Sini Annamaa

KONEOPPIMISEN MERKITYS OHJELMISTOTESTAUKSELLE ORGANISAATIOISSA

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2023

TIIVISTELMÄ

Annamaa, Sini
Koneoppimisen merkitys ohjelmistotestaukselle organisaatioissa
Jyväskylä: Jyväskylän yliopisto, 2023, 28 s.
Tietojärjestelmätiede, kandidaatintutkielma
Ohjaaja: Mehtälä, Saana

Ohjelmistotestauksen rooli moderneissa organisaatioissa on muutoksessa, ja tehokkaampia ohjelmistotestausmenetelmiä etsitään jatkuvasti. Tässä kandidaatintutkielmassa tarkastellaan koneoppimisen integroimista ohjelmistotestaukseen organisaatioiden kontekstissa. Aihetta on tärkeä tutkia, sillä kiinnostus tekoälyteknologioiden, kuten koneoppimisen, hyödyntämiselle on suuressa kasvussa modernissa organisaatiokulttuurissa. On myös tärkeää lisätä aiheeseen liittyvää tietämystä tutkimalla koneoppimisen merkitystä ohjelmistotestauksessa huomioonottamalla organisatoriset näkökohdat. Tutkielman tavoitteena on tunnistaa mahdollisuuksia ja haasteita, jotka liittyvät koneoppimisen integroimiseen ohjelmistotestaukseen organisaation näkökulmasta. Tutkielma toteutetaan systemaattisena kirjallisuuskatsauksena. Tutkielmassa käydään läpi koneoppimisen ja ohjelmistotestauksen käsitteitä sekä tärkeimpiä menetelmiä ja strategioita, joita tällä hetkellä käytetään sekä koneoppimisen että ohjelmistotestauksen soveltamiseen. Tutkielman tulokset viittaavat vahvasti siihen, että koneoppimisen integrointi ohjelmistotestauksessa tuo mukanaan niin hyviä kuin huonoja puolia. Koneoppimisen integrointi mahdollistaa muun muassa testauksen automatisoinnin, tehokkaammat testausprosessit, ja ennakoivan analytiikan hyödyntämisen. Integrointiin liittyy kuitenkin haasteita, kuten tiedon laatu ja määrä, tietoturva- ja yksityisyysongelmat, sekä eettiset näkökohdat. Edellä mainitut kohdat ovat haastavia, koska koneoppimisen integraatiota ei nähdä ainoastaan teknisenä haasteena, vaan myös sekä organisaation kulttuuriin että toimintatapoihin vaikuttavana haasteena.

Asiasanat: koneoppiminen, ohjelmistotestaus, organisaatiot

ABSTRACT

Annamaa, Sini
The Importance of Machine Learning for Software Testing in Organisations
Jyväskylä: University of Jyväskylä, 2023, 28 pp.
Information Systems, Bachelor's Thesis
Supervisor: Mehtälä, Saana

The role of software testing in modern organisations is changing, and more effective software testing methods are constantly being searched. This bachelor's thesis examines the integration of machine learning into software testing in the context of organisations. This is an important topic to explore as interest in the use of artificial intelligence technologies, such as machine learning, is growing rapidly in modern organisational culture. It is important to increase knowledge on the topic by studying the relevance of machine learning for software testing, also by considering organisational aspects. The aim of the study is to identify opportunities and challenges related to the integration of machine learning in software testing from an organisational perspective. The thesis is carried out as a systematic literature review. The thesis will review the concepts of machine learning and software testing and the main methodologies and strategies currently used to apply both machine learning and software testing. The results of the thesis strongly suggest that the integration of machine learning with software testing has both advantages and disadvantages. The integration of machine learning enables, among other things, the automation of testing, more efficient testing processes, and the use of predictive analytics. However, there are challenges to integration, such as the quality and quantity of data, security and privacy concerns, and ethical considerations. The above points are challenging because machine learning integration is seen not only as a technological challenge, but also as a challenge that affects both the culture and practices of the organisation.

Keywords: machine learning, software testing, organisations

KUVIOT

TAULUKOT

SISÄLLYS

1 JOHDANTO

Elämme aikakautta, jolloin yksityishenkilöt ja yritykset tuottavat valtavia määriä dataa eri toimintojen kautta. Alpaydinin (2014) mukaan toiminnot, kuten ostaminen, selaaminen ja postittaminen, sekä triljoonat verkkosivut, tuottavat valtavan määrän dataa tehden kaikista meistä datan tuottajia ja kuluttajia. Hänen mukaansa tästä syntyvän datatulvan myötä käsillämme voi olla pian jopa tietoräjähdys.

Koneellinen oppiminen määritellään joukoksi menetelmiä, joilla havaitaan automaattisesti kuvioita erilaisia tietoja käsiteltäessä (Murphy, 2021). Nämä menetelmät käyttävät havaittuja malleja ennustamaan tulevia tietoja tai tekemään päätöksiä epävarmuuden vallitessa, tehden siitä merkittävän työkalun kasvavassa datatulvassa (Toreini ym., 2020). Koneoppimista korostetaankin tekoälyteknologioista keskeisenä lähestymistapana etenkin suuren datamäärän hallitsemiseen liittyvien vaikeuksien voittamisessa (Pap ym., 2022). Laskentatehon räjähdysmäinen kasvu ja tehokkaiden koneoppimistyökalujen saatavuus ovatkin edistäneet koneoppimisalgoritmien laajaa käyttöönottoa (Yadav ym., 2021).

Ohjelmistoteollisuus on kasvanut merkittävästi, ja siitä on tullut entistä kilpailukykyisempi ja kattavampi eri aloilla (Mohagheghi ym., 2009). Ohjelmistoteollisuudessa hyödynnetään usein ketterää kehittämistä, jossa yhdistyy ensisijaisesti toimiva ohjelmisto, loppukäyttäjien osallistaminen, aktiivinen viestintä sekä nopea reagointikyky (Trzeciak ym., 2022). Modernissa ohjelmistokehityksessä nähdäänkin olevan meneillään uudistus, jossa uudelleen määritellään siihen kuuluvien vaiheiden, kuten ohjelmistotestauksen, rooleja menestyksellisemmän ohjelmistokehityksen mahdollistamiseksi (Alahyari ym., 2017).

Ohjelmistotestausta pidetään ensisijaisena keinona varmistaa ohjelmiston luotettavuus ja täyttää toiminnalliset vaatimukset (Ammann & Offutt, 2016). Ohjelmistotestausta on laajalti hyödynnetty muun muassa keinona auttaa insinöörejä kehittämään korkealuokkaisia ohjelmistoja. Se on tärkeä prosessi, joka suoritetaan laadunvarmistuksen tukemiseksi keräämällä tietoja tutkittavan ohjelmiston luonteesta (Chan ym., 2009). Koneoppimisen kasvava soveltaminen eri aloilla ja työtehtävissä on herättänyt kasvavaa kiinnostusta automatisoida erilaisia ohjelmistosuunnittelutoimintoja, kuten ohjelmistotestausta. Tutkielman

tavoitteena on tutkia koneoppimisen tuomia hyviä ja huonoja puolia ohjelmistotestauksessa organisaatioiden näkökulmasta, ja vastata seuraavaan tutkimuskysymykseen:

- Minkälaisia ovat mahdollisuudet ja haasteet koneoppimisen integroimisesta ohjelmistotestaukselle organisaatioiden kontekstissa?

Tutkielma toteutetaan systemaattisena kirjallisuuskatsauksena. Kirjallisuuskatsauksessa hyödynnetään Templierien ja Parén (2015) esittämiä suuntaviivoja tietoyhteiskuntatieteellisen kirjallisuuden katsausten ohjaamiseksi ja arvioimiseksi. Tutkimuksen kirjallisuus pohjautuu 34 tieteelliseen artikkeliin ja konferenssipaperiin, mukaan lukien muutamaan tieteelliseen kirjaan korvaamaan aiheeseen liittyvän korkealaatuisten artikkeleiden puutetta, erityisesti koneoppimisen osalta.

Tutkielman kirjallisuus on kerätty tietokannoista, kuten IEEE Xplore, JYKDOK, Google Scholar, ScienceDirect ja Scopus. Hakuprosessissa hyödynnettiin muun muassa seuraavia hakusanoja: *artificial intelligence*, *machine learning*, *software development*, ja *software testing*. Rajallisen suomenkielisen aineiston saatavuuden vuoksi tutkielmassa hyödynnetään ainoastaan englanninkielistä kirjallisuutta. Tutkielman kirjallisuutta kerättäessä huomioidaan niin viittausten lukumäärä kuin sen vaikuttavuus alalla. Lisäksi kirjallisuuden tulee olla merkityksellinen tutkimukselle ja pystyä auttamaan sekä tukemaan tutkimuskysymykseen vastaamisessa. Kirjallisuudessa huomioidaan Julkaisufoorumin antama luokitus. Jokainen tutkielmaan valittu kirjallisuus on täyttänyt vähintään Julkaisufoorumin tason yksi kriteerit. On hyvä huomioida, ettei Julkaisufoorumin antama luokitus ole aukoton, sillä luokituksissa ei ole huomioitu esimerkiksi arvioidun tutkimuksen yhteiskunnallista vaikuttavuutta. Tämän vuoksi tutkielman ulkopuolelle on voinut jäädä tutkimuksen aiheen kannalta olennaista kirjallisuutta. Kuitenkin jokainen tutkielmaan valikoitu kirjallisuus on tarkistettu Julkaisufoorumia hyödyntäen, jotta pystytään varmemmin luottamaan ja varmistamaan kirjallisuuden olevan korkealaatuisista vertaisarvioiduista lähteistä.

Tuloksista voidaan huomata koneoppimisen merkityksellisyyden ohjelmistotestaukselle. Koneoppimisen integraatio tuo niin teknillisiä kuin organisaation kulttuuriin ja toimintatapoihin vaikuttavia mahdollisuuksia ja haasteita. Mahdollisuuksia ovat niin automatisoidumpi ohjelmistotestaus, ennustavaa analytiikan hyödyntäminen, tehostettu testausprosessi, kuin poikkeavuuksia onnistuneempi havaitseminen. Haasteiksi tuloksissa tunnistettiin hyödynnetyn datan laatu ja määrä, ohjelmistotestauksen tulosten tulkittavuus ja monimutkaisuus, turvallisuus- ja tietosuojaongelmat, henkilöstön taitovaje, resurssien saatavuus, sekä eettiset näkökohdat. Tutkielman tuloksien perusteella voidaan tehdä johtopäätös, että koneoppimisen merkitys ohjelmistotestaukselle organisaatioissa on merkittävä, tuoden mukanaan niin hyviä kuin huonojakin puolia sekä teknillisesti että organisaatioon suoraan vaikuttavasti.

Tutkielman rakenne on seuraava: johdannon jälkeen tutkielman toisessa luvussa määritellään koneoppiminen, sekä käydään läpi sen menetelmiä ja

organisatorista merkitystä. Kolmannessa luvussa syvennytään ohjelmistotestaukseen niin määritelmän, ohjelmistotestausstrategioiden kuin ohjelmistokehityksellisen näkökulman kautta. Neljännessä luvussa vastataan tutkimuskysymykseen ja käydään läpi koneoppimisen integroinnin mahdollisuudet sekä haasteet ohjelmistotestaukselle organisatorisesta näkökulmasta. Viidennessä luvussa esitellään tutkielman johtopäätökset ja pohditaan jatkotutkimusaiheita.

2 KONEOPPIMINEN

Tässä luvussa käsitellään koneoppimista. Kohdassa 2.1 määritellään ytimekkäästi koneoppiminen, minkä jälkeen kohdassa 2.2 tarkastellaan siihen kuuluvia menetelmiä. Tämän jälkeen kohdassa 2.3 käsitellään koneoppimista organisaatioiden näkökulmasta.

2.1 Koneoppimisen määritelmä

Koneoppiminen (*eng. Machine learning, ML*) on yksi tekoälyn osa-alueista. Sen avulla havaitaan automaattisesti merkityksellisiä tietoja tai kuvioita datasta, joka sittemmin hyödyntää näitä tulevien tietojen tai kuvioiden ennustamiseen (Alpaydin, 2014). Koneoppiminen on älykäs järjestelmä, jolla on kyky oppia ja sopeutua, säästäen järjestelmän suunnittelijan aikaa antamalla tälle mahdollisuuden suorittaa tarvittavat muutokset ilman suurta vaivaa (Fulkerson, 1995). Koneoppimisen avulla tietokoneet oppivat datasta ja parantavat suorituskykyään ilman, että niitä suoranaisesti ohjelmoidaan tiettyä tehtävää varten (Murphy, 2021). Koneoppimisen avulla pystytään rakentamaan automaattisia toimintojen sarjoja jonkin tietyn ongelman ratkaisemiseksi, joko ihmisen osallistumisen tai osallistumattomuuden avulla (Sandeep ym., 2022). Koneoppimisen päätelmissä hyödynnetään tilastotieteen teorioita ja matemaattisia malleja. Niiden avulla koneoppiminen ennustaa tai tekee päätöksiä, testaa hypoteeseja tai tekee oivalluksia joko ennalta tuntemattomasta tai tulevasta datasta (Toreini ym., 2020).

Koneoppimista sovelletaan monin eri tavoin. Esimerkiksi hakukoneet, roskapostin torjuntaohjelmat, digitaalikamerat, sekä autojen onnettomuuksien ehkäisyjärjestelmät perustuvat koneoppimiseen. Koneoppimista pidetään ratkaisevassa asemassa, kun kyse on tietokoneiden tekemistä ennusteista, tietojen luokittelusta, ja tarpeesta sopeutua uusiin muuttuviin olosuhteisiin. Koneoppimismenetelmien käyttö ja sovellukset muuttuvat jatkuvasti teknologian kehityksen myötä, mikä osoittaa koneoppimisen monipuolisuuden tekoälytyökaluna. (Shalev-Shwartz & Ben-David, 2014.)

2.2 Koneoppimismenetelmät

Koneoppimismenetelmät jaetaan perinteisesti neljään kategoriaan: ohjattuun, ohjaamattomaan ja puoliohjattuun oppimiseen sekä vahvistusoppimiseen. Se, minkä koneoppimisen menetelmän valitsee, riippuu halutusta tuloksesta. Koneoppimismenetelmillä on omat mallinsa, jotka on määritelty tietyille parametreille. Näiden parametrien optimoimiseksi koneoppiminen hyödyntää sekä harjoitusdataa että tietokoneohjelman aiempaa kokemusta oppiakseen hyödyntämään uusia ja erilaisia malleja. (Murphy, 2021.)

Ohjatussa oppimisessa opitaan yhdistelemään tietoja tai malleja syötteistä x tuotoksiin y. Ohjatun oppimisen tavoite on selvä – siinä halutaan oppia luokittelu, joka pystyy ennustamaan tulevien esimerkkien merkinnät mahdollisimman tarkasti. Lisäksi sen avulla voidaan arvioida onnistumista tai hypoteesien riskiä käyttämällä merkittyjä harjoitusaineistoja laskemalla empiirinen tappio. Esimerkiksi roskapostin havaitsemisjärjestelmä hyödyntää ohjattua oppimista, jonka avulla se merkitsee saapuvan sähköpostin joko roskapostiksi tai muuksi kuin roskapostiksi. (Shalev-Shwartz & Ben-David, 2014.)

Ohjaamattomalla oppimisella pyritään löytämään tietoa tai kuvioita datasta kertomatta järjestelmälle, mitä etsiä, ja käyttämättä virhelaskureita. Ohjaamattomassa oppimisessa hyödynnetään usein kahta menetelmää: tiheyden estimointia ja klusterointia. Tiheyden estimointi soveltuu erityisesti todennäköisyystiheyksien arviointiin, sillä sen avulla ymmärretään esimerkiksi syöttötietojen kuvioiden jakaumaa. Klusteroinnissa tunnistetaan ryhmiä tai klustereita syötetystä datasta. Esimerkiksi organisaatioiden asiakassegmentoinnissa hyödynnetään klusterointia, jossa samantyyppiset asiakkaat ryhmitellään samoihin luokkiin. Klusterointi auttaa myös havaitsemaan poikkeamia korostamalla sellaisia yksiköitä tai keskittymiä, jotka poikkeavat merkittävästi tunnistetuista ryhmistä. (Alpaydin, 2014.)

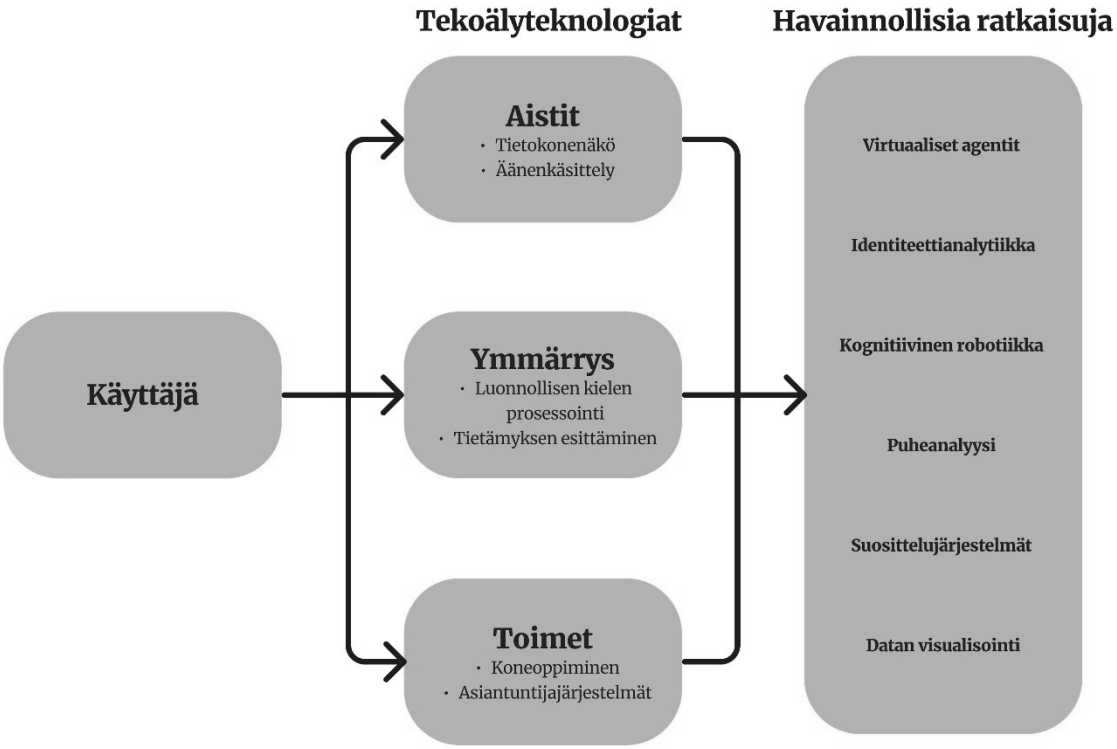
Puoliohjatussa oppimisessa yhdistyvät sekä ohjattu että ohjaamaton oppiminen. Siinä hahmontunnistuksen suorittamiseen käytetään merkkaamatonta dataa yhdessä merkityn datan kanssa. Puoliohjatun oppimisen ydinajatuksena on hyödyntää datajakauman mallioletuksia, jotta voidaan luoda oppija, joka kykenee merkitsemättömien näytteiden merkitsemiseen. Se tarjoaa käytännöllisen lähestymistavan hahmontunnistusvalmiuksien parantamiseen hyödyntämällä merkitsemättömän datan runsautta samalla kun käytettävissä oleva rajallinen merkitty data otetaan huomioon. Menetelmällä on erilaisia sovelluksia monilla aloilla aina tietoverkoista avaruuteen liittyviin aloihin saakka, osoittaen sen mukautuvuuden ja hyödyllisyyden eri tilanteissa ja toimialoilla. (Dang ym., 2022.)

Vahvistusoppimista hyödynnetään tilanteissa, joissa halutusta käyttäytymisestä ei ole saatavilla esimerkkejä, mutta käyttäytymistä on mahdollista arvioida jonkin suorituskriteerin perusteella. Siinä yhdistetään haku ja pitkäkestoinen muisti päätöksenteon tehokkuuden parantamiseksi. Lisäksi vahvistusoppiminen ulottuu myös peräkkäisiin päätöksenteko-ongelmiin, joissa toimia tehdään ajan kuluessa. Menetelmällä on monenlaisia sovelluksia ja laajennuksia,

erityisesti peräkkäisissä päätöksenteko-ongelmissa, joissa se on erinomainen optimoimaan toimia ajan myötä. (Barto & Dietterich, 2004.)

2.3 Koneoppimisen organisatorinen merkitys

Tekoälyteknologioita, kuten koneoppimista, voidaan hyödyntää organisaatioissa laajalti (kuvio 1). Niitä hyödynnetään etenkin, kun kyseessä on monimutkainen tai mukautuvuutta vaativa ongelma tai tarve. Koneoppiminen on ideaalinen etenkin laajojen tehtäväkokonaisuuksien hallintaan, esimerkiksi sähköiseen kaupankäyntiin, joka vaatii suuren määrän digitaalisesti tallennettuja tietoja (Shalev-Shwartz & Ben-David, 2014).



KUVIO 1 Tekoälyteknologioiden hyödyntäminen liiketoiminnassa (mukaillen Sandeep ym., 2022, s. 3).

Koneoppimisen käyttöönotolla liiketoiminnan eri osa-alueilla on merkittäviä organisatorisia vaikutuksia erityisesti riskienhallintaan, toiminnan tehokkuuteen ja työntekijöiden taitojen parantamiseen. Koneoppimispohjainen riskinhallintajärjestelmä esimerkiksi parantaa luottoriskien seurantaa, tunnistaa kuluttajien maksukyvyttömyyden varhaiset varoitusmerkit, ja arvioi dynaamisesti asiakkaiden riskiprofiileja. Arvioimalla likviditeettiriskiä sekä moniulotteisia riski- ja altistumistietoja koneoppimismenetelmien avulla pystytään arvioimaan ja hahmottamaan mahdollisia markkinariskejä. Koneoppimisen avulla pystytään myös

arvioimaan paremmin asiakkaiden sisäänottoa ja torjumaan petoksia systematisoimalla sisäänottoja sekä valvomalla verkkohyökkäyksiä ja mahdollista rahanpesua. (Sandeep ym., 2022.)

Organisaatioissa koneoppimisteknologioiden käyttöönottoa voivat hidastaa useat eri tekijät. Filipe ja muut (2023) nostavat esiin suurimmiksi koneoppimisen integroimattomuuden syiksi epävarmuuden, johdon varauksellisuuden, koneoppimisasiantuntemuksen puutteen, sekä organisaation epäyhtenäiset tietotekniikkaympäristöt. Heidän mukaansa organisaatiot, joilla on kehittyneet teknologia-alustat eivät todennäköisemmin havaitse koneoppimisen tuomia hyötyjä tai ne voivat pysähtyä koneoppimisteknologian käyttöönoton alkuvaiheisiin. Vaikkei organisaatio olisi riittävästi valmistautunut, myös ulkoiset paineet liiketoimintaympäristössä voivat johtaa hätäiseen koneoppimisen käyttöönottoon (Filipe ym., 2023).

Koneoppimisen organisatorinen merkitys on kiistanalainen aihe, ja näkemyksiä on monia riippuen siitä, mitä lähdettä tarkastellaan. Nousevat käyttöönotto kustannukset nousevat vahvasti esiin yhtenä syynä olla integroimatta koneoppimisteknologioita. Esimerkiksi ainoastaan vuonna 2019 tekoälyteknologioiden kustannukset nousivat Euroopassa 49 prosenttiyksikköä edellisvuoteen verrattuna (Dwivedi ym., 2021). Tekoälyteknologioiden, kuten koneoppimisen, integrointi voi vaikuttaa työpaikkoihin. Se, että tekoälyteknologiat voivat korvata monia sääntöihin perustuvia ja toistuvia tehtäviä, tarkoittaa, että tulevaisuudessa voidaan todennäköisesti menettää huomattava määrä työpaikkoja (Dwivedi ym., 2021). Esimerkiksi Manyikanin ja muiden (2017) mukaan jopa kolmannes nykyisistä työpaikoista voi muuttua ja uudistua vuoteen 2030 mennessä tekoälyteknologian integroinnin seurauksena.

Toisaalta on hyvä huomioida tekoälyteknologioihin, kuten koneoppimiseen, liittyvät suuret kassavirrat. Koneoppimista hyödyntävät organisaatiot ovat maailmanlaajuisesti houkutteleva sijoituskohde. Tekoälyteknologioiden, kuten koneoppimisen, myötä organisaatioissa voidaan täydentää tai mahdollisesti jopa korvata inhimillisiä tehtäviä, tarjoten merkittäviä innovaatiomahdollisuuksia. Tekoälyyn perustuvien startup-yritysten investoinnit kasvoivat vuodesta 2010 vuoteen 2020 mennessä keskimäärin 50 prosenttiyksikön kasvuvauhtia. Tekoälyteknologian arvioidaankin tuottavan tulevaisuudessa jopa miljardien dollarien voitot ainoastaan yhdessä vuodessa. (Toosi ym., 2021.)

3 OHJELMISTOTESTAUS

Tämä luku käsittelee ohjelmistotestausta. Luvussa 3.1 määritellään ytimekkäästi ohjelmistotestaus, minkä jälkeen luvussa 3.2 käydään läpi eri ohjelmistotestausstrategioita, sekä niihin liittyviä testausmenetelmiä. Osion lopussa luvussa 3.3. tarkastellaan ohjelmistotestauksen merkitystä ohjelmistokehityksen näkökulmasta.

3.1 Ohjelmistotestauksen määritelmä

Ohjelmistotestaus (*eng. software testing*) on laadunvarmistusta. Se on yksi ohjelmistokehityksen vaiheista, jonka tehtävänä on testata kehitetyn ohjelmiston toimivuutta ja laatua. Ohjelmistotestauksessa varmistetaan, että kehitetty ohjelmisto tekee sen mitä se on suunniteltu tekemään. Siinä ohjelmistoa tai sovellusta testataan oletettujen ja saatujen tulosten välisten erojen havaitsemiseksi. Ohjelmistotestauksessa etsitään ja todennetaan ohjelmistovirheitä. Tästä syystä testattava järjestelmä testataan usein siinä uskossa, että järjestelmässä on virheitä. (Myers ym., 2012.)

Ohjelmistotestaus vaatii monenlaisia resursseja työn tarkkuuden ja kriittisyyden vuoksi. Sitä suorittavat joko yleensä siihen erikoistuneet testausryhmät tai joissakin tapauksissa ohjelmistokehittäjät itse. Sekä ohjelmistojen monitasoisten rakenteiden että tiukkojen laadullisten vaatimusten vuoksi ohjelmistotestaus vaatii testaajalta paljon resursseja, kuten työtunteja ja erilaisten testausmenetelmien hyödyntämistä. Ohjelmistotestauksessa hyödynnetäänkin erilaisia testausstrategioita sekä menetelmiä työn sujuvoittamiseksi, mitä tarkastellaan tarkemmin seuraavassa luvussa. (Shete & Jadhav, 2014.)

3.2 Ohjelmistotestausstrategiat

Testausstrategioilla hahmotellaan ohjelmistotestauksen lähestymistapaa. Ne muodostuvat useista vaiheista, joiden avulla varmistetaan ohjelmistotestauksen onnistuminen. Keskeisimpiä testausstrategioita ovat: yksikkö-, integrointi-, järjestelmä- ja hyväksymistestaus. (Durelli ym., 2019.)

Yksikkötestauksessa keskitytään testaamaan ohjelmistosovellusten perusyksikköjä, kuten moduuleja, menetelmiä tai komponentteja. Kehittäjät luovat testejä ja kehittävät sovelluskoodia testitulosten perusteella, tehden yksikkötestauksesta jatkuvan testaus- ja tarkistusprosessin. Tässä ohjelmistotestausstrategiassa testausta tehdään niin kauan, kunnes kaikki yksikkötestit ovat hyväksyttyjä. Strategiassa yleensä hyödynnetään valkoinen laatikko -testausmenetelmää. Siinä testaajalla on täydet tiedot testattavasta sovelluksesta, mukaan lukien pääsy lähdekoodiin ja suunnitteludokumentteihin. Valkoinen laatikko - testauksessa testit johdetaan järjestelmän ja sen haarojen, yksittäisten ehtojen ja lausekkeiden sisäisistä ominaisuuksista. Sen avulla testataan esimerkiksi järjestelmän tiedonkulkua, ohjausvirtaa, polkuja, haaroja ja lausekkeiden kattavuutta. Valkoinen laatikko -testausmenetelmä vaatii syvällistä koodin tuntemusta, joten se ei ole kovin käytännöllinen laajamittaiseen käyttöön. Se on kuitenkin testausmenetelmänä erittäin tuottava vikojen ja virheiden havaitsemisessa sekä niiden aiheuttamien ongelmien ratkaisemisessa. (Ammann, & Offutt, 2016.)

Integrointitestauksessa testataan useita moduuleja, ja se suoritetaan yleensä yksikkötestauksen jälkeen. Siinä on tavoitteena varmistua siitä, miten kaikki ohjelmistomoduulit toimivat, kun ne yhdistetään suuremmassa mittakaavassa esimerkiksi erilaisissa rakenteissa ja rajapinnoissa. Integrointitestauksessa hyödynnetään niin valkoinen kuin musta laatikko -testausmenetelmiä. Musta laatikko -testauksessa testit johdetaan ohjelmiston ulkoisista kuvauksista, kuten määrittelyistä, vaatimuksista ja suunnittelusta. Musta laatikko -menetelmässä järjestelmän testaajalla ei ole tietoa koodin sisäisistä yksityiskohdista, mutta tämä on tietoinen vaadituista määrittelyistä ja odotetuista tuloksista. Integraatiotestauksessa siis testataan, miten järjestelmä käyttäytyy eri syötteillä, keskittyen tärkeimpiin vaatimuksiin. (Bharti Nagpal, 2016.)

Järjestelmätestauksessa testataan koko järjestelmä kokonaisuutena. Siinä arvioidaan järjestelmän kokonaislaatua. Järjestelmätestauksessa keskitytään toiminnallisten ominaisuuksien lisäksi muihin ominaisuuksiin, kuten järjestelmän luotettavuuteen, ylläpidettävyyteen, turvallisuuteen ja vaatimusmäärittelyjen noudattamiseen. Kun järjestelmätestauksen testit epäonnistuvat, testattava järjestelmä palautetaan takaisin jatkokehitystä varten. Tässä testausstrategiassa hyödynnetään usein musta laatikko -testausmenetelmää, sillä järjestelmätestauksessa testaajalla ei ole minkäänlaista tietoa testattavan järjestelmän koodista tai sen suunnittelutavasta. Toisin kuin integrointitestauksessa järjestelmätestauksessa musta laatikko -testausmenetelmän avulla arvioidaan koko järjestelmän toimivuutta, mukaan lukien sekä toiminnallisia että muita näkökohtia. (Sneha & Malle, 2017.)

Hyväksymistestaus on viimeinen vaihe ennen järjestelmän luovuttamista tai julkaisua loppukäyttäjille. Siinä varmistutaan, että järjestelmä toimii odotetulla tavalla, ja virheiden tunnistamisen sijaan asetetaan etusijalle asianmukaisen toiminnan varmistaminen. Myös hyväksymistestauksessa hyödynnetään pääsääntöisesti musta laatikko -testausta, jonka avulla varmistetaan, että ohjelmisto täyttää hyväksymiskriteerit ja on valmis käyttöönotettavaksi. (Sneha & Malle, 2017.)

3.3 Ohjelmistotestauksen merkitys ohjelmistokehityksessä

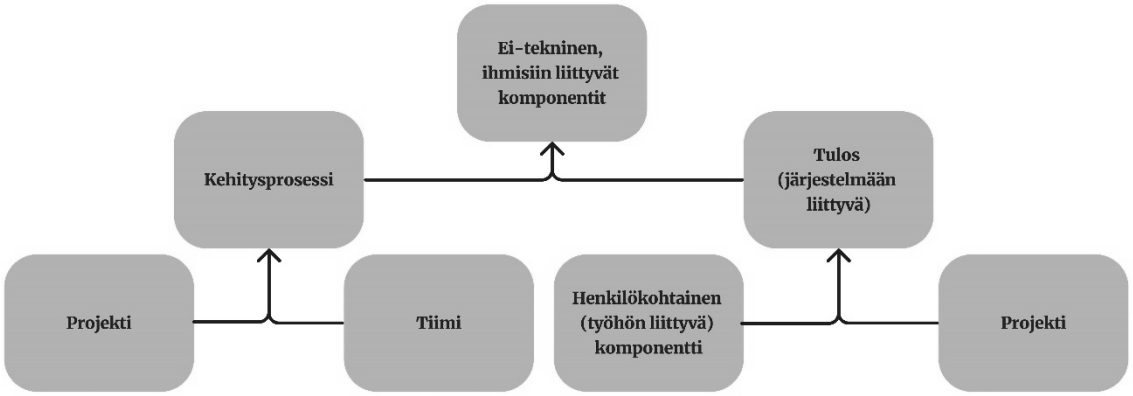
Ohjelmistotestaus on yhä kriittisempää modernissa ohjelmistokehityksessä. Vaikka luotettavan ohjelmiston suunnitteluun vaikuttavat monet tekijät, testaus on ensisijainen tapa, jolla ohjelmistoa arvioidaan. Testivetoiselle kehitykselle on suuri kysyntä, mikä lisää painetta ja tarvetta ohjelmistotestaukselle. (Ammann & Offutt, 2016.)

Testaustoiminta on erittäin tärkeää ohjelmistokehityksen elinkaaren aikana. Se tarjoaa objektiivisen ja riippumattoman näkemyksen ohjelmistosta, jotta liiketoiminnassa voidaan arvioida ja ymmärtää ohjelmistoon liittyviä riskejä ennen ohjelmiston julkistamista. Ohjelmistotestaus nähdäänkin loppupeleissä prosessina, jossa kelpuutetaan ja todennetaan, että ohjelmisto täyttää liiketoiminnalliset ja tekniset vaatimukset, jotka ovat ohjanneet sen suunnittelua ja kehittämistä. (Shete & Jadhav, 2014).

Jos ohjelmistotestausta ei tehdä kunnolla, ohjelmiston laatu voi heikentyä tai siihen voi jäädä virheitä. Nämä voivat johtaa siihen, että ohjelmisto ei vastaa asiakkaiden odotuksia ja vaatimuksia (Trzeciak ym., 2022). Kaikkien virheiden löytäminen on kuitenkin epäkäytännöllistä ja usein mahdotonta. Vergilionin ja muiden (2006) mukaan syötteiden suuri koko ja eri skenaarioiden yhdistelmien suuri määrä voivat esimerkiksi tehdä kattavasta testauksesta epätarkoituksenmukaista. Siksi testaajien on kehitettävä jokin testien riittävyyden standardi, jonka avulla he voivat päättää, milloin ohjelmisto on testattu riittävän perusteellisesti. Myersin ja muiden (2011) mukaan tämä edellyttää esimerkiksi testitulosten huolellista tarkastelua.

Testaustoimintaan investoidaan merkittävästi resursseja. Korkeisiin kustannuksiin vaikuttavat erilaiset resurssit, kuten aika, rahoitus ja henkilöstö (Gillenson ym., 2018). Jopa yli 50 prosenttiyksikköä ohjelmistokehityksen kustannuksista kohdistuu ainoastaan ohjelmistotestaukseen (Myers ym., 2011). Ohjelmistotestauksen arvo organisaatioissa määritellään sen varmistamisena, että jokin ohjelmistoratkaisu täyttää sille asetetut arvotavoitteet ja järjestää testaustehtävät niin, että ne toimivat investointitoimintana (Boehm & Huang, 2003). Ohjelmistotestauksella katsotaan olevan epäsuora vaikutus kehitetyn tuotteen arvoon, mikä ei aina vastaa sille kohdennettuja resursseja. Sheten ja Jadhavinin (2014) mukaan on todennäköistä, että testaukseen käytettävä osuus ohjelmistokehityskustannuksista voi jopa kasvaa tulevaisuudessa, ellei löydetä keinoja tehokkaampaan ohjelmistotestaukseen.

Ohjelmistokehityksen onnistumiseen vaikuttavilla tekijöillä, kuten kehitysprosessilla ja tuloksilla, on tärkeä rooli ohjelmistokehityksen onnistumisessa (kuvio 2). Onnistunut kehitysprosessi ja lopulta tulokset vaativat muun muassa niin onnistunutta projektin suunnittelua kuin oikeita työntekijöitä (Procaccino ym., 2006). Myös Reel (1999) korostaa riittävän ammattitaitoisen tiimin merkitystä. Hänen mukaansa ammattitaitoinen kehitystiimi on yksi merkittävimmistä onnistuneen projektin osatekijöistä.



KUVIO 2 Ohjelmistokehityksen menestyksen osatekijöiden malli (mukaillen Procaccino ym., 2006, s. 80).

Tästä voidaan päätellä, että myös ohjelmistotestaukseen vaaditaan ammattitaitoinen testausryhmä tai yksittäinen testaaja. Ammattitaitoisten työntekijöiden avulla pystytään toimittamaan ajoissa ja kohtuuhintaan hyvin testattu järjestelmä. Tämä myös auttaa koko ohjelmistoprojektia erityisesti varhaisessa vaiheessa vaatimusten ymmärtämisessä, säästäen resursseja sekä välttämällä mahdolliset kalliit ja aikaa vievät jälkityöt. (Procaccino ym., 2006.)

Kyseiset havainnot vahvistavat huomiota ohjelmistotestauksen merkityksellisyydestä ohjelmistokehitysprosessissa. Ne tuovat esiin havainnollistavien esimerkkien avulla, kuinka eri ohjelmistokehityksen osatekijät ovat kytkeytyneet toisiinsa, ja minkälainen merkitys ohjelmistotestauksella on niissä. Testaus mahdollistaa onnistuneemman ohjelmistokehityksen, minkä vuoksi sen merkitys on kasvussa muun muassa moderneissa ohjelmistokehitys projekteissa. Tämä myös korostaa huomiota testauksen kriittisyydestä ja sitä, kuinka ohjelmistokehityksen merkitys on jatkuvassa muutoksessa.

4 KONEOPPIMISEN MERKITYS OHJELMISTOTESTAUKSESSA ORGANISAATIOKONTEKSTISSA

Luvussa käsitellään koneoppimisen integroinnin tuomia mahdollisuuksia ja haasteita ohjelmistotestaukselle. Luvussa 4.1 tarkastellaan koneoppimisen mahdollisuuksia ja luvussa 4.2 sen haasteita. Tässä luvussa pyritään vastaamaan tutkimuskysymykseen eli minkälaisia mahdollisuuksia ja haasteita koneoppimisen integroimisesta ohjelmistotestaukselle on organisaatioiden näkökulmasta.

4.1 Mahdollisuudet

Koneoppimisen integrointi ohjelmistotestauksessa tuo mukanaan monia mahdollisuuksia. Kyseiset mahdollisuudet liittyvät niin itse testausstrategioihin kuin koko ohjelmistokehityksen kaareen organisatorisesta näkökulmasta tarkasteltaessa. Integroinnin myötä pystytään automatisoimaan ohjelmistotestausta, hyödyntämään ennustavaa analytiikkaa, tehostamaan testausprosesseja, sekä havaitsemaan onnistuneemmin poikkeavuuksia (taulukko 1).

TAULUKKO 1 Koneoppimisen mahdollisuudet ohjelmistotestaukselle organisaatiokontekstissa.

Koneoppimisen mahdollisuudet ohjelmistotestauksessa organisaatiokontekstissa	**Lähteet**
Ohjelmistotestauksen automatisointi	Durelli ym., 2019; Wirtz ym., 2018
Ennustavan analytiikan hyödyntäminen	Sandeep ym., 2022; Wirtz ym., 2018; Zheng ym., 2018
Tehostettu testausprosessi	Durelli ym., 2019; Stocco, 2019; Zhang & Tsai, 2003
Onnistuneempi poikkeavuuksien havaitseminen	Boehm & Huang, 2003; Gillenson ym., 2018; Mighetti & Hadad, 2016; Richardson ym., 2012

Koneoppimisen avulla voidaan automatisoida ohjelmistotestausta. Durellin ja muiden (2019) mukaan koneoppimisen algoritmeja hyödynnetään pääasiassa testitapausten luomisessa sekä niiden automatisoimisessa. Heidän mukaansa automatisoinnin avulla pyritään vähentämään testauksen monimutkaisuutta ja kustannuksia, etenkin testattaessa järjestelmiä, joiden testaaminen on usein vaikeaa ja kallista. Wirtzin ja muiden (2019) mukaan tekoälyteknologioilla onkin monenlaisia käyttötarkoituksia organisaatioissa. He nostavat esiin, että prosessien automatisoinnin myötä organisaatioissa vapautuu resursseja, toimintojen tarkkuus paranee ja kustannukset laskevat.

Ennustava analytiikka koneoppimisessa tuo monia mahdollisuuksia. Julkisen sektorin organisaatioissa koneoppimisen integroinnin myötä suuren datamäärän käsittely helpottuu ja tekee siitä ennustettavampaa (Zheng ym., 2018). Tämä voidaan nähdä helpottavan myös ohjelmistotestausta, joka usein vaatii suuren määrän sekä uutta että vanhaa dataa testitapausten suorittamiseksi (Wirtz ym., 2018). Koneoppimisalgoritmien ennakointikyvyn ansiosta voidaan myös ennakoida esimerkiksi työntekijöiden suorituksia, näin mahdollistaen uudenlaisen työntekijöiden arviointitavan (Sandeep ym., 2022). Tämä mahdollisuus voidaan nähdä esimerkiksi auttavan karsimaan ohjelmistotestaukseen liittyviä henkilöstökuluja, jotka syntyvät muun muassa manuaalisen testauksen seurauksena.

Tehokkaampi testausprosessi on mahdollista tekoälyteknologioiden avulla. Durelli ja muut (2019) nostavat esiin havainnon yhä monimutkaisemmista ohjelmistojärjestelmistä. Heidän mukaansa perinteisemmät testaustekniikat eivät enää sovellu kovinkaan hyvin moderneihin usein kompleksisien ohjelmistojärjestelemien testaamiseen. Koneoppimisalgoritmit ovatkin arvokkaita etenkin vaikeasti ymmärretyillä ongelmallisilla alueilla, suurissa tietokannoissa, ja sopeutumista vaativilla alueille (Zhang & Tsai, 2003). Stocco (2019) korostaa tekoälyn potentiaalia ohjelmistosuunnittelun haasteiden ratkaisemisessa ja uskoo, että koneoppimisen hyödyntäminen testauksessa on kasvava trendi. Hän myös nostaa esiin kehittäjän roolin ohjelmistokehityksessä sekä -testauksessa. Stoccon (2019) mukaan ohjelmistokehityksessä ohjelmistokehittäjiltä jopa odotetaan tekoälyyn perustuvien työkalujen integrointia ja hyödyntämistä.

Myös poikkeavuuksien havaitseminen ohjelmistotestauksessa tarjoaa monia mahdollisuuksia organisatorisesta näkökulmasta. Koska ohjelmistotestaus tapahtuu yleensä viimeisessä vaiheessa ohjelmistokehitystä (Gillenson ym., 2018), on sen päällimmäinen tehtävä varmistua kehitetyn ohjelmiston toimivuudesta ja etsiä poikkeavuuksia (Boehm & Huang, 2003). Richardsonin ja muiden (2012) mukaan suurin osa ohjelmistoprojektin budjetista kuluu kehitystyön ja uhkien valmistelun aikana havaittujen poikkeamien korjaamiseen. Myös poikkeavuuksien havaitsemisen myötä voidaan varmistaa onnistuneempi ohjelmistoprojekti (Mighetti & Hadad, 2016).

Tästä voidaan päätellä, että organisaatiotasolla on havahduttu tekoälyratkaisujen kasvaneeseen kulutukseen ja sen hyödyntämistä tuetaan vahvasti. Tämä korostaa havaintoa, että tekoälyteknologioiden, kuten koneoppimisen avulla voidaan tehostaa ohjelmistokehityksen prosesseja, kuten ohjelmistotestausta. Käsitellyt mahdollisuudet tuovat mukanaan paljon hyötyjä organisaatioihin niin lisääntyneenä teknillisenä osaamisena kuin itse organisaatio kulttuuriin ja toimintatapoihin vaikuttavana hyötynä.

4.2 Haasteet

Koneoppimisen integrointimesta ohjelmistotestauksessa liittyy kuitenkin myös monenlaisia haasteita. Tarkasteltaessa organisatorisesta näkökulmasta haasteet koskevat niin tekoälyteknologioiden käyttöönottoa kuin niiden käyttölaajuutta organisaatioissa. Haasteet liittyvät pääasiassa datan laatuun ja määrään, tulkittavuuteen ja monimutkaisuuteen, turvallisuus- ja tietosuojaongelmiin, taitovajeeseen, resursseihin, sekä eettisiin näkökohtiin (taulukko 2).

TAULUKKO 2 Koneoppimisen haasteet ohjelmistotestaukselle organisaatiokontekstissa.

Koneoppimisen haasteet ohjelmistotestauksessa organisaatiokontekstissa	**Lähteet**
Datan laatu ja määrä	Durelli ym., 2019; Rapps & Weyuker, 1985
Tuloksien tulkittavuus ja monimutkaisuus	Zhang & Tsai, 2003; Zhang ym., 2011; Wan ym., 2019
Turvallisuus- ja tietosuojaongelmat	Bostrom ym., 2020; Lin ym., 2014; Wirtz ym., 2018
Taitovaje	LaToza ym., 2006; Wan ym., 2019
Resurssit	Wan ym., 2019
Eettiset näkökohdat	Wirtz ym., 2018

Haaste testausdatan laadusta ja määrästä näkyy riittämättömänä tai virheellisenä testausdatana. Vaikka koneoppimispohjaiset lähestymistavat skaalautuvat

hyvin ohjelmistotestauksessa vaativat ne suuren määrän laadukasta dataa, jonka puute voi johtaa virheellisiin tuloksiin (Rapps & Weyuker, 1985). Durelli ja muut (2019) nostavatkin esille haasteen asiaankuuluvien tietojen saatavuudesta. Heidän mukaansa, jotta koneoppimisen algoritmeja pystytään hyödyntämään tehokkaasti, edellyttävät ne laadukasta tietoa ja dataa sekä mittavaa harjoitteluaineistoa. Tästä voidaan päätellä, että mikäli ei ole saatavilla suuria määriä laadukasta dataa, ei kattavaa ohjelmistotestausta pystytä suorittamaan kovinkaan onnistuneesti. Tämä voi johtaa joidenkin testitapausten testaamattomuuteen, mikä voi olla riski ohjelmiston laadun arvioinnissa ja validoinnissa (Durelli ym., 2019). Tämän myötä voidaan nähdä, että mikäli testattava ohjelmisto julkaistaan ilman kattavaa testausta voi asiakas tai käyttäjä törmätä mahdollisesti virheisiin, joita ei pystytty validoimiaan ja korjaamaan ennen ohjelmiston julkaisua loppukäyttäjille.

Koneoppimismenetelmiä hyödyntävät ohjelmistotestauksen tulokset voivat olla tulkinnanvaraisia ja monimutkaisia. Ohjelmistotestauksen tulosten tulkitsemin haasteita ovat muun muassa niiden monimutkaisuus, muunneltavuus ja näkymättömyys (Zhang & Tsai, 2003). Testaustuloksia on koneoppimisjärjestelmissä vaikeampi toistaa niiden satunnaisuuden vuoksi (Wan ym., 2019). Lisäksi testaustulosten tulkitseminen voi jäädä usein yhden testaustiimin tai testaajan harteille, jotka ovat voineet jopa itse luoda testatut ohjelmistotestit (Zhang ym., 2011). Näin ollen ohjelmistotestauksen tulosten tulkitseminen ja raportointi voi olla hyvin subjektiivista, joka voi vaikeuttaa tulosten ymmärtämistä muissa ohjelmistoprojektin sidosryhmissä. Tämä voi monimutkaistaa ohjelmistotestaustulosten läpikäyntiä jättäen varaa yksilöiden omille tulkinnoille ja päätelmille (Zhang & Tsai, 2003).

Koneoppimismallit voivat käsitellä arkaluontoisia tietoja aiheuttaen turvallisuus- ja tietosuojariskejä. Tekoälyteknologioiden hallinnan laaja-alaisuuteen kuuluu sekä oikeudellisia että sääntelyyn liittyviä näkökohtia, jotka liittyvät esimerkiksi tietoon, algoritmeihin, infrastruktuuriin ja ihmisten väliseen vuorovaikutukseen (Bostrom ym., 2020). Tekoälyn hallinnointi on usein huomiotta jätetty haaste, vaikka sen tulisi olla nykyaikana entistäkin huomioitu (Wirtz ym., 2018). Esimerkiksi tekoälyjärjestelmät, kuten robottisovellukset ovat erityisen alttiita kyberhyökkäyksille (Lin ym., 2014), nostaen esille tekoälyteknologioiden ja -järjestelmien haavoittuvuuden. Tekoälyteknologioilla, kuten koneoppimisella, voi olla suuriakin vaikutuksia ihmisten yksityisyyteen, niiden keräämän datan vuoksi, minkä takia yksityisyyden suoja voidaan nähdä olevan yksi merkittävimmistä haasteista käsiteltäessä koneoppimista organisatorisessa kontekstissa.

Koneoppimisen kehittäminen organisaatiossa vaatii monenlaisia taitoja, mikä voi aiheuttaa taitovajetta työntekijöissä. Ohjelmointitaitojen lisäksi koneoppimisen kehittäminen vaatii niin matematiikan, tietoteorian kuin tilastotieteen erityisosaamista (LaToza ym., 2006). Tämä voi aiheuttaa haasteita koneoppimisen integroimisessa ja kehittämisessä. Organisaatioilla voi olla pulaa ammattitaitoisista osaajista, jotka pystyvät tehokkaasti suunnittelemaan, toteuttamaan ja ylläpitämään koneoppimispohjaisia testausratkaisuja (Wan ym., 2019). Tämä haaste voidaan nähdä ratkaistavan esimerkiksi henkilöstön koulutuksen ja

uusien työntekijöiden palkkaamisella, mikä kuitenkin vaatii entistä enemmän resursseja jo nyt paljon resursseja kuluttavassa ohjelmistokehityksessä.

Koneoppimismallien käyttöönotto testaukseen voi vaatia huomattavia alkuinvestointeja. Wanin ja muiden (2019) mukaan vajavainen budjetti on yksi suurimmista haasteista, joita organisaatiot kohtaavat tekoälyohjelmia käyttöönotossa. Artikkelissaan he tuovat ilmi, että koneoppimisohjelmistokehityskäytäntöjen operationalisointi ja standardointi on välttämätöntä kustannustehokkaiden ja luotettavien järjestelmien kehittämiseksi organisaatioissa. On kuitenkin hyvä huomioida, ettei se takaa täysin kustannustehokasta käyttöönottoa, mikä luo itsessään jo merkittävän haasteen koneoppimismenetelmien käyttöönotossa ohjelmistotestaukseen.

Eettisten tekoälykäytäntöjen varmistaminen ja ennakkoluuloihin puuttuminen ovat kriittisiä näkökohtia koneoppimisenintegroinnissa. Eettisiä haasteita ovat Wirtzin ja muiden (2018) mukaan muun muassa tekoälyn sääntöjen laatiminen ihmisen käyttäytymistä varten, koneen ja ihmisen arvotuksen yhteensopivuus, moraaliset dilemmat sekä syrjinnän estäminen. Tutkijoiden mukaan huolet esimerkiksi työvoiman korvaamisesta ja tuntemattomista taloudellisista vaikutuksista nostetaan myös keskustelussa usein esille. Tästä nouseekin esiin pohdinta, kuinka eettistä työvoiman korvaaminen tekoälyteknologioilla oikeasti on.

Käsitellyt kohdat tuovat esiin haasteet, jotka voivat vaikuttaa koneoppimisen integrointiin ohjelmistotestauksessa. Käsitellyt haasteet tuovat mukanaan esimerkin mahdollisista haitoista, joita organisaatioiden näkökulmasta voidaan kokea sekä teknillisesti että organisaation toimintatapoihin ja kulttuuriin vaikuttavana. Tekoälyteknologiat, kuten koneoppiminen, tulisivatkin olla jatkuvaan arvioinnin kohteena, jotta niiden käyttöönotto on mahdollisimman onnistunutta organisaatioissa.

5 YHTEENVETO

Meneillään on ajanjakso, jolloin niin yksityishenkilöt kuin yritykset tuottavat valtavia määriä dataa. Suuren datamäärän hallitsemiseksi organisaatiot ovat kääntyneet yhä enemmän tekoälyteknologioiden, kuten koneoppimisen, puoleen. Tekoälyteknologioiden odotetaankin edistävän innovointia ja talouskasvua organisaatioissa. Kuitenkin myös huoli työpaikkojen siirtymisestä ja tuntemattomista taloudellisista vaikutuksista nostetaan usein asiayhteydessä esille. Ohjelmistoteollisuus on kasvanut merkittävästi, ja siitä on tullut entistä kilpailukykyisempi ja kattavampi eri aloilla. Tämän vuoksi koneoppimisen soveltaminen on herättänyt kasvavaa kiinnostusta automatisoida erilaisia ohjelmistosuunnittelutoimintoja muun muassa toivona säästää resursseja ja toimittaa arvokkaampia ohjelmistoja.

Tehokkaampia ohjelmistotestausmenetelmiä etsitään jatkuvasti. Ohjelmistotestaus on tärkeä prosessi, joka suoritetaan laadunvarmistuksen tukemiseksi keräämällä tietoa tutkittavan ohjelmiston luonteesta. Sitä pidetään ensisijaisena keinona varmistaa ohjelmiston luotettavuus ja täyttää toiminnalliset vaatimukset. Ohjelmistokehityksessä onkin meneillään vallankumous, jossa uudelleen määritellään testauksen roolia ohjelmistotuotteiden menestyksen kannalta. Tässä kandidaatintutkielmassa perehdyttiin koneoppimisen merkitykseen ohjelmistotestauksessa. Tutkimuksen tavoitteena oli löytää koneoppimisen tuomat mahdollisuudet ja haitat ohjelmistotestaukselle organisaatiokontekstissa.

Tutkielma toteutettiin systemaattisena kirjallisuuskatsauksena. Tutkielmassa hyödynnettiin pääsääntöisesti tieteellisiä artikkeleita ja konferenssipapereita, mutta mukaan mahtui myös tieteellisiä kirjoja aiheeseen liittyvän korkealaatuisten artikkeleiden puutteen takia. Kirjallisuus kerättiin tietokannoista, kuten IEEE Xplore, JYKDOK, Google Scholar, ScienceDirect ja Scopus. Hakuprosessissa hyödynnettiin muun muassa seuraavia hakusanoja: *artificial intelligence*, *machine learning*, *software development*, ja *software testing*. Rajallisen suomenkielisen aineiston saatavuuden vuoksi tutkielmassa hyödynnettiin ainoastaan englanninkielistä kirjallisuutta. Kirjallisuutta kerättäessä huomioitiin niin viittausten lukumäärä kuin sen vaikuttavuus alalla. Lisäksi tutkimuskirjallisuuden tuli olla merkityksellistä ja pystyä auttamaan sekä tukemaan tutkimuskysymykseen

vastaamisessa. Kirjallisuutta kerättäessä huomioitiin Julkaisufoorumin antama luokitus. Tutkielmassa hyödynnettiin ainoastaan kirjallisuutta, joka on täyttänyt vähintään Julkaisufoorumin tason yksi kriteerit.

Tutkimuskysymys, johon tutkielmassa pyrittiin vastaamaan, oli: ”*Minkälaisia ovat mahdollisuudet ja haasteet koneoppimisen integroimisesta ohjelmistotestaukselle organisaatioiden näkökulmasta*?”. Lähdekirjallisuus osoitti, että koneoppimisen integrointi ohjelmistotestaukseen tuo paljon mahdollisuuksia, mutta myös haasteita organisatorisesta kontekstista tarkasteltaessa. Mahdollisuudet koneoppimisen integroinnista ohjelmistotestaukseen organisatorisesta näkökulmasta liittyivät etenkin toimivampaan ja kustannustehokkaampaan ohjelmistokehitysprosessiin. Koneoppimisen integroinnin mahdollisuuksiksi tunnistettiin esimerkiksi automatisoidumpi ohjelmistotestaus, ennustavan analytiikan hyödyt, sekä poikkeavuuksien parempi tunnistus. Tunnistetut haasteet koskivat koneoppimisen integroinnin monimutkaisuutta, esimerkiksi ottamalla huomioon integroimiseen ja ohjelmistotestaukseen liittyviä konsepteja, kuten datan laadun ja määrän, turvallisuus- ja tietosuojaongelmat, resurssit, sekä eettiset näkökohdat.

Tulosten perusteella koneoppimisella on merkittävä merkitys ohjelmistotestaukselle esimerkiksi testausprosessien parantamisena ja tehokkuuden lisäämisenä. Samalla on kuitenkin tärkeää kiinnittää huomiota integrointiin liittyviin haasteisiin ja varmistaa, että esimerkiksi turvallisuus- ja tietosuojaongelmat sekä eettiset kysymykset otetaan asianmukaisesti huomioon. On myös hyvä huomioida tekoälyteknologioiden, kuten koneoppimisen, kiistanalaiset näkemykset sen hyödyntämisestä organisaatioissa. Jotkut osapuolet näkevät koneoppimisen hyödyntämisen hyvinkin positiivisesti, esimerkiksi suurien investointien ja kustannustehokkaamman työnteon ansiosta. Kuitenkin toiset näkevät koneoppimisen hyödyntämisen riskinä tuoden esille esimerkiksi mahdollisten työpaikkojen menetyksen sekä tietosuojariskit. Siksi tekoälyteknologiat, kuten koneoppiminen, tulisi olla jatkuvan arvioinnin kohteena, jotta organisaatioissa voidaan toteuttaa ja varmistaa sen mahdollisimman onnistunut käyttöönotto.

Jatkotutkimukset ovat tarpeen aiheelle, sillä edelleen on tarve tutkia syvällisemmin koneoppimisen käyttöä ohjelmistotestauksessa organisaatioympäristöissä. Jatkotutkimukset voisivat kohdistua siihen, miten koneoppimisen integrointi ohjelmistotestaukseen toteutuu käytännössä. Organisaatioympäristöä voitaisiin myös ottaa kattavammin mukaan tutkimukseen selvittämällä esimerkiksi tapaustutkimuksen avulla integroinnin merkitystä ohjelmistotestaukseen spesifissä organisaatiossa. Myös ohjelmistotestauksen organisatorista merkitystä olisi hyvä eritellä lisää, sillä materiaalia ei tästä paljon löytynyt.

LÄHTEET

Alahyari, H., Berntsson Svensson, R., & Gorschek, T. (2017). A study of value in Agile Software Development Organizations. *Journal of Systems and Software*, *125*, 271–288. https://doi.org/10.1016/j.jss.2016.12.007

Alpaydin, E. (2014). *Introduction to Machine Learning* (3. uud. painos). London, England: The MIT Press, 1–46.

Ammann, P., & Offutt, J. (2016). *Introduction to Software Testing* (2. uud. painos). Cambridge: Cambridge University Press, 1-39. https://doi.org/10.1017/9781316771273

Barto, A. & Dietterich, T. (2004). *Reinforcement learning and its relationship to supervised learning*. Handbook of Learning and Approximate Dynamic Programming, 47–64.

Bharti Nagpal, S. G. (2016). Descriptive Study of Software Testing & Testing Tools. *International Journal of Innovative Research in Computer and Communication Engineering*, *4*(6), 10288–10295. https://doi.org/10.15680/ijircce.2016.0406006

Boehm, B., & Huang, L. G. (2003). Value-based software engineering: A case study. *Computer*, *36*(3), 33–41. https://doi.org/10.1109/mc.2003.1185215

Bostrom, N., Dafoe, A., & Flynn, C. (2020). Public policy and Superintelligent AI. *Ethics of Artificial Intelligence*, 293–326. https://doi.org/10.1093/oso/9780190905033.003.0011

Chan, W. K., Ho, J. C., & Tse, T. H. (2009). Finding failures from passed test cases: Improving the pattern classification approach to the testing of mesh simplification programs. *Software Testing, Verification and Reliability*, *20*(2), 89–120. https://doi.org/10.1002/stvr.408

Dang, W., Guo, J., Liu, M., Liu, S., Yang, B., Yin, L., & Zheng, W. (2022). A semi-supervised Extreme Learning Machine algorithm based on the new weighted kernel for Machine Smell. *Applied Sciences*, *12*(18), 9213. https://doi.org/10.3390/app12189213

Durelli, V. H., Durelli, R. S., Borges, S. S., Endo, A. T., Eler, M. M., Dias, D. R., & Guimaraes, M. P. (2019). Machine learning applied to software testing: A systematic mapping study. *IEEE Transactions on Reliability*, *68*(3), 1189–1212. https://doi.org/10.1109/tr.2019.2892517

Dwivedi, Y. K., Hughes, L., Ismagilova, E., Aarts, G., Coombs, C., Crick, T., Duan, Y., Dwivedi, R., Edwards, J., Eirug, A., Galanos, V., Ilavarasan, P.

V., Janssen, M., Jones, P., Kar, A. K., Kizgin, H., Kronemann, B., Lal, B., Lucini, B., … Williams, M. D. (2021). Artificial Intelligence (AI): Multidisciplinary perspectives on emerging challenges, opportunities, and agenda for research, practice and policy. *International Journal of Information Management*, *57*, 101994. https://doi.org/10.1016/j.ijinfomgt.2019.08.002

Filipe, P., Ruivo, P., & Oliveira, T. (2023). Assessing machine learning adoption at the firm level: The moderating effect of the environmental context. *Procedia Computer Science*, *219*, 1034–1042. https://doi.org/10.1016/j.procs.2023.01.381

Fulkerson, B. (1995). Machine Learning, neural and statistical classification. *Technometrics*, *37*(4), 459–459. https://doi.org/10.1080/00401706.1995.10484383

Gillenson, M. L., Zhang, X., Stafford, T. F., & Shi, Y. (2018). A literature review of software test cases and future research. *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. https://doi.org/10.1109/issrew.2018.00015

Jadhav, A., Kaur, M., & Akter, F. (2022). Evolution of software development effort and cost estimation techniques: Five decades study using automated text mining approach. *Mathematical Problems in Engineering*, *2022*, 1–17. https://doi.org/10.1155/2022/5782587

LaToza, T. D., Venolia, G., & DeLine, R. (2006). Maintaining mental models. *Proceedings of the 28th International Conference on Software Engineering*. https://doi.org/10.1145/1134285.1134355

Lin, P., Abney, K., & Bekey, G. A. (2014). *Robot ethics: The Ethical and Social Implications of Robotics*. Cambridge, Massachusetts: The MIT Press, 85-90.

Manyika, J., Lund, S., Chui, M., Bughin, J., Woetzel, J., Batra, P., Ko, R., & Sanghvi, S. (2017). *Jobs lost, jobs gained: Workforce transitions in a time of automation.* McKinsey Global Institute, 4-5.

Mighetti, J. P., & Hadad, G. D. (2016). A requirements engineering process adapted to Global Software Development. *CLEI Electronic Journal*. https://doi.org/10.19153/cleiej.19.3.7

Mohagheghi, P., Dehlen, V., & Neple, T. (2009). Definitions and approaches to model quality in model-based software development – a review of literature. *Information and Software Technology*, *51*(12), 1646–1669. https://doi.org/10.1016/j.infsof.2009.04.004

Murphy, K. P. (2021). *Machine learning: A Probabilistic Perspective*. Cambridge, Massachusetts: The MIT Press, 1-33.

Myers, G. J., Sandler, C., & Badgett, T. (2012). *The art of software testing* (3. uud. painos). John Wiley & Sons, 5-18.

Pap, J., Mako, C., Illessy, M., Kis, N., & Mosavi, A. (2022). Modeling organizational performance with Machine Learning. *Journal of Open Innovation: Technology, Market, and Complexity*, *8*(4), 177. https://doi.org/10.3390/joitmc8040177

Procaccino, J. D., Verner, J. M., & Lorenzet, S. J. (2006). Defining and contributing to software development success. *Communications of the ACM*, *49*(8), 79–83. https://doi.org/10.1145/1145287.1145291

Rapps, S., & Weyuker, E. J. (1985). Selecting software test data using data flow information. *IEEE Transactions on Software Engineering*, *SE-11*(4), 367–375. https://doi.org/10.1109/tse.1985.232226

Reel, J. S. (1999). Critical success factors in software projects. *IEEE Software*, *16*(3), 18–23. https://doi.org/10.1109/52.765782

Richardson, I., Casey, V., McCaffery, F., Burton, J., & Beecham, S. (2012). A process framework for global software engineering teams. *Information and Software Technology*, *54*(11), 1175–1191. https://doi.org/10.1016/j.infsof.2012.05.002

Sandeep, S. R., Ahamad, S., Saxena, D., Srivastava, K., Jaiswal, S., & Bora, A. (2022). To understand the relationship between machine learning and artificial intelligence in large and diversified business organisations. *Materials Today: Proceedings*, *56*, 2082–2086. https://doi.org/10.1016/j.matpr.2021.11.409

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge: Cambridge University Press, 19-29. https://doi.org/10.1017/cbo9781107298019

Shete, N., & Jadhav, A. (2014). An empirical study of test cases in software testing. *International Conference on Information Communication and Embedded Systems (ICICES2014)*. https://doi.org/10.1109/icices.2014.7033883

Sneha, K., & Malle, G. M. (2017). Research on software testing techniques and software automation testing tools. *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*. https://doi.org/10.1109/icecds.2017.8389562

Stocco, A. (2019). How artificial intelligence can improve web development andtesting. *Proceedings of the Conference Companion of the 3rd International Conference on Art, Science, and Engineering of Programming*. https://doi.org/10.1145/3328433.3328447

Templier, M., & Paré, G. (2015). A framework for guiding and evaluating literature reviews. *Communications of the Association for Information Systems*, *37*. https://doi.org/10.17705/1cais.03706

Toosi, A., Bottino, A. G., Saboury, B., Siegel, E., & Rahmim, A. (2021). A brief history of AI: How to prevent another winter (a critical review). *PET Clinics*, *16*(4), 449–469. https://doi.org/10.1016/j.cpet.2021.07.001

Toreini, E., Aitken, M., Coopamootoo, K., Elliott, K., Zelaya, C. G., & van Moorsel, A. (2020). The relationship between trust in AI and Trustworthy Machine Learning Technologies. *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. https://doi.org/10.1145/3351095.3372834

Trzeciak, M., Sienkiewicz, Ł. D., & Bukłaha, E. (2022). Enablers of open innovation in software development micro-organization. *Journal of Open Innovation: Technology, Market, and Complexity*, *8*(4), 174. https://doi.org/10.3390/joitmc8040174

Vergilio, S. R., Maldonado, J. C., & Jino, M. (2006). Infeasible paths in the context of data flow based testing criteria: Identification, classification and prediction. *Journal of the Brazilian Computer Society*, *12*(1), 73–88. https://doi.org/10.1007/bf03192389

Wan, Z., Xia, X., Lo, D., & Murphy, G. C. (2020). How does machine learning change software development practices? *IEEE Transactions on Software Engineering*, 1–1. https://doi.org/10.1109/tse.2019.2937083

Wirtz, B. W., Weyerer, J. C., & Geyer, C. (2019). Artificial Intelligence and the public sector—applications and challenges. *International Journal of Public Administration*, *42*(7), 596–615. https://doi.org/10.1080/01900692.2018.1498103

Yadav, V., Botchway, R. K., Senkerik, R., & Kominkova Oplatkova, Z. (2021). Robotic Automation of software testing from a machine learning viewpoint. *MENDEL*, *27*(2), 68–73. https://doi.org/10.13164/mendel.2021.2.068

Zhang, D., & Tsai , J. J. (2003). Machine Learning and Software Engineering. *Software Quality Journal*, *11*(2), 87–119. https://doi.org/10.1023/a:1023760326768

Zhang, H., Babar, M. A., & Tell, P. (2011). Identifying relevant studies in software engineering. *Information and Software Technology*, *53*(6), 625–637. https://doi.org/10.1016/j.infsof.2010.12.010

Zheng, Y., Yu, H., Cui, L., Miao, C., Leung, C., & Yang, Q. (2018). SmartHS: An AI platform for improving government service provision. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1). https://doi.org/10.1609/aaai.v32i1.11382