Tampereen yliopisto

Rasmus Kivinen

DEVOPS MODERNIN OHJELMISTOKEHITYKSEN TUKIPILARINA

Kandidaatintyö
Informaatioteknologian ja viestinnän tiedekunta
Huhtikuu 2022

TIIVISTELMÄ

Rasmus Kivinen: DevOps modernin ohjelmistokehityksen tukipilarina
Kandidaatintyö
Tampereen yliopisto
Tieto- ja sähkötekniikan kandidaattiohjelma
Huhtikuu 2022

Ketterä ohjelmistokehitys on mullistanut ohjelmistotuotannon menetelmiä jo useamman vuosikymmenen ajan. Sen tarpeita täyttämään ohjelmistojen toimituksen ja operoinnin suhteen on kehittynyt toimintamalli nimeltä DevOps. Nimi on yhdistelmä englanninkielisistä sanoista development (suom. kehitys) ja operations (suom. operointi).

DevOps on viimeisen kymmenen vuoden aikana nopeasti yleistynyt ohjelmistotuotannon kulttuurillinen ja teknillinen toimintamalli, jonka tavoitteena on mahdollistaa ohjelmakoodin nopeampi koonti, testaaminen ja julkaisu erilaisia periaatteita ja niitä tukevia työkaluja hyödyntäen. Tämän tutkielman tavoitteena on selvittää ja avata DevOpsin taustoja, periaatteita, käytäntöjä ja etuja. Tämä suoritetaan keräämällä kokoon siihen liittyvää kirjallisuutta mahdollisimman laajasti, jotta lukijan olisi mahdollista muodostaa tarkka kokonaiskuva ja -ymmärrys aiheen ympärille.

DevOpsin syntyperästä selviää muun muassa sen sidonnaisuus ketterään ohjelmistokehitykseen ja erityisesti siihen liittyvään nopeatahtiseen inkrementaaliseen kehitykseen. Yhdeksi DevOpsin syntyperistä katsotaan kuvapalvelu Flickrin toteuttama organisaatiollinen ja tekninen muutosprojekti, jonka tavoitteena oli mahdollistaa päivitysten jatkuva ja nopea julkaisu. Tämä on tärkeä osa DevOpsin historiaa, sillä koko konseptin ja DevOps-nimityksen katsotaan usein syntyneen kahden Flickrin työntekijän pitämästä konferenssiesityksestä vuonna 2009. Esityksessä he kuvailivat muutosprojektia ja sen eri osa-alueita.

Tutkimuksessa selviää myös DevOpsin ydinperiaatteita kuten ketteryys, automaatio ja yhteistyökulttuuri sekä se, miksi juuri nämä periaatteet ovat oleellinen osa DevOps-toimintamallia. Lisäksi käsitellään näitä periaatteita tukevia toimintatapoja ja työkaluja, kuten jatkuva integraatio, toimitus ja valvonta sekä palvelininfrastruktuurin määrittely koodina. DevOpsin tuomiksi hyödyiksi ilmenee muun muassa muutosten toimituksen nopeus, tuottavuuden paraneminen, kehitystiimien tyytyväisyys sekä korkeampi ohjelmistojen laatu.

Avainsanat: DevOps, ohjelmistotuotanto, ketterä ohjelmistokehitys, jatkuva julkaisu

SISÄLLYSLUETTELO

1. JOHDANTO

Ohjelmistoala on luonteeltaan jatkuvasti muuttuva ja kehittyvä. Nykyajan liikemaailmassa on luonnollista, että yritykset haluavat virtaviivaistaa prosessejaan tehokkuuden ja tuottavuuden lisäämiseksi, eikä ohjelmistoala ole tämän suhteen poikkeus. Ohjelmistotuotanto on laajalti siirtynyt perinteisestä vesiputousmallista ketterään ohjelmistokehitykseen, jonka perusajatuksena on ohjelmiston nopeatempoinen iteratiivinen parantaminen, eikä niinkään pitkien kehitysperiodien lopussa tapahtuvat suuret päivitykset. Tämän johdosta viimeisen kymmenen vuoden aikana suurta suosiota on saavuttanut DevOpsiksi kutsuttu ideologia, jonka tarkoituksena on mahdollistaa ohjelmakoodin nopeampi koonti, testaaminen ja julkaisu muun muassa automaatiota ja tietynlaista organisaatiollista kulttuuria korostamalla. DevOpsia pidetään kuitenkin usein pelkkänä muotisanana, joka voi oikeastaan tarkoittaa lähes mitä tahansa liittyen ketterän ohjelmistokehityksen toimitus- tai julkaisuvaiheeseen.

Tämän tutkielman tavoitteena on perehtyä DevOpsin taustoihin, periaatteisiin ja koettuihin hyötyihin sekä koota yhteen ja tutkia siihen liittyvää kirjallisuutta, sillä aiheesta tehty tutkimus on nykyhetkellä melko tuoretta ja monelta kannalta jopa ristiriitaista.

Tutkimuskysymys 1: Mistä DevOps on syntynyt ja mitä ongelmia se pyrkii korjaamaan?

Tutkimuskysymys 2: Mitkä ovat DevOpsin keskeisimmät periaatteet ja käytännöt?

Tutkimuskysymys 3: Mitä etuja DevOps-toimintamalli tuo verrattuna perinteisempiin ohjelmistojen kehittämisen ja operoinnin toimintatapoihin?

Luvussa 2 käsitellään tämän tutkielman taustoja ja sen tarkoituksena on vastata kysymyksiin DevOpsin alkuperästä ja siitä tehdystä tutkimuksesta. Lisäksi luvussa tarkastellaan itse DevOps-käsitteen määritelmää yleisesti sekä tämän tutkielman kontekstissa. Luku 3 listaa ja määrittelee tutkimuksissa havaittuja DevOpsin yleisiä periaatteita ja työkaluja mahdollisimman laajasti sekä analysoi niiden toimintatapoja. Luvussa 4 kootaan yhteen kysely- ja haastattelututkimuksissa selvinneitä DevOpsin koettuja hyötyjä ohjelmistoyritysten kontekstissa.

2. TAUSTAA

Ohjelmiston vieminen tuotantoon joko uuden julkaisun tai ohjelmistopäivityksen muodossa on perinteisesti ollut suuri ja riskialtis tapahtuma, jossa vastuu siirtyy ohjelmistoa kehittäneeltä tiimiltä erilliselle operointitiimille, jonka tehtävänä on ohjelmakoodin koonti, toimitus, asennus ja ylläpito. Tämä aiheuttaa prosessissa epäjatkuvuuskohdan, jonka takia edistys voi pahimmillaan pysähtyä pitkäksikin aikaa. Tällaisessa tilanteessa palautteen saaminen tehdystä työstä ei ole mahdollista, jolloin ketterälle ohjelmistokehitykselle tärkeä inkrementaalinen kehitys voi keskeytyä. Täten on myös selvää, että ohjelmiston päivitysten tahti on melko hidas, sillä jokainen päivitys vaatii merkittäviä manuaalisia toimenpiteitä operointitiimiltä.

2.1 Alkuperä

DevOps konseptina alkoi yleistymään hieman ennen 2010-luvun alkua. Yksi huomattavimmista liikkeen edistäjistä oli John Allspawn ja Paul Hammondin pitämä esitys ”10+ Deploys Per Day: Dev and Ops Cooperation at Flickr” Velocity 2009-konferenssissa. Allspaw ja Hammond esittelivät siinä niin kulttuurimuutoksia Flickrillä kuin myös tapoja, joilla pitkälle automatisoitu ohjelmakoodin koonti, testaus ja tuotantoonvienti oli toteutettu. [6] [11] Pian tämän jälkeen puheesta inspiroituneena belgialainen Patrick Debois järjesti ensimmäistä kertaa Devopsdays-konferenssin, joka on sittemmin levinnyt joka puolelle maailmaa ja järjestetään useamman kerran vuodessa.

Vuonna 2013 yhdysvaltalainen Puppet Incorporated julkaisi ensimmäisen State of Devops -raportin. Jo silloin DevOpsin hyödyiksi todettiin muun muassa huomattavasti nopeampi tuotantoonvienti sekä pienempi virhealttius. [4]

2.2 Tutkimuksen nykytilanne

DevOpsiin liittyvä tutkimus on yleistynyt vasta lähivuosina, eikä esimerkiksi yhtenevää määritelmää itse käsitteelle ole vielä saatu [7]. Kuitenkin tilanne on muutaman viime vuosien aikana parantunut hieman: esimerkiksi IEEE Xplore-tietokannasta hakusanalla DevOps ennen vuotta 2018 julkaistuja artikkeleita löytyy 200 kappaletta, kun taas vuodesta 2018 kirjoitushetkeen asti artikkeleita löytyy yli 400 kappaletta.

Aihepiirin ympärillä empiirisen tutkimuksen tekeminen on hyvin haastavaa käsitteen löyhän määrittelyn sekä DevOpsin monimuotoisuuden takia, eikä tämän tutkielman tiedonhakuprosessissa löytynyt yhtään sopivaa empiiristä tutkimusta. Tämän johdosta tärkeimmät lähteet muodostuvat kysely- ja haastattelututkimuksista sekä muista kirjallisuuskatsauksista.

2.3 Määritelmä

DevOpsin tarkan määritelmän puute on yleisesti tunnistettu niin yritys- kuin myös akateemisessa maailmassa [12]. Yksi ensimmäisistä ehdotuksista yhtenäistää DevOpsin määritelmä tuli Andrew Dyckin ja kollegojen [1] paperista vuonna 2015. Sen jälkeen on julkaistu useita tutkimuksia, joiden tarkoitus on määritellä käsite sekä yhtenäistää DevOps-periaatteita. Tässä paperissa lähteenä tämänlaisia ovat muun muassa Gallin [3], Lwakataren [6] ja Stahlin [12] ja kollegojen tutkimukset. Näistä papereista selviää muun muassa se, että osa ohjelmistoalan ammattilaisista pitää DevOpsia puhtaasti kulttuurillisena ilmiönä ja osa ainoastaan joukkona teknisiä menetelmiä tai työkaluja.

On kuitenkin määriteltävä käsite tämän tutkimuksen kontekstissa, jotta aihepiiri voidaan rajata sopivasti. Pidettäköön siis määritelmänä Gallia sekä Macarthya ja kollegoja [3] [7] mukaillen: DevOps koostuu organisaatiollisesta kulttuurista, jossa kehitys- ja operointitiimit tekevät yhteistyötä, sekä ketterän ohjelmistokehityksen periaatteista ja automaatiotyökaluista, joiden avulla voidaan edistää ohjelmistotuotannossa nopeampaa koontia, testausta ja julkaisua sekä tukea ketterää ohjelmistokehitystä.

Myös aikaisemmin mainittu Dyckin et al. [1] ehdotus määritelmälle oli melko lähellä tätä määritelmää. Tämän tutkielman ehdotuksen ja Dyckin et al. paperissa mainitun määritelmän erona on kuitenkin painotus ketterään ohjelmistokehitykseen ja automaatioon.

3. DEVOPSIN PERIAATTEET JA TYÖKALUT

Kuten luvussa 1 mainittiin, DevOpsin voidaan katsoa koostuvan organisaatiollisesta kulttuurista ja sitä tukevista työkaluista. Tämän luvun tavoitteena on listata kirjallisuuden osoittamia DevOpsin periaatteita sekä niitä tukemaan tarkoitettuja työkaluja mahdollisimman laajasti, jotta niistä voitaisiin luoda mahdollisimman kattava kokonaiskuva.

3.1 DevOpsin periaatteet

DevOps-ideologia sisältää tiettyjä yleisesti tunnettuja perusperiaatteita. Näille ei kuitenkaan ole yhtenäistä määritelmää, joten tässä kappaleessa luetellaan kaikki periaatteet joita DevOpsin katsotaan yleisimmin koostuvan. Kaikkien alla mainittujen periaatteiden noudattaminen ei ole vaatimus DevOpsin toteutumiselle. Listaus on tehty mukaillen Lwakataren [6], Gallin [3] ja de Françan [2] papereita.

3.1.1 Yhteistyökulttuuri

DevOpsin kulttuurillisen puolen perusajatus on poikkifunktionaalinen tiimien yhteen tuominen ja yhteistyön sujuvuuden varmistaminen. [5] [6] Tämän toteutuminen vaatii usein organisaatiomuutosta, jossa esimerkiksi kehittäjätiimin ja operointitiimin roolit ja vastuut sekoittuvat keskenään tai tiimien väliin tuodaan erillinen DevOps-tiimi, jonka tarkoitus on toimia yhdistävänä tekijänä [9].

Yhteistyön onnistumisen perusedellytyksenä on sujuva kommunikaatio tiimien jäsenten välillä: parhaana välineenä tähän pidetään yleisesti kasvotusten käytäviä keskusteluja tai vaihtoehtoisesti viestintäsovelluksia kuten Slackia, mutta tikettijärjestelmät kuten Jira eivät täytä tätä vaatimusta. [3]

3.1.2 Automaatio

Kaiken mahdollisen automatisointia pidetään toimivan DevOpsin vaatimuksena, ja joissain tapauksissa DevOps-käsite itsessään voidaan jopa ymmärtää pelkästään joukkona automaatiotyökaluja, jotka tukevat ohjelmiston jatkuvaa julkaisua [7]. Automaatiolla tarkoitetaan DevOpsiin liittyen muun muassa ohjelmakoodin koonnin, testauksen, julkaisun

ja valvonnan automatisointia.

Korkealla automaation tasolla pyritään vähentämään tarvittavia manuaalisia toimenpiteitä ohjelmiston integraation ja toimituksen yhteydessä. Näitä prosesseja sekä niissä käytettäviä työkaluja käsitellään tarkemmin luvussa 3.2.

3.1.3 Nopea palaute

Palautteensaannin virtaviivaistaminen mahdollistaa nopean reagoinin esimerkiksi tilanteissa, joissa vikoja sisältävää ohjelmakoodia on yritetty siirtää koodikantaan. Kappaleessa 3.1.2 mainittu automaattitestaus pystyy oikein toteutettuna hyvin pitkälti estämään tällaiset tilanteet, mikä varmistaa esimerkiksi sen, että koodikannan päähaara on aina valmis tuotantoonvientiä varten.

Nopea palautteen saaminen ja mahdolliset palautteeseen perustuvat suunnittelumuutokset ovat oleellinen osa ketterää ohjelmistokehitystä. DevOps-toimintamalli tukee tätä automaation lisäksi toimitusten tiheällä tahdilla. Tämän johdosta asiakkaat voivat tarjota tarkkaa ja ripeää palautetta, usein yhdestä pienestä muutoksesta kerrallaan. [13]

3.1.4 Jatkuva kehitys

Nopean palautteen ja toimivan laadunvalvonnan johdosta ohjelmiston jatkuva parantaminen ja kehitys ovat tärkeä osa DevOps-periaatteita. Ohjelmiston laadun jatkuvaa inkrementaalista parantamista pidetäänkin monessa tapauksessa tärkeämpänä kuin kehityksen nopeaa tahtia [3].

Kuten luvussa 2 mainitaan, jatkuva ja inkrementaalinen kehitys on yksi niistä ketterän kehityksen periaatteista, joista koko DevOps-toimintamalli on syntynyt. Valtaosa seuraavassa luvussa käsitellyistä työkaluista ja toimintamalleista onkin tarkoitettu tukemaan jatkuvaa kehitystä tavalla tai toisella.

3.2 DevOps-työkalut

Tämän osion tavoitteena on listata oleellisimpia työkaluja tai toimintamalleja, joista DevOpsin tekninen puoli koostuu. Lisäksi työkalujen toimintaa analysoidaan korkealla tasolla.

3.2.1 Jatkuva integraatio

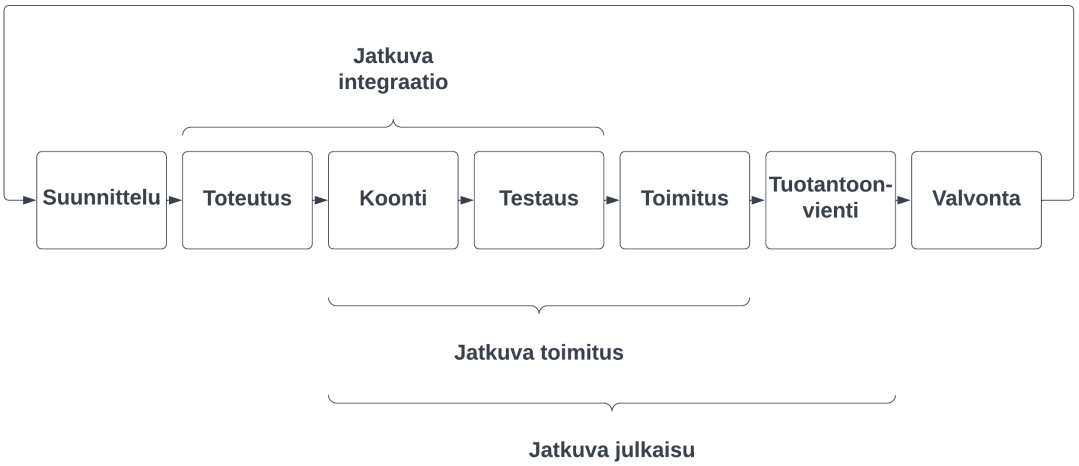
Stahlin ja kollegojen [12] mukaan jatkuvalla integraatiolla tarkoitetaan tiheään tahtiin tapahtuvaa ja pitkälle automatisoitua ohjelmakoodin integraatiota kehittäjiltä keskinäiseen koodikantaan. Tämän prosessin keskeinen ongelma on uuden ohjelmakoodin yhteensopivuuden ja toimivuuden varmistaminen [8].

Jatkuvan integraation suosituimpia ohjelmistoja ovat muun muassa Jenkins, TeamCity ja Travis CI. Niiden toimintaperiaate on pitkälti sama: kun koodikantaan tehdään muutos, työkalu tekee tarvittaessa lähdekoodin koonnin ennalta määrätyillä metodeilla. Tämän jälkeen työkalu suorittaa ohjelmalle esimerkiksi yksikkö-, integraatio- ja järjestelmätestit. Mikäli koottu ohjelma läpäisee nämä testit, pidetään viimeisintä koontiversiota hyväksyttynä. Mikäli ohjelma ei läpäise kaikkia kriittiseksi määritettyjä testejä, pidetään koontiversiota rikkinäisenä. Tällöin viimeisimmän muutoksen tehneen kehittäjän tai jopa koko tiimin tärkeimpänä prioriteettina tulisi olla havaittujen virheiden korjaus, jotta ajantasainen koontiversio olisi toimiva. [8]

3.2.2 Jatkuva toimitus

Jatkuva integraatio ja jatkuva toimitus luokitellaan usein yhteisen termin CI/CD (eng. Continuous Integration/Continuous Delivery) alle, sillä niiden toiminta on hyvin pitkälti sidottu toisiinsa: jatkuvan toimituksen tarkoitus on siirtää hyväksytty koontiversio koodikannasta joko testaus- tai tuotantoympäristöön, jossa koontiversiolle suoritetaan useimmiten hyväksyntätestaus [7]. Tämän jälkeen koontiversio voidaan siirtää tuotantoon, useimmiten yhdellä komennolla. Mikäli tämäkin tapahtuu automaattisesti, kutsutaan koko prosessia nimellä jatkuva julkaisu (eng. Continuous Deployment) [12].

Jatkuvan integraation ja toimituksen perusajatuksena on se, että ohjelmakoodin viimeisimmän version koontikäännös on mahdollista viedä tuotantoon millä hetkellä tahansa ja se, että tuotantoonvienti onnistuu mahdollisimman vähällä manuaalisella työllä. Tämä mahdollistaa versiopäivitysten tiheän tahdin loppukäyttäjälle, mikä tukee jatkuvan kehityksen periaatetta. Tämä on syynä luvussa 3.2.1 mainitulle virhekorjausten korkealle prioriteetille.



Kuva 1. *Jatkuvan integraation, toimituksen ja julkaisun sijainnit ja automatisoidut toiminnot ohjelmistokehityksen elinkaaressa.*

3.2.3 Infrastruktuuri koodina

Viime vuosina pilvialustat kuten Google Cloud Platform ja Microsoft Azure ovat saavuttanut suurta suosiota fyysisten, palveluntarjoajan tiloissa sijaitsevien palvelimien korvikkeena. Näillä alustoilla toimivien palvelimien infrastruktuurin määrittelyyn on pyritty kehitettämään prosessi, jossa määrittely tehdään koneluettavassa muodossa, eikä esimerkiksi manuaalisesti interaktiivisen ohjauspaneelin kautta. Tällöin pilvialustan konfigurointiin on olemassa yksi ennalta määritelty menettely.

Infrastruktuuri koodina (eng. Infrastructure as Code, IaC) muodostuu tiedostoista, jotka sisältävät halutun konfiguraation tiedot tai komennot, joilla pilvialusta saadaan konfiguroitua. Nämä tiedostot säilytetään usein versionhallinnassa ja niihin voidaan jopa soveltaa useita ohjelmistotuotannon periaatteita ja menetelmiä, sillä ne ovat itsessään enemmän tai vähemmän ohjelmakoodia. [14]

3.2.4 Valvonta

Kun tavoitteena on koontiversioiden nopea tuotantoonvienti, korostuu tehokkaan ja kattavan valvonnan merkitys entisestään tuotantoonviennin jälkeen. Lokitiedostojen tallentaminen, metriikkojen seuraaminen ja poikkeustilanteissa hälytysten tekeminen ovat kaikki elintärkeitä ohjelmiston toimivuuden ja jatkuvuuden kannalta. [8]

Jatkuva automatisoitu reaaliaikainen valvonta automatisoidun julkaisuputken kanssa mahdollistaa nopean reagoinnin tilanteissa, joissa ohjelmistossa ilmenee vikatilanne [3]. Lwakataren ja kollegojen [6] tutkimuksen kaikissa tapauksissa kehittäjät olivat itse vastuussa tuotteensa valvonnasta, jonka automatisointiin käytettiin työkaluja kuten New Relic, Kinesis ja Graylog.

4. DEVOPSIN HYÖDYT

DevOpsin koettujen hyötyjen löytämiseksi on toteutettu useita kysely- ja haastattelututkimuksia, joiden tuloksia kootaan yhteen ja analysoidaan tässä luvussa.

4.1 Muutosten toimituksen nopeus

Lwakataren ja kollegojen mukaan yleisin havaittu hyöty DevOps-toimintamallin käyttöönoton jälkeen on ohjelmistoon tehtyjen muutosten toimituksen nopeus. Yhdessä tapauksessa aiemmin kuukausia kestänyt toimitus saattoi kestää vain joitakin päiviä. Nopea toimitus koettiin erityisen hyödylliseksi kiireellisiä muutoksia kuten virhekorjauksia tehdessä. [6]

Vaikka nopeampien muutosten mahdollisuus on monessa käsitellyistä tutkimuksista tärkeä itseisarvo, on sillä myös implikaatioita moniin muihin koettuihin hyötyihin. Riungu-Kalliosaaren ja kollegojen [10] mukaan DevOps-toimintamalli mahdollistaa jatkuvan uusien asioiden kokeilun, minkä johdosta kyetään määrittämään uusien ideoiden arvoa asiakkaille tehokkaasti.

4.2 Tuottavuuden ja laadun paraneminen

DevOps-toimintamallin on havaittu lisäävän ohjelmistotiimien tuottavuutta usealla eri tavalla [2]. Meyer [8] toteaa artikkelissaan, että hitaiden testiohjelmien odottaminen laskee kehittäjien tuottavuutta. Lwakataren ja kollegojen [6] tutkimuksessa havaittiinkin manuaalisten askelten vähentämisen lisäävän tuottavuutta toimitusprosessissa.

Jokaisessa Lwakataren ja kollegojen [6] tutkimuksessa käsitellyistä tapauksista DevOps-toimintamallin käyttöönotto oli parantanut havaittua ohjelmiston laatua muun muassa testauksen ja tuotantoonviennin automatisoinnin johdosta . Myös Riungu-Kalliosaaren et al. [10] mukaan ohjelmistojen laatu oli korkeampi DevOpsia hyödyntävissä yrityksissä, muun muassa tehokkaampien palautesyklien ansiosta.

Näitä tuloksia tulkittaessa ja erityisesti abstrakteihin metriikoihin kuten ohjelmiston laatuun liittyen on huomattava, että tulokset perustuvat tutkimuksessa haastateltujen henkilöiden omiin havaintoihin ja kokemuksiin. Ohjelmiston laadun katsotaan usein olevan osittain sidonnainen siinä esiintyvien virheiden määrään, joten pienemmän virhealttiuden voidaan tulkita olevan tärkeä tekijä tässä aspektissa.

4.3 Vikojen nopeampi löytäminen

Yhtenä DevOps-toimintamallin suurimmista ja yleisimmin ilmenevistä vahvuuksista pidetään pienempää virhealttiutta [6]. Suuri syy tälle on se, että mahdolliset viat ohjelmakoodissa löydetään tyypillisesti aikaisessa vaiheessa muutoksen elinkaarta. De Françan ja kollegojen [2] mukaan tämän mahdollistavat DevOps-menetelmät kuten lähdekoodin jakaminen, jatkuva integraatio ja automatisoidut tuotantoonviennit.

Vikojen aikainen löytäminen sekä muutosten tiheys ja pieni koko johtavat useissa tapauksissa korkeaan varmuuden tunteeseen ja vähentyneeseen stressiin päivityksiä siirrettäessä tuotantoon [6]. Tämä ilmiö edistää muun muassa luvussa 4.4 käsiteltävää kehitystiimien tyytyväisyyttä sekä luvussa 4.2 käsiteltävää korkeampaa tuottavuutta. Jälleen ilmenee, että DevOpsin koetut hyödyt ovat usein vahvasti sidottuja toisiinsa eikä niinkään yksittäisiä etuja.

4.4 Kommunikaation paraneminen ja kehitystiimien tyytyväisyys

Luvussa 3 mainittu yhteistyökulttuuri pyrkii vähentämään kehitys- ja operointitiimien välillä tapahtuvaa niin kutsuttua siiloutumista yhdistämällä tiimejä niin funktionaalisesti kuin myös fyysisesti [2]. Tämä ilmenee muun muassa kokemuksien ja tiedon jakamisen lisääntymisenä, lisääntyneenä luottamuksena sekä toimivampana yhteistyönä [10] [6].

DevOps-toimintamallin hyödyntäminen on vähentänyt kehitys- ja operointitiimien jäsenten kokemaa stressiä ja lisännyt koettua tyytyväisyyttä Riungu-Kalliosaaren [10] sekä Lwakataren [6] ja kollegojen tutkimuksissa. Tämä johtuu muun muassa edellisessä kappaleessa mainitusta luottamuksen lisääntymisestä, paremmasta kommunikaatiosta sekä luvussa 4.3 mainitusta itsevarmuuden tunteesta muutoksia julkaistaessa.

5. POHDINTA JA YHTEENVETO

Tässä tutkielmassa selvitettiin, koottiin yhteen ja analysoitiin kirjallisuuskatsauksen muodossa DevOps-toimintamallin historiaa, tärkeimpiä piirteitä ja sen tuomia hyötyjä. Kirjoitushetkeen mennessä tehty tutkimus aihepiirin ympärillä on pitkälti hajanaista, joten tämän tutkielman tavoitteena olikin myös tarjota suhteellisen helposti ymmärrettävä yleiskuvaus ja määrittely itse DevOps-käsitteestä.

Tutkielmassa löydettiin tärkeimpiä yleisesti tutkimuksissa esille tulleita DevOpsin periaatteita: yhteistyökulttuuri, automatisointi, nopea palautteensaanti ja jatkuva kehitys. Tärkeimmiksi DevOpsin toiminnoista nousi esille jatkuva integraatio ja toimitus, infrastruktuuri koodina ja automatisoitu valvonta.

DevOpsin ohjelmistoyrityksille ja -tiimeille tuomien hyötyjen on ilmennyt olevan muutosten tekemisen ja toimituksen nopeus, parempi tuottavuus sekä ohjelmiston laatu, kehitystiimien tyytyväisyys sekä nopeampi vikojen paikantaminen. Lisäksi kommunikaation on huomattu parantuneen varsinkin poikkifunktionaalisten tiimien tapauksessa.

Tämän tutkielman aihepiirin ulkopuolelle jäivät esimerkiksi DevOpsin käyttöönoton haasteet ja mahdolliset haittapuolet. Monet käytetyistä lähteistä käsittelivät varsinkin käyttöönoton haasteita. Mikäli tämä tutkielma on lukijan ensikosketuksia DevOps-käsitteeseen, kannattaa nämäkin puolet varmasti huomioida.

DevOps on mullistanut monia toimintatapoja ohjelmistoalalla, mutta sen tultua valtavirran noudattamaksi toimintamalliksi itse DevOps ei ole kokenut suuria perusteellisia muutoksia. DevOpsin tulevaisuus varsinkin automaation suhteen on kuitenkin varsin mielenkiintoinen esimerkiksi tekoälyn käytön lisääntymisen johdosta ohjelmistokehityksen ja operoinnin kontekstissa.

LÄHTEET

[1] Andrej Dyck, Ralf Penners ja Horst Lichter. ”Towards Definitions for Release Engineering and DevOps” (2015). DOI: 10.1109/RELENG.2015.10.

[2] Breno B. Nicolau de França, Helvio Jeronimo ja Guilherme Horta Travassos. ”Characterizing DevOps by Hearing Multiple Voices”. Teoksessa: *Proceedings of the 30th Brazilian Symposium on Software Engineering*. SBES ’16. Maringá, Brazil: Association for Computing Machinery, 2016, s. 53–62. ISBN: 9781450342018. DOI: 10.1145/2973839.2973845. URL: https://doi-org.libproxy.tuni.fi/10.1145/2973839.2973845.

[3] Michael Gall ja Federico Pigni. ”Taking DevOps mainstream: a critical review and conceptual framework”. *European Journal of Information Systems* (2021). DOI: 10.1080/0960085X.2021.1997100.

[4] Puppet Inc. *2013 State of DevOps Report*. URL: https://puppet.com/resources/report/2013-state-devops-report (viitattu 20. 04. 2022).

[5] Welder Pinheiro Luz, Gustavo Pinto ja Rodrigo Bonifácio. ”Adopting DevOps in the real world: A theory, a model, and a case study”. *Journal of Systems and Software* 157 (2019), s. 110384. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2019.07.083. URL: https://www.sciencedirect.com/science/article/pii/S0164121219301517.

[6] Lucy Ellen Lwakatare, Terhi Kilamo, Teemu Karvonen, Tanja Sauvola, Ville Heikkilä, Juha Itkonen, Pasi Kuvaja, Tommi Mikkonen, Markku Oivo ja Casper Lassenius. ”DevOps in practice: A multiple case study of five companies”. *Information and Software Technology* 114 (2019), s. 217–230. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2019.06.010. URL: https://www.sciencedirect.com/science/article/pii/S0950584917302793.

[7] Ruth W. Macarthy ja Julian M. Bass. ”An Empirical Taxonomy of DevOps in Practice”. Teoksessa: *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 2020, s. 221–228. DOI: 10.1109/SEAA51224.2020.00046.

[8] Mathias Meyer. ”Continuous Integration and Its Tools”. eng. *IEEE software* 31.3 (2014), s. 14–16. ISSN: 0740-7459.

[9] Kristian Nybom, Jens Smeds ja Ivan Porres. ”On the Impact of Mixing Responsibilities Between Devs and Ops”. Teoksessa: *Agile Processes, in Software Engineering, and Extreme Programming*. Toim. Helen Sharp ja Tracy Hall. Cham: Springer International Publishing, 2016, s. 131–143. ISBN: 978-3-319-33515-5.

[10] Leah Riungu-Kalliosaari, Simo Mäkinen, Lucy Ellen Lwakatare, Juha Tiihonen ja Tomi Männistö. ”DevOps Adoption Benefits and Challenges in Practice: A Case Study”. eng. Teoksessa: *Product-Focused Software Process Improvement*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, s. 590–597. ISBN: 9783319490939.

[11] Sanjeev Sharma. *The DevOps Adoption Playbook: A Guide to Adopting DevOps in a Multi-Speed IT Enterprise*. eng. Somerset: John Wiley Sons, Incorporated, 2017. ISBN: 1119308747.

[12] Daniel Stahl, Torvald Martensson ja Jan Bosch. ”Continuous practices and devops: beyond the buzz, what does it all mean?” Teoksessa: *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 2017, s. 440–448. DOI: 10.1109/SEAA.2017.8114695.

[13] Manish Virmani. ”Understanding DevOps & bridging the gap from continuous integration to continuous delivery”. Teoksessa: *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)*. 2015, s. 78–82. DOI: 10.1109/INTECH.2015.7173368.

[14] Michael Wittig. *Amazon Web Services in action*. eng. Second edition. Shelter Island, New York: Manning, 2019. ISBN: 1-63835-719-6.