laadun parantamiseen, ei tätä pidetä DevOpsin vaikuttavuutta kumoavana seikkana. [31]

3.3 DevOpsin ja CI/CD:n vaikutus ohjelmistoprojektin onnistumiseen

Luvussa 3.1 tarkasteltiin ohjelmistokehityksen ja DevOpsin kriittisiä menestystekijöitä ja onnistumisen määrittelemisen kriteerejä. Näiden tekijöiden ja kriteerien tarkastelu auttaa ymmärtämään DevOpsin vaikutuksia ohjelmistoprojektin onnistumiseen vertaamalla niiden kesken havaittua korrelaatiota käytetyissä lähteissä. Taulukosta 3.2 havaitaan, että jokainen valituista onnistumisen kriteereistä on kirjallisuudessa todettu täyttyvän DevOpsissa, joten voidaan tulkita, että DevOpsin käyttöönotolla on yhteys ohjelmistoprojektin onnistumisen kriteerien toteutumiseen, erityisesti kriteereihin Aika (K1), Budjetissa pysyminen (K2), Asiakastyytyväisyys (K3) ja Ohjelmiston laatu (K4). Chatterjeen ja Mittalin (2024) mukaan DevOpsissa CI/CD:llä on merkittävä vaikutus ohjelmistoprojektin onnistumiseen lisäämällä huomattavasti projektin tehokkuutta, nopeutta, ja laadukkaan ja luotettavan tuotteen luomista ensimmäisellä yrittämällä (Right First Time). Nämä ovat onnistumisessa keskeisiä ominaisuuksia ja vaikuttavat positiivisesti asiakastyytyväisyyteen, jota DevOps on myös havaittu parantavan. [13] Tarkastelun perusteella voidaan tulkita, että DevOps tukee onnistumisen kannalta keskeisiä kriteerejä ja on todistetusti vaikuttanut positiivisesti ohjelmistoprojektien onnistumiseen.

Offermanin et al. (2022) tutkimuksessa todettiin, että DevOpsin käyttöönotossa on kuitenkin havaittu myös negatiivisia vaikutuksia. Tutkimuksen mukaan DevOpsin käyttöönoton vuoksi osa tutkimukseen osallistuneista henkilöistä koki, että työnteon miellyttävyys on laskenut ja muutospohjaiset koodikatselmoinnit vaikuttavat negatiivisesti organisaation tehokkuuteen, asiakkaan tyytyväisyyteen, ohjelmiston laatuun, suorituskykyyn ja tavoitteiden saavuttamiseen. Tutkimukseen osallistuneet raportoivat, että julkaisujen ajankohta on DevOps-projektissa ennalta-arvaamaton. Tutkimuksessa kuitenkin mainittiin, että negatiiviset vaikutukset eivät ole huomat-

tavan suuria, tai johtuvat organisaation sisäistä haasteista, kuten DevOpsin käytänteiden tietämyksen puutteellisuus. [29]

Freferin et al. (2018) tutkimuksessa pohdittiin projektinhallintaa ja ohjelmistokehitystä myös kestävän kehityksen näkökulmasta. Kestävä kehitys tarkoittaa nykyisten tarpeiden tyydyttämistä viemättä tulevaisuuden sukupolvilta mahdollisuutta tyydyttää omat tarpeet samalla tavalla tai paremmin. Kestävä kehitys voidaan jakaa taloudelliseen, sosiaaliseen ja ympäristölliseen kestävyyteen. Näiden näkökulmien huomioon ottaminen on etenkin nykypäivänä olennaista myös ohjelmistokehityksessä. Kiinnostus kestävän kehityksen ja ohjelmistokehityksen yhdistämiseen onkin noussut ja kirjallisuudessa on todettu kiinnostuksen vain kasvavan tulevaisuudessa. [28] Kestävästä kehityksestä ohjelmistotuotannossa ja etenkin DevOpsissa on varsin vähän kirjallisuutta. On kuitenkin todistettu, että kestävän kehityksen periaatteita noudattava DevOps-projekti parantaa ohjelmistojen laatua ja toimitusprosessia. [14]

4 DevOpsin haasteet ja ennaltaehkäisevä ratkaiseminen

DevOpsin käyttöönotto organisaatioissa on todistetusti parantanut ohjelmistoprojektien onnistumista ja laatua, niin organisaatiollisista, kuin myös teknillisistä näkökulmista. DevOpsin käyttöönotto ja toimiva toteutus ei kuitenkaan ole ongelmatonta. DevOpsin käyttöönotto vaatii laajan muutoksen organisaatiossa, johon liittyy useita haasteita, jotka voivat estää tavoitteiden saavuttamisen, tai johtaa jopa projektin epäonnistumiseen. Tämän vuoksi on tärkeää ymmärtää, mitä haasteita tulee odottaa tämän muutoksen aikana ja kuinka näitä haasteita voidaan ratkaista ennaltaehkäisevästi. [34]

Tässä luvussa tutkitaan DevOpsin käyttöönoton mahdollisia haasteita, niin organisaation, kuin myös ohjelmiston tasolla ja näiden haasteiden mahdollisia ratkaisuja, tai ehkäisykeinoja. Kuvassa 4.1 on lueteltu DevOpsissa havaittujen haasteiden pääteemat. Luvussa vastataan tutkimuskysymykseen TK2: "Mitkä ovat DevOps-käyttöönoton yleisimmät haasteet ja miten niitä voidaan ratkaista, tai ennaltäehkäistä?"

4.1 Haasteet DevOpsissa

DevOpsin käyttöönotto ei ole aina välttämättä onnistunut. Riungu-Kalliosaaren et al. (2016) tutkimuksessa tuli esille, että näkyvimmät haasteet DevOpsissa liitty-

vät organisaation viestintämalleihin, joustamattomaan organisaatiokulttuuriin, toimialasta ja toimintaympäristöstä johtuviin rajoitteisiin, sekä DevOpsin merkityksen epäselvyyteen. [34]

Organisaation tasolla kulttuuri ja kommunikaatio ovat hyvin keskeisessä asemassa ja näiden riittämätön toteutus vaikuttaa merkittävästi kaikkiin DevOpsin vaiheisiin. Etenkin haasteeksi on todettu osastojen välinen kommunikaatio, jotka usein saattavat olla eri sijainneissa. [5] Kehitystiimin ja tuotantotiimin välisen kommunikaation on myös todettu kärsivän, jos se tapahtuu ainoastaan sähköisten systeemien kautta. Tämä voi johtaa hitaampaan reagontiin ongelmien ilmetessä.

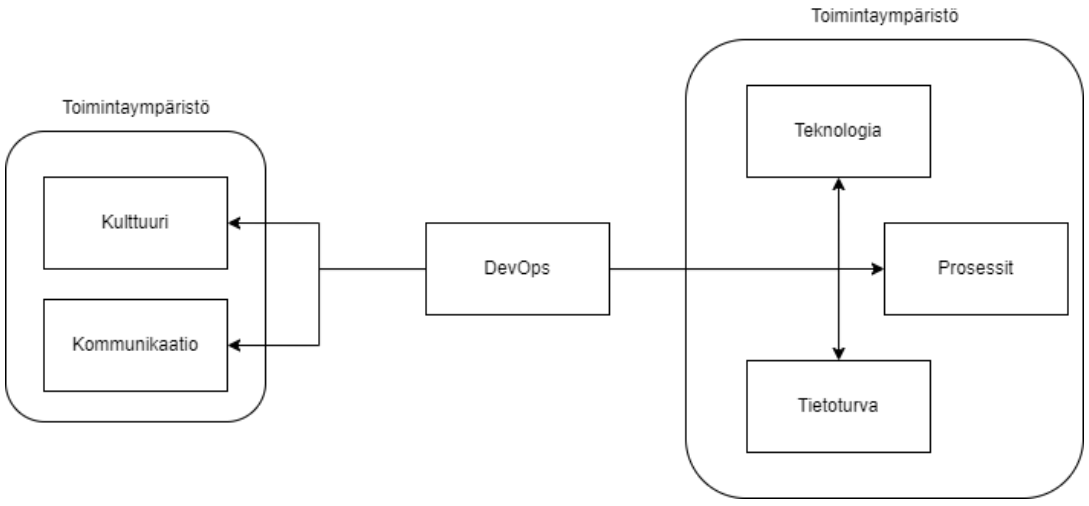
Kulttuurin muutos aikaisemmin käytetystä kehitysmenetelmästä DevOpsiin voi olla etenkin haastavaa, jos yrityksellä on syvät juuret ja vakiintuneet työtavat. Kehittäjät mahdollisesti joutuvat opettelemaan uusia tehtäviä ja muuttamaan tuttuja työskentelytapojaan. Vastuualueet muuttuvat ja yhdistyvät, joka voi olla haastavaa organisoida. Tämä tuo esiin johdon tärkeyden, jonka virheet ovatkin useammin esteenä menestykselle yksityisten työntekijöiden sijasta. [5], [34]

Toimintaympäristö voi myös olla esteenä DevOpsin käyttöönotolle ja DevOps ei välttämättä aina sovi jokaisiin tilanteisiin. Kehitysympäristöissä voi olla rajoitteita, jotka voivat tehdä automaattisesta testauksesta ja laadunvaarmistuksesta monimutkaista. Tekniset rajoitteet voivat perustua muun muassa lainsäädäntöön, tai yritysten välisiin sopimuksiin. [34] Monimuotoinen, tai erilaisista osista koostuva ympäristö, eli heterogeeninen ympäristö on todettu haasteeksi DevOpsin käyttöönotolle. [5], [34], [35] Yritysten täytyy pohtia oman tilanteensa mukaan, onko DevOpsin käyttöönotto kannattavaa.

Organisaation teknologian tulee myös olla yhteensopiva DevOpsin työskentelytapojen kanssa. Tässäkin näkökulmassa vaikuttaa toimintaympäristö. Tiukat tietoturvajärjestelyt voivat hankaloittaa, tai jopa estää DevOpsin tärkempien työkalujen tehokkaan käytön. Riungu-Kalliosaaren et al. (2016) mukaan yrityksen koolla on

myös mahdollisesti vaikutusta muutoksen vaikeustasoon. Pienemmillä yrityksillä on vähemmän muutettavaa ja työskentelytapojen juuret eivät ole vielä erityisen syviä, kun taas suuremmat yritykset vaativat enemmän aikaa reagoida DevOpsin tuomiin muutoksiin. [34] Koska DevOps ei ole yleisesti määritelty käsite, voi organisaatiossa haasteen aiheuttaa yhteisymmärryksen puute. Roolit ja niiden vastuut voivat olla epäselviä, joka voi hidastaa projektia, tai johtaa virheisiin. [36]

Luvussa 3 todettiin DevOpsin parantavan ohjelmiston ja organisaation tietoturvaa. Tietoturva on myös todettu yhdeksi haasteeksi DevOpsissa. Haasteeksi voi muodostua DevOpsin nopea kehitys- ja toimitustahti, joka voi olla ristiriidassa perinteisten turvallisuuskäytäntöjen kanssa. Aikaisemmin mainittu DevOpsin yhteisymmärryksen puute voi organisaatiossa johtaa epäselvyyteen siitä, kuka on vastuussa tietoturvan osista. Tämä voi johtaa valvonnan ja reagoinnin puutteisiin. [36] Rafin et al. (2020) suorittamassa tutkimuksessa tuli esille, että merkittävimmät tietoturvaongelmat DevOpsissa ovat automaatiotyökalujen ja tietoturvallisten kehitysstandardien puute. Nämä molemmat kuuluvat DevOpsin prosessien haasteisiin. [36]



Kuva 4.1: Havaitut haasteet DevOpsissa

4.2 Haasteiden ennaltaehkäisevä ratkaiseminen

Luvussa 4.1 todettiin haasteeksi kehitys- ja tuotantotiimin välinen kommunikaatio vain sähköisten systeemien kautta. Yksi DevOpsin tärkeimmistä tavoitteista on tuoda nämä kaksi tiimiä yhteen, joten puutteellisella kommunikaatiolla ei DevOpsin hyötyjä pystytä käyttämään hyväksi. Kasvokkainen viestintä voi olla tehokas ratkaisu tähän ongelmaan, mutta toimintaympäristö voi estää tämän, jos keskustelun osapuolet ovat kaukana toisistaan. Tässä tapauksessa etäkommunikaatiotaidot ovat keskeisessä asemassa. Parempi kommunikaatio ei kuitenkaan välttämättä tarkoita enempää kommunikaatiota, vaan liiallinen kokousten pitäminen voi vaikuttaa negatiivisesti tuottavuuteen. [5], [34]

Kulttuurin muutos on haastava prosessi organisaatiossa riippumatta muutosten määrästä, mutta hyvin sitoutuneella johdolla, yhteistyön ja luottamuksen rakentamisella voidaan poistaa muurit organisaation sisällä. Organisaation DevOps-käytännöt tulee myös kouluttaa asianmukaisesti koko tiimille, jotta kaikki ovat yhteisymmärryksessä rooleista ja niiden tehtävistä. Tällä tavalla epäselvyydet vähenevät ja projektin työskentely on tarvittavan nopeaa, tehokasta, laadukasta ja tietoturvallista. [5], [34]–[36]

Tietoturva on tehokkainta ottaa huomioon jo projektin alkuvaiheissa ja tätä varten on kehitetty DevSecOps (Development, Security, Operations), joka tarkoittaa turvallisuuskäytänteiden sisällyttämistä DevOpsin vaiheisiin. DevSecOps tarjoaa jatkuvasti turvallisuutta integroituna CI/CD-putkeen ja automoituna. CI/CD-käytänteiden ja tietoturvastandardien runsas osallistaminen ohjelmistoprojektiin vaikuttaa samanaikaisesti työn sujuvuuteen, laatuun ja tietoturvaan. [36]

5 Analyysi ja pohdintaa

Tämän tutkielman perusteella voidaan todeta, että DevOpsin vaikutuksia ohjelmiston laatuun on tutkittu varsin vähän systemaattisesti ja erityisesti laadun kokonaisvaltainen määrittely jää usein pinnalliseksi. Toisaalta, laatu käsitteenä on tapauskohtainen, monitulkintainen ja sen mittaaminen on haastavaa erityisesti silloin, kun pyritään vertailemaan organisaatioita ja niiden projektien välisiä eroja. Muun muassa ISO 25010 -standardi tarjoaa viitekehyksen ohjelmiston laadun mittaamiseen, mutta DevOpsin tapauksessa sen käyttö kirjallisuudessa on vielä harvinaista.

Lisäksi voidaan todeta, että useimmissa DevOpsiin liittyvissä tutkimuksissa keskitytään positiivisiin vaikutuksiin, kuten toimitusnopeuden paranemiseen, tai tiimien välisen yhteistyön tehostumiseen. Tutkimuksissa mainittiin harvoin negatiivisista vaikutuksista ja mainittaessa varsin lyhyesti. Tutkimuksessa todettiin, että yritykset eivät ole halukkaita tuomaan esiin negatiivisia puolia etenkin ohjelmistonsa laadussa, joten sen tarkempaa tutkimusta ei sallita. Tämän pohjalta voidaan myös pitää mahdollisena, että yritykset eivät halua jakaa julkisesti epäonnistuneita DevOps-siirtymiä tai muita negatiivisia vaikutuksia, kuten teknisen velan kertymistä, henkilöstön uupumista, tai tietoturvan puutteita. Negatiivisten puolien pois jättäminen tutkimuksista luo aukon tietämykseemme DevOpsin todellisista vaikutuksista organisaatioille, tai ohjelmistoille.

Yksi merkittävä havainto on, että yhteistä mittaristoa tai standardia DevOpsin, tai yleisesti ohjelmistoprojektin onnistumisen arviointiin ei ole vakiintunut. Tämä

vaikeuttaa vertailukelpoisten tutkimustulosten muodostamista ja rajoittaa DevOpsin ja ohjelmistokehityksen käytänteiden kehittämistä erilaisissa konteksteissa. Tutkimuksen perusteella olisi tärkeää kehittää mittareita, jotka ottavat huomioon niin tekniset, kuin myös ihmimmilliset ja organisaationalliset näkökulmat, kuten tiimien tyytyväisyys, yhteistyön laatu ja sidosryhmien kokemukset.

DevOpsin tutkimuksessa otettiin myös harvoin esiin kehitystiimin ja sidosryhmien tyytyväisyys. DevOps pyrkii tuomaan tiimin yhteen ja toimittamaan ohjelmistoja nopeammin, mutta todellinen arvonluonti laajemmasta näkökulmasta tapahtuu vasta, kun sekä loppukäyttäjät, osakkaat, että kehitystiimi kokevat olevansa tyytyväisiä ohjelmistoon ja sen täyttävän odotukset. Tyytyväisyys ei välttämättä ole myöskään vain ohjelmiston teknistä laatua, vaan kokonaisvaltainen kokemus kehitysprosessista ja sen lopputuloksesta.

DevOpsia ja nykypäivän ohjelmistotuotantoa pohtiessa tulee yllätyksettömästi mieleen myös vastuullisuus ja kestävä kehitys, jotka ovat olleet viime vuosina hyvin yleisesti keskustelussa mukana alasta riippumatta. Yliopisto-opinnoissa jopa kielikursseilla suuressa roolissa vastaan tullut aihepiiri herättää minussa monesti kiinnostusta ja herätti myös tälläkin kertaa.

Keskustelu vastuullisuudesta IT-ympäristössä ei ole enää pelkästään laitteiston, tai energiakulutuksen tasolla käytävää puhetta, vaan ohjelmistokehityksessä vastuullisuus voi näkyä esimerkiksi eettisenä koodina, läpinäkyvyytenä kehitysprosesseissa, turvallisuuden huomioimisen kaikissa vaiheissa tai saavutettavuuden parantamisena. DevOps on varsin joustava kehitysmalli, joten vastuullisuuden periaatteiden lisääminen DevOpsiin voisi olla hyvin mahdollista ja ne voisivat tuoda jopa liiketaloudellisia hyötyjä organisaatiolle nykyhetkessä ja tulevaisuudessa.

Kestävä kehitys yleensä rinnastetaan yleensä vain ympäristölliseen vastuullisuuteen, mutta ohjelmistokehityksen näkökulmasta tämä voi tarkoittaa myös pitkäikäisiä ja ylläpidettäviä järjestelmiä, jotka eivät aiheuta teknistä velkaa tulevaisuudessa.

DevOps voi toimia välineenä tämänkaltaisen kestävyyden rakentamisessa. Kun jokainen muutoksen vaihe on testattu, dokumentoitu ja läpinäkyvä, voidaan myös ohjelmiston elinkaarta hallita vastuullisesti. Vastuullisuus näkyy myös tiimin sisäisessä toiminnassa avoimena kommunikaationa, tasavertaisuutena ja jatkuvana oppimisena, jotka edistävät sosiaalista kestävyyttä tiimissä.

Mielenkiintoista oli myös erimielisyys johdon tärkeydestä ja organisaatiollisesta järjestyksestä ohjelmistokehityksessä. Hierarkisen järjestyksen rooli ja merkitys vaihtelivat näkökulmasta riippuen. Hierarkia nähtiin positiivisena tekijänä, joka voi selkeyttää rooleja erityisesti suurissa orgaanisaatioissa ja mahdollistaa tehokasta vastuunjakoa ja päätöksentekoa. Tällainen rakenne voi tuoda järjestelmällisyyttä ja vakautta nopeasti muuttuvassa kehitysprosessissa, mutta voi myös muodostua haitalliseksi DevOpsin keskeisille arvoille, kuten tiimien väliselle yhteistyölle, ketteryydelle ja jatkuvalle palautteelle. Liiallinen hierarkisuus voi hidastaa reaktiokykyä, estää nopeaa palautetta ja vähentää tiimien välistä yhteistyötä, jotka ovat täsmälleen niitä asioita, joiden parantamiseen DevOps on kehitetty.

Tämä osoittaa, että DevOpsin käytännöt eivät ole yksiselitteisiä, tai yleisesti organisaatioissa sovellettavia, vaan ne on mukautettava organisaation rakenteeseen, kulttuuriin ja tavoitteisiin sopivaksi. DevOpsin kannalta keskeistä ei ole niinkään hierarkian olemassaolo, vaan miten organisaation rakenne tukee jokapäiväistä toimintaa, viestintää ja yhteisiä tavoitteita. Joustava ja ympäristön mukaan muovautuva rakenne, jossa saadaan säilytettyä järjestys ja autonomia, näyttää usein toimivimmalta lähestymistavalta.

6 Yhteenveto

Tässä tutkielmassa tutkittiin kirjallisuuskatsauksen avulla DevOpsin ja Jatkuvan Integraation ja Jatkuvan Toimituksen käyttöä, haasteita ja niiden vaikutusta ohjelmistoprojektien onnistumiseen ja laatuun. Työssä tarkasteltiin DevOpsin periaatteita muun muassa CAMS-viitekehyksen kautta. Onnistumisen ja laadun mittaamiseen käytettiin kirjallisuudessa esiintyneitä ominaisuuksia, kriittisiä menestystekijöitä ja ISO 25010 ohjelmiston laadun standardia. Lisäksi tutkittiin kirjallisuudessa ilmenneitä DevOpsin käyttöönottoon liittyviä haasteita ja miten niitä voidaan ratkaista, tai ennaltaehkäistä.

Tutkielman perusteella DevOps ja etenkin CI/CD voivat parantaa ohjelmiston laatua muun muassa nopeamman palautteen, automaation ja tiimien välisen yhteistyön kautta (TK1). DevOpsilla oli myös positiivinen vaikutus ohjelmistoprojektin onnistumiseen löydettyjen kriteerien täyttymisen perusteella (TK1). Haasteita DevOpsissa esiintyi erityisesti kulttuurissa, kommunikaatiossa ja DevOpsin käytänteiden puutteellisessa tuntemuksessa (TK2). Näille haasteille löydettiin ennaltaehkäiseviä ratkaisuja, joista näkyvimmät olivat johdon sitoutuminen, yhteistyöllisen ja läheisen tiimityön vaaliminen ja tiimin kouluttaminen organisaation DevOps-käytänteisiin (TK2).

Tutkielmassa yksi keskeinen havainto on laadun ja onnistumisen määrittämisen monimutkaisuus ja laajuus. Ohjelmistoprojektin onnistuminen ja laatu ei rajoitu vain teknisiin ominaisuuksiin, vaan sitä täytyy pohtia monesta eri näkökulmasta.

Organisaation, sidosryhmien ja loppukäyttäjien tyytyväisyys tuotteeseen ja sen kehitysprosessiin määrääkin onnistumisen. Tulevaisuuden tutkimuksessa olisikin tärkeää tarkastella DevOpsin vaikutuksia ohjelmiston onnistumiseen ja laatuun kokonaisvaltaisemmin, mukaan lukien kestävän kehityksen periaatteet ja pitkän aikavälin vaikutukset. Nykyinen kirjallisuus keskittyy usein lyhyen aikavälin hyötyihin, kuten toimitusnopeuteen, mutta voiko DevOpsin käyttöönotto olla jollain osa-alueella haitallista tulevaisuuden kannalta? Voiko tulevaisuudessa kestävä kehitys olla osa DevOpsin onnistumisen mittareita? Tähän voisi kuulua muun muassa energiatehokkuus, teknisen velan hallinta ja henkilöstön hyvinvointi.

Kaiken kaikkiaan DevOpsin käyttö ohjelmistoprojekteissa on osoittautunut hyödylliseksi ja tehokkaaksi menetelmäksi, mutta sen tarkemmista vaikutuksista ohjelmistoprojektin onnistumiseen ja lopputuotteen laatuun tarvitaan edelleen lisää kontekstiin sidottua tutkimusta ja käytännön vertailuja.

Lähdeluettelo

[1] Shubham ja L. M. Saini, "The Impact of Devops on Software Quality", teoksessa *2024 Third International Conference on Smart Technologies and Systems for Next Generation Computing (ICSTSN)*, heinäkuu 2024, s. 1–6. DOI: 10.1109/ICSTSN61422.2024.10670849.

[2] A. Mishra ja Z. Otaiwi, "DevOps and software quality: A systematic mapping", *COMPUTER SCIENCE REVIEW*, vol. 38, marraskuu 2020, ISSN: 1574-0137. DOI: 10.1016/j.cosrev.2020.100308.

[3] F. El Aouni, K. Moumane, A. Idri, M. Najib ja S. U. Jan, "A systematic literature review on Agile, Cloud, and DevOps integration: Challenges, benefits", *INFORMATION AND SOFTWARE TECHNOLOGY*, vol. 177, tammikuu 2025, ISSN: 0950-5849. DOI: 10.1016/j.infsof.2024.107569.

[4] R. Amaro, R. Pereira ja M. M. da Silva, "Mapping DevOps capabilities to the software life cycle: A systematic literature review", *INFORMATION AND SOFTWARE TECHNOLOGY*, vol. 177, tammikuu 2025, ISSN: 0950-5849. DOI: 10.1016/j.infsof.2024.107583.

[5] L. Leite, C. Rocha, F. Kon, D. Milojicic ja P. Meirelles, "A Survey of DevOps Concepts and Challenges", *ACM Computing Surveys*, vol. 52, nro 6, s. 1–35, marraskuu 2020, arXiv:1909.05409 [cs], ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3359981.

[6] J. Humble ja D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley Professional.* 2010.

[7] L. E. Lwakatare, P. Kuvaja ja M. Oivo, "Relationship of DevOps to Agile, Lean and Continuous Deployment", teoksessa *Product-Focused Software Process Improvement*, P. Abrahamsson, A. Jedlitschka, A. Nguyen Duc, M. Felderer, S. Amasaki ja T. Mikkonen, toim., Cham: Springer International Publishing, 2016, s. 399–415, ISBN: 978-3-319-49094-6.

[8] Rishab, *The DevOps Guide*, 2025. url: `https://thedevops.guide/`.

[9] N. Azad, "Understanding DevOps critical success factors and organizational practices", teoksessa *2022 IEEE/ACM International Workshop on Software-Intensive Business (IWSiB)*, toukokuu 2022. DOI: `10.1145/3524614.3528627`.

[10] A. M. Mowad, H. Fawareh ja M. A. Hassan, "Effect of Using Continuous Integration (CI) and Continuous Delivery (CD) Deployment in DevOps to reduce the Gap between Developer and Operation", teoksessa *2022 International Arab Conference on Information Technology (ACIT)*, ISSN: 2831-4948, marraskuu 2022, s. 1–8. DOI: `10.1109/ACIT57182.2022.9994139`.

[11] R. Amaro, R. Pereira ja M. Mira da Silva, "DevOps Metrics and KPIs: A Multivocal Literature Review", *ACM Computing Surveys*, vol. 56, nro 9, s. 1–41, huhtikuu 2024, Publisher: Association for Computing Machinery. DOI: `10.1145/3652508`.

[12] M. Akbar, S. Mahmood, M. Shafiq, A. Alsanad, A. Alsanad ja A. Gumaei, "RETRACTED ARTICLE: Identification and prioritization of DevOps success factors using fuzzy-AHP approach", *Soft Computing*, vol. 27, s. 1907–1931, heinäkuu 2020. DOI: `10.1007/s00500-020-05150-w`.

[13] P. S. Chatterjee ja H. K. Mittal, ”Enhancing Operational Efficiency through the Integration of CI/CD and DevOps in Software Deployment”, teoksessa *2024 Sixth International Conference on Computational Intelligence and Communication Technologies (CCICT)*, huhtikuu 2024, s. 173–182. DOI: 10.1109/CCICT62777.2024.00038.

[14] M. Zohaib, A. Alsanad ja A. Abdullah Alhogail, ”Prioritizing DevOps Implementation Guidelines for Sustainable Software Projects”, *IEEE Access*, vol. 12, s. 71 109–71 130, 2024. DOI: 10.1109/ACCESS.2024.3402832.

[15] S. R. Dileepkumar ja J. Mathew, ”Transforming Software Development: Achieving Rapid Delivery, Quality, and Efficiency with Jenkins-Based CI/CD Pipelines”, teoksessa *2023 Annual International Conference on Emerging Research Areas: International Conference on Intelligent Systems (AICERA/ICIS)*, marraskuu 2023, s. 1–6. DOI: 10.1109/AICERA/ICIS59538.2023.10420251.

[16] Kharnagy, *DevOps Toolchain*, CC BY-SA 4.0, via Wikimedia Commons. Saatavilla: https://upload.wikimedia.org/wikipedia/commons/0/05/Devops-toolchain.svg, 2016.

[17] T. F. Dullmann, O. Kabierschke ja A. van Hoorn, ”StalkCD: A Model-Driven Framework for Interoperability and Analysis of CI/CD Pipelines”, teoksessa *Proceedings - 2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2021)*, IEEE, 2021, s. 214–223.

[18] N. Azad ja S. Hyrynsalmi, ”Multivocal Literature Review on DevOps Critical Success Factors”, teoksessa *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, sarja EASE ’24, New York, NY, USA: Association for Computing Machinery, kesäkuu 2024, s. 520–527, ISBN: 979-8-4007-1701-7. DOI: 10.1145/3661167.3661236.

[19] M. V. Belzen, J. Trienekens ja R. Kusters, "Critical success factors of continuous practices in a DevOps context", teoksessa *28th International Conference of Information Systems Development (ISD 2019)*, 2019.

[20] J. K. Leidecker ja A. V. Bruno, "Identifying and Using Critical Success Factors", *Long Range Planning*, vol. 17, nro 1, s. 23–32, 1984. DOI: 10.1016/0024-6301(84)90163-8.

[21] M. Shahin, "Architecting for DevOps and Continuous Deployment", teoksessa *Proceedings of the ASWEC 2015 24th Australasian Software Engineering Conference*, 2015, s. 147–148. DOI: 10.1145/2811681.2824996.

[22] M. Niazi, D. Wilson ja D. Zowghi, "Critical Success Factors for Software Process Improvement Implementation: An Empirical Study", *Software Process Improvement and Practice*, vol. 11, nro 2, s. 193–211, 2006. DOI: 10.1002/spip.261.

[23] J. A. V. M. K. Jayakody ja W. M. J. I. Wijayanayake, "Critical Success Factors for DevOps Adoption in Information Systems Development", *International Journal of Information Systems and Project Management*, vol. 11, nro 3, s. 60–82, 2023. DOI: 10.12821/ijispm110304.

[24] S. Rafi, M. A. Akbar, A. A. AlSanad, L. AlSuwaidan, H. A. Al-ALShaikh ja H. S. AlSagri, "Decision-Making Taxonomy of DevOps Success Factors Using Preference Ranking Organization Method of Enrichment Evaluation", *Mathematical Problems in Engineering*, vol. 2022, s. 1–15, 2022. DOI: 10.1155/2022/2600160.

[25] F. G. Aleu ja E. M. Van Aken, "Systematic Literature Review of Critical Success Factors for Continuous Improvement Projects", *International Journal of Lean Six Sigma*, vol. 7, nro 3, s. 214–232, 2016. DOI: 10.1108/IJLSS-06-2015-0025.

[26] 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. DOI: 10.1016/j.jss.2007.08.020.

[27] G. P. Sudhakar, "A model of critical success factors for software projects", *Journal of Enterprise Information Management*, vol. 25, nro 6, s. 537–558, 2012. DOI: 10.1108/17410391211272829.

[28] A. A. Frefer, M. Mahmoud, H. Haleema ja R. Almamlook, "Overview Success Criteria and Critical Success Factors in Project Management", *Industrial Engineering & Management*, vol. 7, nro 1, s. 1 000 244, 2018, ISSN: 2169-0316. DOI: 10.4172/2169-0316.1000244.

[29] T. Offerman, R. Blinde, C. J. Stettina ja J. Visser, "A Study of Adoption and Effects of DevOps Practices", teoksessa *2022 IEEE 28th International Conference on Engineering, Technology and Innovation (ICE/ITMC) & 31st International Association for Management of Technology, IAMOT Joint Conference*, sarja International ICE Conference on Engineering Technology and Innovation, ISSN: 2334-315X, IEEE; Univ Lorraine; IEEE Technol & Engn Management Soc; Int Assoc Management Technol; Res Network Innovat; ERPI; INP Ensgsi; EMRP; SIMPPE; Grand Est; Grp Actibac; ECTP; Inst Carnot; L OCTRI, 2022, ISBN: 978-1-6654-8817-4. DOI: 10.1109/ICE/ITMC-IAMOT55089.2022.10033313.

[30] International Organization for Standardization ja International Electrotechnical Commission, "ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models", ISO/IEC, Geneva, Switzerland, tekninen raportti 25010, 2011, Saatavilla: https://www.iso.org/standard/35733.html.

[31] M. Saleem, F. Saleem, F. Arif, A. B. Tariq, M. A. Riaz ja N. Abbas, "Evaluating Software Quality in DevOps Practices in Pakistan: A Survey Approach", teoksessa *2023 18th International Conference on Emerging Technologies (ICET)*, marraskuu 2023, s. 183–189. DOI: 10.1109/ICET59753.2023.10374716.

[32] Á. J. d. C. Andrade, E. Veloso ja G. Santos, "What We Know About Software Dependability in DevOps - A Tertiary Study", *Proceedings of the XXII Brazilian Symposium on Software Quality*, ACM Other conferences, s. 178–187, marraskuu 2023, ISSN: 9798400707865. DOI: 10.1145/3629479.3629502.

[33] P. Perera, R. Silva ja I. Perera, "Improve software quality through practicing DevOps", teoksessa *2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, 2017, s. 1–6. DOI: 10.1109/ICTER.2017.8257807.

[34] L. Riungu-Kalliosaari, S. Mäkinen, L. E. Lwakatare, J. Tiihonen ja T. Männistö, "DevOps Adoption Benefits and Challenges in Practice: A Case Study", teoksessa *Product-Focused Software Process Improvement (PROFES 2016)*, P. Abrahamsson, A. Jedlitschka, A. Nguyen-Duc, M. Felderer, S. Amasaki ja T. Mikkonen, toim., sarja Lecture Notes in Computer Science, vol. 10027, Springer, Cham, 2016, s. 590–597. DOI: 10.1007/978-3-319-49094-6_44.

[35] J. Faustino, D. Adriano, R. Amaro, R. Pereira ja M. M. da Silva, "DevOps benefits: A systematic literature review", *Software: Practice and Experience*, 2022. DOI: 10.1002/spe.3096.

[36] S. Rafi, W. Yu, M. A. Akbar, A. Alsanad ja A. Gumaei, "Prioritization Based Taxonomy of DevOps Security Challenges Using PROMETHEE", *IEEE Access*, vol. 8, s. 103 131–103 149, 2020. DOI: 10.1109/ACCESS.2020.2998819.