

Jaakko Utriainen

DEVOPS-TOIMINTAMALLI TEKOÄLYN AIKAKAUDELLA

Kandidaatintutkielma
Informaatioteknologian ja viestinnän tiedekunta
Marraskuu 2025

TIIVISTELMÄ

Jaakko Utriainen: DevOps-toimintamalli tekoälyn aikakaudella
Kandidaatintutkielma
Tampereen yliopisto
Tietotekniikka, Tieto- ja sähkötekniikan koulutus
Marraskuu 2025

Tekoälyn käyttö on yleistynyt viime vuosina valtavasti. Tekoälyn jatkuva kehittyminen tuo mukanaan monia muutoksia, joille IT-ala on erityisen herkkä. Tässä kirjallisuuskatsauksessa selvitetään odotuksia ja ennusteita viimeisen vuosikymmenen aikana ohjelmistokehitysalalla suosioon nousseen DevOps-toimintamallin käytöstä ja kehityksestä tekoälyn yleistyessä. Lisäksi tutkimuksessa eritellään tekoälyn tuomia mahdollisuuksia ja haasteita DevOps-mallin eri vaiheisiin. Tutkielman aineisto koostuu hyvin tuoreista ja vertaisarvioituista tieteellisistä julkaisuista. Työssä kerrotaan aineistoissa esiin nostettuja mahdollisuuksia korkealla tasolla teemoittain sekä tarkastellaan tekoälyn tuomia muutoksia DevOps-projektien hallintaan ja johtamiseen. Työssä analysoidaan DevOpsin leviämistä ohjelmistokehityksen ulkopuolelle, erityisesti DevOpsista tekoälymallien hallintaan kehittyneen toimintamallin – MLOpsin – avulla. Työn loppupuolella DevOpsin ja tekoälyn haasteet ovat koottu yhtenäiseen taulukkoon.

Tutkielman keskeisimpänä havaintona erottuu ohjelmistokehitysalan vahva usko DevOpsin menestykseen tekoälyn aikakaudella. Tekoälytyökalut vahvistavat monia DevOpsin ydinpiirteitä, kuten nopeaa kehitystä, automaatiota ja iteratiivisuutta. Lisäksi suuria mahdollisuuksia on löydetty testauksesta, ohjelmiston toiminnan valvomisesta ja virheiden ennakoimisesta. Toisaalta esiin nousee myös monia haasteita. Haasteista nousi esiin kirjallisuudessa erityisesti tekoälymallien luotettavuuden puute, datan hallinta ja alan nuoruuden tuomat haasteet, kuten standardien ja vakiintuneiden käytäntöjen puute.

Avainsanat: AI, AIOps, DevOps, DevSecOps, koneoppiminen, MLOps, tekoäly.

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin Originality Check -ohjelmalla.

TEKOÄLYN KÄYTTÖ

Opinnäytteessäni on käytetty tekoälysovelluksia:

- Ei
- Kyllä

SISÄLLYSLUETTELO

1. JOHDANTO	1
2. TUTKIMUSMENETELMÄT	3
3. TEKOÄLY DEVOPS-ELINKAARESSA	4
3.1 Ohjelmistosisällön tuotanto tekoälyn avulla.....	5
3.2 Jatkuva integraatio ja käyttöönotto.....	6
3.3 Ohjelmistojen turvallisuus ja tekoäly	8
3.4 Projektinhallinta	9
4. DEVOPSIN LAAJENTUMINEN JA MLOPS	10
4.1 MLOPS ja muut alat.....	10
4.2 Standardisoituminen ja AIDOaRt	11
5. DEVOPSIN JA TEKOÄLYN HAASTEET	12
6. KESKUSTELU	14
7. YHTEENVETO.....	16
LÄHTEET	17

LYHENTEET JA MERKINNÄT

AI	engl. Artificial Intelligence eli tekoäly, sateenvarjokäsite malleille ja työkaluille, joissa hyödynnetään erilaisia koneoppimis- ja syväoppimisalgoritmeja sekä neuroverkkoja
CI/CD	engl. Continuous Integration and Continuous Deployment, jatkuva integraatio ja jatkuva käyttöönotto
DevOps	Development and Operations, lähestymistapa ohjelmistokehitykseen, missä yhdistetään ohjelmistokehitys ja palveluntarjonta (engl. operations). [1].
SDLC	engl. Software Development Lifecycle, ohjelmistokehityksen elinkaari.
MLOps	engl. Machine Learning and Operations, DevOps-menetelmä sovellettuna tekoälymallien kehitykseen, hallintaan sekä ylläpitoon

1. JOHDANTO

DevOps on toimintamalli sähköisten palveluiden tuotantoon, joka on kasvattanut suosioaan viime vuosien aikana [1], [2], [3]. DevOpsille ei ole yhtenäistä tai tarkkaa määritelmää. Useissa tutkimuksissa sitä kuvaillaan paradigmaksi, ajattelutavaksi tai jopa omaksi kulttuuriksi [1], [3], [4], [5], [6], mutta toisissa siitä puhutaan kehitysmenetelmänä [2]. Tarkasta määritelmästä huolimatta tutkimuksessa on laajalti konsensus mitä DevOpsin tärkeimpiin ominaisuuksiin ja tavoitteisiin kuuluu.

DevOpsin keskeisimmät ajatukset on tuoda kehittäjien ja palveluntarjoajan (engl. operations) työryhmät lähelle toisiaan. Tämä tapahtuu käytännössä kommunikoinnin sekä automaation avulla. Näin mahdollistetaan nopea reagointi asiakkaiden tarpeisiin sekä joustavuus ominaisuuksien integroinnissa sekä julkaisussa.

Nykyään DevOps on käytössä lähinnä ohjelmistokehityksessä, mutta sen toimintatapoja on hiljattain alettu ottaa käyttöön myös muilla aloilla, kuten ajoneuvojen tuotannossa sekä tekoälymallien kehityksessä ja ylläpidossa [1], [3], [6].

IT-alalta on noussut viime vuosina valtavaan suosioon toinenkin ilmiö: tekoäly. Tekoälymallien ja työkalujen kehittyminen 2020-luvulla on tuonut monille tekniikan aloille suuria muutoksia ja IT-ala ei ole poikkeus [3], [6], [7]. Esimerkiksi Şimşek et al. raportoi tietokoneiden laskentatehon kasvun, datan saatavuuden ja koneoppistekniikoiden kehityksen myötä tekoälyn sovelluskohteiden määrän ohjelmistokehityksessä kasvaneen [4]. Tekoälyn käyttö ohjelmistoalalla kiinnostaa myös tutkijoita, mikä näkyy molempia aloja käsittelevän tutkimuksen moninkertaistumisena viime vuosien aikana [7]. Karlovs-Karlovskis et al. kirjallisuuskatsaus kertoo tapaustutkimuksien puutteen sekä tutkimusten yleisen tason viestivän alan kypsymättömyydestä [7].

DevOps-työkalut ja tekoäly eivät ole missään nimessä kilpailevia teknologioita, vaan tutkimus viittaisikin niiden yhteiskäytöllä olevan suuria mahdollisuuksia. Kuten tässä kirjallisuuskatsauksessa huomataan, DevOps-kehitysmenetelmää on ehdotettu hallitsemaan tekoälymallien elinkaarta MLOpsin muodossa, ja tekoälyyn pohjautuvat työkalut tehostavat lähes kaikkia DevOps-elinkaaren vaiheita. Toisaalta myös uudet mahdollisuudet tuovat mukanaan aina myös uhkia, kuten tekoälyn ennalta-arvaamattomuus sekä yksityisyysongelmat.

Tässä työssä pyritäänkin tarkastelemaan ja erittelemään tekoälyn ja DevOpsin synergiaa. Tämä kaikki pyritään tiivistämään vastaamalla tutkimuskysymyksiin:

1. Pysyykö DevOps relevanttina menetelmänä tekoälyn aikakautena?
2. Mitä mahdollisuuksia tai haasteita tekoäly tuo DevOps-toimintamalliin?

Luvussa 2 "Tutkimusmenetelmät" kerrotaan, miten kirjallisuuskatsaukseen on kerätty aineistot, aineistojen rajaamisperusteet, ja selvennetään termistöä aineistossa. Luvussa 3 perehdytään tekoälyn ja DevOpsin mahdollisuuksiin DevOpsin elinkaaren vaiheiden avulla. Luvussa 4 esitellään tutkimuksista ilmenneitä ajatuksia DevOpsin kehityksestä laajemmin ja kerrotaan tähän liittyen DevOpsista alkunsa saaneesta MLOpsista. Viimein luvussa 5 kootaan taulukkoon DevOpsin ja tekoälyn integraatioon liittyviä haasteita, joita on tuotu esiin läpi kirjallisuuskatsauksen.

2. TUTKIMUSMENETELMÄT

Ainestoa on haettu Andor-hakupalvelulla hakusanalla "AI or "Artificial Intelligence") AND DevOps". Lisäksi haku rajattiin näyttämään vain vuonna 2023 ja sen jälkeen julkaistuja vertaisarvioituja julkaisuja. Aineiston kriteereitä muodostettaessa tunnistettiin rajauksen olevan melko tiukka vuonna 2025. Kriteeriin päädyttiin, sillä tekoäly on kehittynyt niin valtavaa tahtia, ettei esimerkiksi vuoden 2020 tutkimukset ole täysin relevantteja nykytilanteessa. Katsaukseen sisällytettiin vain englanninkieliset tutkimukset. Julkaisuja löytyi haulla 27 kappaletta, joista rajattiin edelleen otsikon ja tiivistelmän perusteella lopullinen määrä 12.

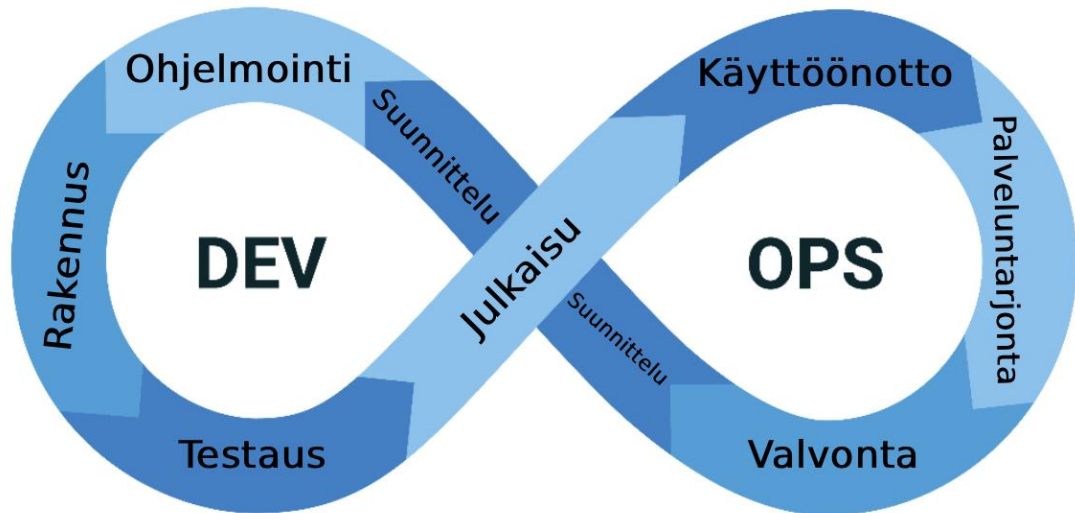
On huomioitavaa, että katsauksessa esiintyneellä lyhenteellä, CI/CD:llä, voidaan tarkoittaa myös jatkuvaa integraatiota ja jatkuvaa toimitusta (engl. continuous integration and continuous delivery). Useammin kirjallisuudessa puhutaan käyttöönotosta (engl. deployment), mutta myös toimitus-termiä esiintyy ajoittain. Termit ovat läheisiä ja välillä niitä käytetäänkin sekaisin tai niillä tarkoitetaan samaa asiaa. Erona termeillä on se, että jatkuva toimitus vie putken askeleen pidemmälle, ja toimittaa ohjelmiston suoraan asiakkaille käyttöympäristöön. Jatkuvassa käyttöönotossa ohjelmisto vaatii vielä ihmisen hyväksynnän ennen tuotantoympäristöä [5]. Katsauksen aineistossa termit olivat eroteltu vain yhdessä tutkimuksessa [8]. Useammassa työssä käytettiin molempia termejä sekaisin, ja niillä tarkoitettiin samaa asiaa [2], [5], [6], [7]. Tässä työssä noudatetaan samaa tapaa käyttää kyseistä termiä, eli CI/CD:llä tarkoitetaan yleisemmin käytettyä käyttöönottoa (engl. deployment). Tämän lisäksi kirjallisuuskatsauksessa on käytetty töitä, missä on käytetty toimitus-sanaa, sillä ero on tässä tapauksessa merkityksetön, eikä se vaikuta tuloksiin tai päätelmiin, ja tapa vaikuttaa yleiseltä alalla.

3. TEKOÄLY DEVOPS-ELINKAARESSA

Tutkimus ohjelmistojen tuotantoon on kiinnostanut tutkijoita viimeisen erityisesti viimeisen vuosikymmenen aikana. Tutkimus ohjelmistotuotannon kehitykseen kasvoi tasaisesti vuodesta 2014 vuoteen 2021. Vuodesta 2021 alkaen kasvava osa alan tutkimuksesta vaikuttaa siirtyneen tekoälyyn, ja osittain myös ohjelmistotuotannon kehitykseen tekoälyn avulla. [9] Tutkimuksissa esiintyy monia eri tarkastelukulmia. Tässä luvussa tarkastellaan ohjelmistokehityksen ja tekoälyn synergiaa ohjelmistokehityksen elinkaaren mallien avulla (SDLC). Lyhenne SDLC tulee englannin kielen sanoista ”software development lifecycle”. Ohjelmiston elinkaarta kuvaavat mallit kartoittavat ohjelmistoprojektin vaiheita alusta loppuun. Sen tarkoituksena on tarjota systemaattinen metodologia ja perusta hyvälaatuiselle ohjelmiston kehitykselle ja ylläpidolle [4]. Tunnetuimpia malleja on ketterä kehitys sekä vesiputousmalli.

SDLC:itä on erilaisia, ja ne pitävät sisällään eri vaiheita. Yleensä ohjelmistokehityksen elinkaaren mallit pitävät sisällään vähintään suunnittelun, toteutuksen, testauksen, julkaisun ja ylläpidon. Joillakin malleilla on lisäksi niille ominaisia, uniikkeja vaiheita. Useimmiten suurimmat erot eri SLDC:iden syntyvät kuitenkin siitä, miten nämä vaiheet käydään läpi. Esimerkiksi perinteisessä vesiputousmallissa vaiheet käydään lineaarisesti peräjälkeen, V-mallissa abstraktista tasosta konkreettiseen toteutukseen ja takaisin, ja ketterässä kehityksessä syklisesti monta kertaa ympäri usealla iteraatiolla.

DevOps juontaa juurensa vahvasti ketterään kehitykseen. Historiansa takia DevOpsissa olennaisista ovat useat iteraatiot sekä näiden myötä syklisyys ja joustavuus. Syklisyys näkyy esimerkiksi itseään ympäri kiertävässä kahdeksikossa, millä usein kuvataan DevOpsin vaiheita ja kulkua [1], [2], [7]. DevOps-kahdeksikkoa havainnollistaa kuva 1.



Kuva 1. DevOps-toimintamallin vaiheet mukaillen [7, kuva 5]

Tutkimukset osoittavat monien DevOpsin elinkaaren vaiheiden todennäköisesti hyötyvän tekoälyn integraatiosta [1], [4], [6], [7], [10]. Esimerkiksi Şimşek et al. kertoo: “tekoälyn viimeisimmät kehityskäsköt, erityisesti automaatiossa ja fasilitaatiotekniikoissa, tarjoaa valtavia mahdollisuuksia ohjelmistokehityksen prosessien suoraviivaistamiselle ja parannukselle, potentiaalisesti mullistaen niiden tehokkuuden ja vaikuttavuuden” [4].

3.1 Ohjelmistosisällön tuotanto tekoälyn avulla

Generatiivinen AI (engl. artificial intelligence) eli tekoäly on tuonut mukanaan valtavia muutoksia ohjelmointiin ja koodin kirjoitukseen. Koodin kirjoitukseen tekoälyllä on kehitetty useita työkaluja, kuten GitHub Copilot ja TabNine [1]. Tekoälypohjaiset työkalut voivat esimerkiksi tukea koodin generaatiota ehdottamalla koodikatkelmia ja useiden kehittäjien tai työryhmien yhteistyötä ehdottamalla sopivia rajapintakutsuja [10]. Koodin luomiseen tarkoitettujen työkalujen lisäksi monet ohjelmoijat käyttävät yleisiä suuria kieli-malleja (LLM), kuten ChatGPT [1], [7]. Näiden käytössä kuitenkin piilee omat riskinsä. Riskejä eritellään tarkemmin kappaleessa 5 ”DevOpsin ja tekoälyn haasteet”.

Myös dokumentaatiota on mahdollista luoda automaattisesti tekoälytyökaluilla [3], [7]. Kattava dokumentaatio helpottaa yhteistyötä useiden kehittäjien ja työryhmien välillä. Tekoälyn avulla dokumentaation luomiseen kuluu vähemmän aikaa, ja sen ylläpito ja päivitys on helpompaa.

Sekä dokumentaation että koodin generaatiossa piilee kuitenkin samankaltaiset ongelmat nykyisellä generatiivisilla tekoälyteknologialla; mallit eivät ymmärrä kontekstia laajasti, ja ne suoriutuvat heikosti kompleksisten ohjelmien ja useiden abstraktiotasojen

kanssa [3, 11]. Yksinkertaisia ohjelmia sekä lyhyitä koodinpätkiä ja niihin kommentteja tekoälymallit luovat melko sujuvasti, mutta yksinkertaisissakin tapauksissa mallit ovat toistaiseksi niin epäluotettavia, että tarvitaan aina asiantunteva ihminen tarkistamaan lopputulos [2], [4], [7], [11].

3.2 Jatkuva integraatio ja käyttöönotto

Yksi DevOpsin määrittävistä piirteistä on siinä käytetyt CI/CD putket. CI/CD lyhenne tulee sanoista jatkuva integraatio (Continuous Integration) ja jatkuva käyttöönotto (Continuous Deployment). Jatkuvalla integraatiolla tarkoitetaan sitä, että uudet ominaisuudet, korjaukset sekä muutokset integroidaan mahdollisimman nopeasti ja pienissä palasissa kehitysympäristöistä eteenpäin testiympäristöihin tai jopa valmiiseen julkaisuun. Tämä tapahtuu usein siten, että ohjelmistokehittäjä vie tekemänsä muutokset etätietovarastoon (repository), koodi rakennetaan automaattisesti ja seuraavaksi rakennettu koodi käy läpi automatisoidut testit. Jos muutokset läpäisevät kaikki nämä vaiheet, koodi integroidaan viimeiseen tarkastukseen produktion kaltaiseen ympäristöön, tai suoraan produktion eli julkaisuun. [11] CI/CD putki nopeuttaa ohjelmistotuotteen julkaisuaikaa merkittävästi ja mahdollistaa tarvittaessa erittäin nopeat korjaukset sekä muutosten palautukset [1], [3].

Kokonaisuudessaan tutkimus tekoälyn tuomista mahdollisuuksista jatkuvassa integroinnissa ja käyttöönotossa on heikohkoa. Tutkimuksissa aliedustettuina aiheina olivat erityisesti koodin rakennus, käyttöönotto sekä julkaisu, jotka ovat keskeisiä osia CI/CD putken toiminnasta [2], [6], [7], [11]. Näissä kuitenkin arvioitiin olevan hyviä mahdollisuuksia jatkotutkimukselle. Huomionarvoista on myös, että koodin generaatio tekoälyllä puolestaan vie yksinään alan tutkimuksesta yli puolet. [2], [7]

Toimiessaan CI/CD putki on jo automatisoitu, joten itse putken toimintaa ei todennäköisesti saada optimoitua tekoälyn avulla. Paremmat mahdollisuudet optimoinnille onkin siis putken luonnissa, sen ylläpidossa sekä poikkeustapausten estossa ja käsittelyssä [4], [11]. Tutkimuksessa onkin ehdotettu, että tekoälyllä DevOps-insinöörit voivat arvioida käyttöönotto ja integraatio putkea, ehdottaa siihen parannuksia ja luoda käyttöönotto skriptejä eli komentosarjoja [4].

Yksinkertaisten skriptien tapaan myös esimerkiksi testien luominen voi viedä reilusti aikaa ja resursseja, vaikka ne olisivatkin suhteellisen suoraviivaisia luoda. Tekoälytyökaluista johtuvan nopeamman ja optimoidun koodin kirjoituksen ansiosta testejä saadaan luotua ja hallittua tehokkaammin [7], [10]. Tekoälyllä voidaan myös luoda suoraan testi-

tapauksia, tunnistaa kriittisiä rajatapauksia ja analysoida automaattisesti virheitä [1]. Tekoälyllä luotuihin tuloksiin tai analyysihin ei voi kuitenkaan luottaa sokeasti, vaan aina tarvitaan ihminen varmistamaan tulokset [2], [4], [11]. Tekoäly voi esimerkiksi luoda tautologisia yksikkötestejä tai ilmoittaa virheellisiä poikkeuksia. Tautologiset testit ovat hyödyttömiä testejä, jotka ohjelmisto läpäisee aina.

Kattava testaus on tekoälyn aikakaudella tärkeämpää kuin koskaan ennen. Ohjelmistojen staattinen analyysi sekä koodin ymmärrys on tärkeämpää tekoälyn aikakaudella kuin koskaan ennen [1], sillä koodia suuri osa koodista on tekoälyn kirjoittamaa [1], ja tekoäly on altis harhoille sekä ennalta-arvaamattomalle käytökselle [2], [4].

Kattavastakin testauksesta huolimatta poikkeuksia, kuten laitteiston vikatoiminta, ohjelmistovirheet tai tietoverkko-ongelmat [9], tapahtuu jokaisessa ohjelmistoprojektissa, ja niitä tulee tarkkailla jatkuvasti ja käsitellä mahdollisimman tehokkaasti. Tekoälyn avulla on mahdollista monitoroida jatkuvasti lokeja ja huomata niistä kuvioita, mitä ihmiset eivät välttämättä huomaisi. Näillä kuvioilla voidaan ennustaa tulevia poikkeuksia [3], [9], [11], mutta lisäksi myös esimerkiksi ennustaa kysyntää tai inventaarion tarvetta tietyille ajalle [1], [5].

Tekoälystä voi olla apua myös poikkeustapausten käsittelyssä. Perinteisillä poikkeusentunnistusmenetelmillä on haasteita toimia DevOps-ympäristöissä datan kompleksisuuden ja valtavan määrän takia [9], [10]. Haasteita on myös datan säilömisessä, käsittelyssä sekä visualisoinnissa [9]. Perinteisesti DevOps-insinöörit joutuvat poikkeustapausten sattua tutkimaan lokeja ja generoitua dataa manuaalisesti [11]. Ongelma on erityisen haasteellinen suurissa projekteissa, sillä datan määrä kasvaa nopeasti niin suureksi, että sitä on mahdotonta käsitellä tehokkaasti [11]. Toisaalta tekoälymalleja on vaikea kouluttaa, sillä hyviä merkittäviä datasettejä, joita tarvitaan tekoälymallien valvottuun koulutukseen (engl. supervised learning), on erittäin vähän [9], [11]. Valvomattomaan oppimiseen (engl. unsupervised learning) ja semi-valvottuun oppimiseen (engl. semi-supervised learning) nojaavista malleista on puolestaan haastava saada tarkkoja suurien datamäärien sekä sisäisten palveluiden monimutkaisen logiikan takia [11]. Mallin heikko tarkkuus merkitsee käytännössä epätydyttävää tehokkuutta [11]. Datasettien sekä toimivien tekoälymallien vähyys oli tunnistettu kirjallisuudessa ongelmakohtaksi ja Ahmed et al. tarjoaakin kehittämänsä tekoälymallia paikkaamaan puutetta [11] ja vastaavasti Kånåhols et al. julkaisee merkityn datasettinsä [9].

3.3 Ohjelmistojen turvallisuus ja tekoäly

Joidenkin lähteiden mukaan yksi DevOpsin heikkouksista on perinteisesti ollut turvallisuus, sillä DevOps projekteissa turvallisuus on otettu huomioon liian myöhään kehitysprosessissa [2]. Tähän ongelmaan on jopa noussut vastaukseksi DevOpsista jatkokehitynyt toimintamalli: DevSecOps. DevSecOps tarkoittaa yksinkertaisesti DevOpsia, missä on yhdistetty kehityksen ja IT-toimintojen lisäksi turvallisuus [2]. Tekoälyn tuomat muutokset luovat DevOpsiin uusia turvallisuusriskejä, joten on syytä analysoida riittävästi perinteiset DevSecOps-menetelmät riskienhallintaan tekoälyn aikakaudella. Toisaalta tekoäly luo myös uusia turvallisuusmahdollisuuksia.

Fu et al. listasi tutkimuksessaan ”AI for DevSecOps” 11 eri tehtävää, missä tekoälyllä voitaisiin tulevaisuudessa parantaa ohjelmistojen kyberturvallisuutta projektin DevOps-syklin aikana [2]. Taulukko 1 esittelee tehtävät kootusti.

Taulukko 1. *Tekoälyn turvallisuustehtävät mukailten [2, taulukko 16]*
DevOps kehityksen vaihe **Tehtävä**

DevOps kehityksen vaihe	Tehtävä
Kehitys ja koodin kirjoitus	Ohjelmistojen haavoittuvuuksien havaitseminen
	Ohjelmistojen haavoittuvuuksien luokittelu
	Automatisoitu haavoittuvuuksien korjaus
	AI-turvallisuustyökalut ohjelmointiympäristöissä
Koodin rakennus, testaus ja käyttöönotto	AI-turvallisuustyökalut CI/CD putkissa
	Konfiguraatioiden ja asetusten varmistus
	Infrastruktuurin skannaaminen
Suunnittelu	Uhkamallinnus
	Ohjelmistojen muutosvaikutusanalyysi
Toiminnot ja monitorointi	Lokien analyysi ja virheiden havaitseminen
	Kyber-fyysisten järjestelmien turvallisuus

Tutkimuksessa löydettiin uusia mahdollisuuksia parantaa ohjelmistojen turvallisuutta lähes jokaisessa DevOps-syklin vaiheessa. Osassa ehdotuksissa, kuten uhkamallinnuksessa, tekoälytyökalut toimivat ihmisten apuna helpottaen ja tehostaen heidän työtään. Tutkimuksessa oli ehdotuksia myös täysin autonomisista tekoälytyökaluista, joilla voitaisiin parantaa turvallisuutta. Tällaisia olivat esimerkiksi automatisoitu haavoittuvuuksien korjaus sekä AI-työkalut CI/CD-putkissa.

Tutkimuksessa huomautettiin kuitenkin, että tekoälyperusteiset turvatoimet **eivät** ole yksinään riittäviä. Tutkimusympäristöt eivät vastaa täysin oikeiden produktioympäristöjen tilanteita, ja järjestelmien vuorovaikutus ihmisten kanssa saattaa tuottaa haasteita, joita tekoälytyökalut eivät kykene korjaamaan halutulla tavalla. On myös huomioitavaa, että

vaikka DevOps on alkanut leviämään ohjelmistokehityksen ulkopuolellekin [1], [3], [6], kyseinen tutkimus ja taulukko koskee lähinnä ohjelmistoprojekteja

3.4 Projektinhallinta

Tekoälyn vaikutus projektinhallintaan henkilöstön ohjauksen näkökulmasta on epävarmaa. Ennusteet ovat jakaneet asiantuntija pitkälti kahteen joukkoon. Ensimmäinen joukko arvioi, että tekoäly ei kykene korvaamaan ihmisten välistä vuorovaikutusta, ja siten olisi kykenemätön sivuuttamaan esimerkiksi johtajien rooleja. He perustelevat väitettä sillä, että AI-malleilta puuttuu maalaisjärki, empatia, moraalit ja muita ihmisille ominaisia kognitiivisia kykyjä, minkä takia esimerkiksi ryhmäytyminen, ryhmähengen ylläpito ja motivaation rakentaminen on tekoälylle vaikeaa. [4]

Toinen joukko tutkijoita puolestaan uskoo tekoälyn pystyvän hoitamaan organisaationaalisia tehtäviä tulevaisuudessa, minkä takia osa johtajiston rooleista jää tarpeettomaksi [4]. Tekoälyn avulla voidaan helpottaa organisointia ja analysoida tilastoja sekä metriikoita. Sitä voidaan käyttää apuna suunnittelussa, ja tehdä arvioita työvoiman sekä -tuntien tarpeesta.

Tekoälyn avulla tehdystä data-analyysistä ja mielipiteidenkeruusta on hyötyä erityisesti, silloin kun ne yhdistetään DevOpsille ominaisiin lyhyisiin palautesilmukoihin. Näiden työkalujen avulla voidaan kerätä tehokkaasti dataa ja palautetta asiakkailta, mikä helpottaa päätöstentekoa ja ominaisuuksien priorisointia [3], [8].

4. DEVOPSIN LAAJENTUMINEN JA MLOPS

Toistaiseksi DevOps on ollut käytössä vain lähinnä ohjelmistoalalla, josta se on saanut alkunsa. Hiljattain DevOps-menetelmiä on kuitenkin sovellettu onnistuneesti muillekin aloille [1], [3], [6]. Monet MLOpsiin keskittyvät tutkimukset arvioivat DevOps-käytäntöjen leviämisen jatkuvan ohjelmistoalan ulkopuolelle muun muassa MLOpsin myötä [5], [6], [8], [12].

4.1 MLOPS ja muut alat

MLOps lyhenne tulee englanninkielisistä sanoista ”machine learning and operations” eli koneoppiminen ja palveluntarjonta. Se tarkoittaa koneoppimismallien kehittämistä, ylläpitoa sekä hallintaa DevOpsin periaatteiden mukaisesti [5], [6], [8], [12]. Lähdekoodin sijasta koodin muutoksia, iteraatioita ja versionhallintaa voidaan käyttää tekoälymallien hallinnassa. Malleille voidaan esimerkiksi tuoda uutta koulutusdataa. Tämä esimerkkitapauksen uusi data siirtyisi jälleen automatisoituun putkeen, missä mallit voidaan kouluttaa, testata ja validoida [5]. Putken lopussa päivitetty malli voidaan halutessaan julkaista. Julkaisun jälkeen MLOps-toimintamalliin kuuluu myös jatkuva mallien toiminnan seuraaminen. Mallien jatkuvan seuraamisen ansiosta voidaan esimerkiksi saada ilmoituksia, jos mallin tarkkuus putoaa liian alas [5]. MLOpsilla saadaan tehostettua eri työryhmien välistä työskentelyä sekä jatkuvan palautteen saamista ja palautteen perusteella tehtyjen muutosten integroimista tekoälymalleihin [5]. Lisäksi MLOpsilla, kuten perinteisellä DevOpsillakin voidaan nopeuttaa merkittävästi tuotteen markkinoilletuontiaikaa [5].

MLOpsin on arveltu lisäävän myös tekoälymallien luotettavuutta kehittämällä niiden jäljittelevyyttä sekä toistettavuutta [12]. Tekoälyn luotettavuus ja sen tulosten läpinäkyvä toimivuus ovat yksiä yleisimpiä huolenaiheita tekoälyn kehityksessä [2], [4], [12]. Luotettavuus nousee erityisesti ongelmaksi, kun malleja täytyy muuttaa tai päivittää. Tähän on yhtenä syynä niin sanottu CACE-periaate. CACE on Google Researchin nimittämä lyhenne sanoista: ”changing anything changes everything” eli suomennettuna ”minkä tahansa muuttaminen muuttaa kaiken” [12]. CACE tarkoittaa, ettei malleihin voida tehdä pieniäkään muutoksia huoletta. Muutosten myötä pitää varmistaa mallin toiminta täysin uudelleen, sillä niiden laajuudesta ei voida olla varmoja ja pieneltäkin vaikuttavalla muutoksella voi olla suuria vaikutuksia mallin toimintaan.

Tekoälyn ennalta-arvaamattomuuden takia pelkkä lopputuloksen, eli valmiin mallin, testaaminen ei riitä takaamaan luotettavaa mallin toimintaa [12]. Luotettavuuden kannalta

lopullisen mallin testausta tärkeämpää onkin mallien kehittämisprosessi sekä hallinta [12]. Tutkimuksissa on ehdotettu, että selkeä ja strukturoitu kehitysprosessi ja luotettava infrastruktuuri olisi mahdollista luoda MLOpsin avulla [5], [12].

Tekoälymallit ovat yleistyneet valtavasti, ja niitä käytetään jo monipuolisesti eri toimialoilla [3], [6], [12] esimerkiksi julkisväestön mielipiteiden arvioimiseen ja kyberturvallisuuden tehostamiseen [5], [8]. Tämän voidaan arvioida johtavan DevOps-insinöörien kasvavaan kysyntään, sillä heitä tarvitaan DevOps putkiston ylläpitoon ja kehitykseen, sekä poikkeustilanteiden hoitoon [6], [12]. Tekoälyyn pohjautuviin ratkaisuihin ei ole hopealuoti-tekoälymallia, mikä toimisi monenlaisissa käyttökohteissa, vaan tarvitaan asiantuntija valitsemaan sopivat mallit ja algoritmit [9]. Toisaalta tekoälyn kehityksellä arvioidaan olevan myös heikentävä vaikutus DevOps-insinöörien tarpeeseen, sillä esimerkiksi CI/CD putkien hoito vaikuttaa onnistuvan tekoälyltä melko hyvin [4].

4.2 Standardisoituminen ja AIDOaRt

IT-ala on murroksessa tekoälyn tuomien muutosten takia. Nopeiden ja suurien muutosten takia alalle ei ole vielä muodostunut selkeitä standardeja tai käytäntöjä. Sopivien työkalujen ja standardien puute onkin noussut esiin monissa eri tutkimuksissa [2], [5], [6], [12].

Alan kypsyessä rakenteellisuutta ja yhtenäisyyttä kuitenkin tarvitaan [12]. Yksi merkittävimmistä yrityksistä rakentaa koheesiota ja eheistää alan teknologioita on eurooppalainen AIDOaRt projekti. AIDOaRt on ehdotettu arkkitehtuuri, mikä yhdistää DevOpsin ja mallipohjaisen suunnittelun sekä tehostaa näitä erilaisilla tekoälyyn ja koneoppimiseen pohjautuvilla optimisaatioilla sekä parannuksilla [10]. Käytännössä sitä voidaan ajatella konkreettisenä arkkitehtuurina MLOpsille.

AIDOaRt:in on raportoitu parantavan projektien onnistumista kokonaisuudessaan, helpottavan integraatiota, vaatimusten täyttämistä sekä selkeän ja iteroitavan suunnittelun luomisessa. Heikkoutena AIDOaRt:issa tunnistettiin olevan erityisesti sen käyttöönotto alkuvaiheissa arkkitehtuurin monimutkaisuuden takia. Lisäksi haasteita oli arkkitehtuurin laadun analyysin arvioimisessa. [10]

Toinen tutkimuksissa esiin noussut merkittävä liike, joka sääntelisi ja yhdenmukaistaisi tekoälytyökaluja, on Euroopan unionin ehdotettu tekoälyasetus (engl. the AI act) [12]. Asetuksen tavoitteena olisi kehittää ennen kaikkea *luotettavaa* tekoälyä. Asetuksen mukaiset luotettavat tekoälyjärjestelmät olisivat lainmukaisia, eettisiä ja vakaita. [12] Monet nykyhetken puhututtavimmat tekoälyn haasteet liittyvätkin turvallisuuteen ja luotettavuuteen.

5. DEVOPSIN JA TEKOÄLYN HAASTEET

Jokaisessa kirjallisuuskatsauksessa analysoidussa tutkimuksessa käsiteltiin DevOpsin, MLOpsiin ja tekoälyyn liittyviä haasteita sekä riskejä. AI-tehosteiseen DevOpsin eli AIOpsiin ja MLOpsiin kohdistuvat ongelmat ovat hyvin samankaltaisia [6]. Ongelmien yhtenäisyyden takia niitä käsitellään alan tutkimuksessa toisinaan yhdessä [6], kuten tässäkin työssä on tehty. Taulukko 2 esittelee haasteet kootusti kategorisoituna.

Taulukko 2. *DevOpsin ja tekoälyn haasteet*

Kategoria	Tunnistettu riski	Tutkimus
Nykyhetken tekniset haasteet	Datan hallinta	[3], [5], [9], [10]
	Heikko koulutusdatan laatu ja määrä	[2], [8], [9], [11]
	Työkalujen, mallien ja standardien puute	[2], [5], [6], [12]
Luotettavuus	Mallien selittämättömyys ja läpinäkymättömyys	[1], [2], [4], [7], [12]
	Tekoälyn vinoumat/harhat (biases)	[3], [4]
	Yliluotto tekoälyyn johtaa virheisiin	[4], [7]
Turvallisuusriskit	Kyberturvallisuus uhat	[1], [2]
	Yksityisyyden loukkaukset	[5], [12]
	IP-oikeuksien rikkominen	[1], [5], [12]
Riskit tulevaisuudessa	Lait voivat rajoittaa tulevaisuudessa?	[5], [12]
	Pelko työpaikkojen häviämisestä	[4]
	Tekninen velka	[1]

Monessa tutkimuksessa nostettiin luonnollisesti esiin nykyhetkisen teknologian puutteita sekä ongelmakohtia. Ongelmiksi tunnistettiin datan hallinta, hyvälaatuisen koulutusdatan puute sekä vakiintuneiden ratkaisuiden ja käytäntöjen puute. Tekoälymallien koulutuksessa ja DevOps-putkien lokeissa dataa on usein niin valtava määrä, että sen käsittely ja sen yhtenäisyyden varmistaminen voi olla haasteellista [3], [5], [9], [10]. Nykyteknologian ongelmiin kehittyvät koko ajan ratkaisuja, ja monien mainituista ongelmista voidaan arvella lievenevän tai jopa ratkeavan tulevina vuosina.

Mallien luotettavuus on noussut useassa tutkimuksessa esiin yhtenä suurimmista haasteista tekoälyn kanssa. Nykyajan tekoälymallien päätöksenteko ei ole tarpeeksi läpinäkävää tai determinististä, eli sitä ei voida varmuudella ennustaa. [1], [2], [4], [7], [12] Ennalta-arvaamattomuuden lisäksi tekoälymalleihin voi ujuttautua niiden koulutusdatan mukana vinoumia tai harhauksia, mikä heikentää mallien ja samalla koko ohjelmistojen luotettavuutta [3], [4], [7].

Tutkimusten mukaan tekoälyn koulutukseen tarvittavan datan keruussa voi ilmetä myös ongelmia. Huolta on muun immateriaalioikeuksien rikkomuksista, kuten tekijänoikeusrikkouksista, yksityisyysoikeuksien rikkomuksista sekä kyberturvallisuusriskeistä [1], [2], [5], [12]. On esimerkiksi mahdollista, että tekoälymallin koulutusdataan pääsee salaista tietoa, mikä voi ilmaantua koulutetun mallin tuloksiin.

Epävarmuutta teknologian kehitykselle luo myös ennusteet tekoälyä koskevien lakien ja sääntelyn muutoksista tulevaisuudessa [5], [12]. On mahdollista, että sääntely tulee rajoittamaan tulevaisuudessa tekoälymallien toimintaa tai kouluttamista, mikä voi heikentää joidenkin tekoälytyökalujen tehoa. Lisäksi tutkimuksissa mainittiin huolen aiheiksi työllisyyden epävarmuus [4], sekä kasaantuva tekninen velka [1]. Teknisellä velalla tarkoitetaan esimerkiksi ohjelmistoja, joita on haastava ylläpitää ja päivittää. Tekninen velka on jo nykyäänkin ongelma nopean toimituksen ja iteraatioiden ohjelmistokehityksessä. Jos tahtia kiihdytetään entisestään tekoälyllä, on mahdollista, että myös teknisen velan määrä tulee nousemaan. [1]

6. KESKUSTELU

Tekoälyn kehitys on tuonut mukanaan valtavia muutoksia monille aloille. Suuret muutokset voivat luoda uusia mahdollisuuksia, mutta myös herättää huolta esimerkiksi työtehtävien muutoksista tai poistumisesta. IT-alalla tekoälyn siivittäjä murros on ollut erityisen tuntuva. Tähän on varmasti monia syitä; IT-alalla saatetaan ottaa innokkaammin uusia teknologioita käyttöön kuin monilla muilla aloilla. Tekoälyteknologioiden suosioon IT-alalla vaikuttaa varmasti myös se, että tekoälyn kehitys itsessään on osa IT-alaa. Tämä tarkoittaa sitä, että IT-alan asiantuntijat ymmärtävät tekoälyteknologiasta todennäköisesti monien muiden alojen asiantuntijoita enemmän, joten työkaluja voidaan ottaa käyttöön, vaikka ne olisivat vielä varhaisessa kehityksen vaiheessa, ja siten vielä monimutkaisia ymmärtää ja käyttää.

Suuren paradigman muutoksen tapahtuessa on hyvä tarkastella olemassa olevia käytäntöjä ja ajattelutapoja, ja pohtia onko niitä syytä päivittää tai korvata kokonaan. Tämän työn ensimmäinen tutkimuskysymys pyrki vastaamaan näihin kysymyksiin DevOps-toimintamallin osalta. Kirjallisuuskatsauksessa analysoidut, tuoreet tutkimukset olivat miltein yksimielisiä siitä, että DevOps pysyy relevanttina toimintamallina ja saattaa jopa kasvattaa suosiotaan. Tekoälyllä ja DevOps-toimintamallilla ennustetaan olevan hyvä synergia, ja niiden arvellaan tehostavan toisiaan; tekoälytyökalut parantavat DevOps-työnkulkua [1], [2], [4], [6], [7], [10], [11] ja DevOps puolestaan vaikuttavaa erittäin lupaavalta menetelmältä ylläpitää ja hallita tekoälymalleja MLOpsin keinoin [5], [12]. Tutkimuksissa nousi esiin myös muun muassa AIDOOaRt projekti [10], missä DevOps on keskeisessä roolissa. AIDOOaRt on suuri Euroopan laajainen pyrkimys standardisoida AI-veitoista kyberfyysisten järjestelmien kehitystä. AIDOOaRt:in kaltaiset suuret projektit kertovat alan jatkuvasta kiinnostuneisuudesta DevOpsiin.

Toisella tutkimuskysymyksellä pyrittiin analysoimaan tarkemmin tekoälyn tuomia muutoksia. Kirjallisuuskatsauksessa tarkastellut tutkimukset toivat esiin monia mahdollisuuksia DevOps-syklin eri vaiheisiin useasta näkökulmasta [1], [2], [4], [6], [7], [10], [11]. Tutkimukset osoittivat mahdollisuuksia tehostaa koodin ja dokumentaation kirjoittamista tekoälytyökaluilla [1], [3], [7], parantaa ohjelmistojen turvallisuutta ja luotettavuutta [2], [12], parantaa CI/CD-putkia ja helpottaa niiden luomista sekä ylläpitämistä [1], [4], [7], [10], monitoroida ohjelmistojen toimintaa sekä ennustaa virheitä [3], [9], [11] ja helpottaa projektinhallintaa [3], [4].

Tekoälyn siivittäjä kehitys ei kuitenkaan ole vailla omia haasteitaan. *Jokaisessa* kirjallisuuskatsauksessa analysoidussa tutkimuksissa tuotiin esiin tekoälyn luomia ongelmakohtia. Haasteista yleisin ja vakavin oli luottamuksen puute. Tutkimukset osoittivat, että tekoälyn kanssa työskennellessä tarvitaan aina asiantunteva ihminen varmistamaan tekoälyn tulokset [2], [4], [11]. Tämä johtuu siitä, että nykyteknologian tekoälymallit ovat läpinäkymättömiä päätöksenteossa, eikä sen vastauksia voida ennustaa varmuudella. Ongelma korostuu erityisesti sovelluskohteissa, joissa virheet ovat kriittisiä, kuten lääketieteeseen tai maanpuolustukseen luodut ohjelmistot tai tekoälymallit. Olisi myös syytä ottaa huomioon mahdolliset ongelmat tilanteissa, joissa tekoälymallien tai -työkalujen virhetoimintaa myös valvotaan tekoälyllä. Yliluotto tekoälyyn voi johtaa huomaamattomiin ongelmiin. Muita yleisesti esiintyneitä ongelmia oli yksityisyys- ja IP-oikeuksien rikkinen [1], [5], [12] sekä alan kypsymättömyys ja sen luomat haasteet, kuten hyvälaatuisen datan puute ja epävarmuus sääntelystä tulevaisuudessa [2], [5], [6], [12].

Tässä työssä saavutetut tulokset koskevat tekoälyä ja DevOpsia vain yleisellä tasolla, eikä tiettyihin työkaluihin tai teknologioihin oteta kantaa. Tässä työssä nimeltä mainittuja työkaluja tulisi tulkita vain havainnollistavina esimerkkeinä. Täten tutkimuksen tulokset soveltuvat paremmin pohjaksi jatkotutkimukselle aiheeseen liittyviin teknologioihin, erikoisaloihin tai menetelmiin kuin suositukseksi käytännön projekteihin. On myös huomioitavaa, että vaikka DevOps on alkanut hiljattain leviämään myös IT-alan ulkopuolelle, keskittyy valtaosa nykytutkimuksesta vielä toistaiseksi ohjelmistokehitykseen. Tutkimuksessa muodostetut päätelmät haasteista ja mahdollisuuksista eivät siis ole luotettavia ohjelmistokehityksen ulkopuolella. Toisaalta monet mahdollisuudet ja haasteet eivät ole sidoksissa ohjelmistoaan, joten tuloksia voidaan pitää suuntaa antavina ohjelmistoaan ulkopuolellakin.

7. YHTEENVETO

Tuore tutkimus viittaa vahvasti DevOps-toimintamallin käytön jatkuvan ja mahdollisesti jopa kasvavan tekoälyn yleistyessä. DevOps-ajattelutapaan kuuluu keskeisinä osina automatisaatio ja nopea kehitys, ja tutkimukset osoittavat tekoälyn tuovan monia mahdollisuuksia kehittää automaatiota sekä nopeuttaa ohjelmistokehitystä entisestään. Tekoälytyökalut nopeuttavat ja helpottavat koodin kirjoitusta, CI/CD-putkien luomista ja hallintaa, sekä virheiden valvomista ja ennustamista. Käänteisesti DevOps-toimintamallia ja erityisesti DevOpsista jatkokehittyntä MLOps-mallia pidetään hyvänä tapana hallita, ylläpitää ja kehittää tekoälymalleja. Tekoäly ja DevOps ruokkivat siis toistensa kehittymistä. Tutkimukset osoittivat myös, ettei tekoälyllä pystytä täysin korvaamaan ihmisrooleja ohjelmistokehityksessä. Suurimpia haasteita tekoälyn ja DevOpsin integraatiolle ja kehitykselle on tekoälyn luotettavuuden puute, datan hallinta ja alan nuoruus. Nuoruuden takia alalla ei ole vielä vakiintuneita käytäntöjä tai standardeja, eikä kaikkiin tarpeisiin sopivia malleja tai datasettejä. Tutkimukset osoittavat luottamuksen tekoälyn ja DevOpsin yhteistoiminnan menestykseen olevan kuitenkin vahvaa, mikä näkyy esimerkiksi suurena Euroopan laajuisena AIDOOaRt-yhteistyöprojektina. Jatkotutkimukselle olisi tarvetta erityisesti matalalla tasolla spesifeihin ongelmiin, sekä konkreettisiin teknologioihin ja mahdollisuuksiin.

LÄHTEET

- [1] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "DevOps 2.0," *IEEE Softw.*, vol. 42, no. 2, pp. 24–32, Mar. 2025, doi: 10.1109/MS.2025.3525768.
- [2] M. Fu, J. Pasuksmit, and C. Tantithamthavorn, "AI for DevSecOps: A Landscape and Future Opportunities," *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 4, pp. 1–61, May 2025, doi: 10.1145/3712190.
- [3] H. H. Olsson and J. Bosch, "Strategic digital product management: Nine approaches," *Information and Software Technology*, vol. 177, p. 107594, Jan. 2025, doi: 10.1016/j.infsof.2024.107594.
- [4] T. Şimşek, Ç. Gülşeni, and G. A. Olcay, "The Future of Software Development With GenAI: Evolving Roles of Software Personas," *IEEE Eng. Manag. Rev.*, pp. 1–8, 2024, doi: 10.1109/EMR.2024.3454112.
- [5] H. J. Watson and D. Larson, "MLOps: From a Cottage Industry to a Factory Approach," *International Journal of Business Intelligence Research*, vol. 15, no. 1, pp. 1–22, Nov. 2024, doi: 10.4018/IJBIR.358916.
- [6] J. Diaz-de-Arcaya, A. I. Torre-Bastida, G. Zárate, R. Miñón, and A. Almeida, "A Joint Study of the Challenges, Opportunities, and Roadmap of MLOps and AIOps: A Systematic Survey," *ACM Comput. Surv.*, vol. 56, no. 4, pp. 1–30, Apr. 2024, doi: 10.1145/3625289.
- [7] U. Karlovs-Karlovskis, "Generative Artificial Intelligence Use in Optimising Software Engineering Process: A Systematic Literature Review," *Applied Computer Systems*, vol. 29, no. 1, pp. 68–77, June 2024, doi: 10.2478/acss-2024-0009.
- [8] A. Mussina, D. Yedilkhan, Y. Alimzhanov, A. Nugumanova, S. Aubakirov, and A. Mansurova, "USING MLOPS FOR DEPLOYMENT OF OPINION MINING MODEL AS A SERVICE FOR SMART CITY APPLICATIONS," *Scientific Journal of Astana IT University*, pp. 104–124, Mar. 2025, doi: 10.37943/21CPQX5616.
- [9] G. Kånåhols, S. Hasan, and P. Erik Strandberg, "Integrating Time Series Anomaly Detection Into DevOps Workflows," *IEEE Access*, vol. 13, pp. 46459–46477, 2025, doi: 10.1109/ACCESS.2025.3550665.
- [10] R. Eramo, B. Said, M. Oriol, H. Bruneliere, and S. Morales, "An architecture for model-based and intelligent automation in DevOps," *Journal of Systems and Software*, vol. 217, p. 112180, Nov. 2024, doi: 10.1016/j.jss.2024.112180.
- [11] A. Hany Fawzy, K. Wassif, and H. Moussa, "Framework for automatic detection of anomalies in DevOps," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 3, pp. 8–19, Mar. 2023, doi: 10.1016/j.jksuci.2023.02.010.
- [12] M. Borg, "Pipeline Infrastructure Required to Meet the Requirements on AI," *IEEE Softw.*, vol. 40, no. 1, pp. 18–22, Jan. 2023, doi: 10.1109/MS.2022.3211687.