Lauri Vastela

Generatiivisen tekoälyn hyödyntäminen ohjelmistokehityksen prosesseissa

Tekniikan ja innovaatiojohtamisen yksikkö
Pro gradu -tutkielma
Tietojärjestelmätiede, Kauppatieteiden maisteri

Vaasa 2025

VAASAN YLIOPISTO
Tekniikan ja innovaatiojohtamisen yksikkö
Tekijä: Lauri Vastela
Tutkielman nimi: Generatiivisen tekoälyn hyödyntäminen ohjelmistokehityksen prosesseissa
Tutkinto: Kauppatieteiden maisteri
Oppiaine: Tietojärjestelmätiede
Työn ohjaaja: Tomi Pasanen
Valmistumisvuosi: 2025 **Sivumäärä:** 82

TIIVISTELMÄ:
Tämän pro gradu -tutkielman tavoitteena on tarkastella generatiivisten tekoälytyökalujen omaksumista ja hyödyntämistä ohjelmistokehityksen prosesseissa. Tutkimuksessa selvitetään, kuinka kyseiset työkalut voivat tukea ohjelmistokehittäjiä ohjelmistokehityksen elinkaaren eri vaiheissa. Lisäksi käsitellään tekoälyn käyttöönottoon liittyviä haasteita ja sen vaikutuksia työympäristöön, kuten kehittäjien rooleihin ja tarvittaviin osaamisiin.

Generatiivinen tekoäly tuottaa uutta sisältöä pelkän analysoinnin sijaan. Se pohjautuu laajoihin tietoaineistoihin ja monimutkaisiin neuroverkkorakenteisiin, joiden ansiosta se kykenee luomaan monimutkaisia, jopa luovia ratkaisuja. Ohjelmistokehityksen näkökulmasta generatiiviset mallit voivat tehostaa koodin kirjoittamista, testausprosessia ja virheenkorjausta, mutta ne saattavat tuoda mukanaan myös uusia virhelähteitä ja muuttaa kehittäjien toimenkuvaa.

Aihetta on tärkeä tutkia, sillä tekoälyn käyttöönotto tarjoaa ohjelmistokehitykselle merkittäviä mahdollisuuksia, kuten kehitysajan lyhentämistä ja laadun parantamista, mutta se vaatii myös panostusta osaamiseen, työroolien uudelleenmäärittelyyn sekä teknisiin ja eettisiin kysymyksiin. Tämän tutkielman tavoitteena on syventää ymmärrystä siitä, miten tekoälytyökalut voivat parantaa kehitysprosessia ja millaisia haasteita niiden hyödyntäminen tuo mukanaan.

Tutkimus toteutettiin kvalitatiivisena tutkimuksena ja tutkimusaineisto on kerätty puolistrukturoidun teemahaastatteluiden kautta. Teoreettisessa viitekehyksessä perehdytetään lukija generatiivisen tekoälyn taustalla oleviin teknologioihin kuten koneoppimiseen, luonnollisen kielen käsittelyyn ja suuriin kielimalleihin. Tämän lisäksi esitellään tekoälyn hyödyntämistä ohjelmistokehityksen prosesseissa kuten ohjelmoinnissa, testauksessa ja vaatimusmäärittelyssä. Haastateltavaksi valikoitui kokeneita ammattilaisia ohjelmistokehityksen alalta, joilla on laaja käsitys tekoälytyökalujen hyödyntämisestä omassa työkuvassaan. Empiirisessä osiossa teemat valikoituivat tutkimuskysymysten ja teoreettisen viitekehyksen pohjalta.

Tutkimuksen perusteella tekoälytyökalut tukevat monella eri tapaa ohjelmistokehittäjiä. Tulokset osoittavat generatiivisen tekoälyn olevan hyödyllistä ohjelmistokehityksessä varsinkin koodin generoinnissa ja ideoiden luonnostelussa. Työkaluja on hyödyllistä käyttää ohjelmistokehityksen rutiininomaisissa tehtävissä ja säästäen aikaa ajattelutyölle. Tekoälytyökaluilla on myös haittansa, kuten virheellisten tuotosten tuottaminen ja mahdolliset huolet tietoturvariskeistä. Tekoälyn hyödyntäminen ohjelmistokehityksessä vaatii sen käyttäjältä tiedostuksen sen hyödyistä ja haitoista.

AVAINSANAT: Generatiivinen tekoäly, Tekoäly, Suuret kielimallit, Syväoppiminen, Koneoppiminen, Luonnollisen kielen käsittely, Ohjelmistokehitys

Sisällys

Kuvat

Taulukot

1 Johdanto

Tässä luvussa käsitellään tämän pro gradu -tutkielman taustaa, rakennetta, aineistoa sekä tutkimusmenetelmää. Tutkimuksessa tarkastellaan generatiivisten tekoälytyökalujen hyödyntämistä ohjelmistokehityksen prosesseissa. Tavoitteena on selvittää, miten generatiiviset tekoälytyökalut voivat tukea ohjelmistokehittäjiä parantaen ohjelmistojen laatua sekä kehittäjien tuottavuutta. Lisäksi tutkimuksessa tarkastellaan generatiivisen tekoälyn käyttöönoton omaksumista, haasteita ja sen vaikutuksia työympäristöön, kuten kehittäjien rooleihin ja osaamistarpeisiin.

1.1 Tutkimuksen tausta

Tekoäly on kehittynyt paljon ja sitä on tutkittu laajasti viimeisen vuosikymmenen aikana, saaden runsaasti huomiota erityisesti viime vuosina (Patil & Pramod, 2024, s. 1). Tekoälyn mullistava potentiaali käy yhä ilmeisemmiksi eri aloilla, ja tekoälymallit osoittavat kyvykkyyksiä muun muassa luonnollisen kielen ymmärtämisessä sekä generatiivisien mallien tuottamassa sisällössä ihmisen antamien syötteiden avulla (Zhang ja muut, 2021, s. 10–11). Yksi alue, jossa tämä potentiaali on erityisen ilmeinen, on ohjelmistotekniikka, joka on keskeinen toiminto nykyorganisaatioissa. Sen merkitystä korostaa ohjelmistojen yhä laajempi levinneisyys eri tuotteissa ja palveluissa, joissa digitaaliset ominaisuudet parantavat niiden arvoa (Ghai ja muut, 2024, s. 1–2).

Ohjelmistokehityksen koko elinkaaren vaiheissa tekoälytyökalut voivat toimia arvokkaina kumppaneina. Tekoälyn hyödyntäminen ohjelmistokehityksessä ei vain rajoitu pelkästään automaatioon, vaan se voi myös auttaa kehittäjiä luomaan innovatiivisia ja käyttäjäystävällisempiä ohjelmistoratkaisuja (Ghai ja muut, 2024, s. 1–2). Ohjelmistokehitystekniikka ei ole poikkeus uusien trendien käyttöönotossa ja niiden vaikutuksissa. Tekoäly on jo pitkään muokannut ohjelmistoalaa, helpottaen kehittäjien työtä automatisoimalla toistuvia tehtäviä, parantamalla virheenjäljitystä, tehostamalla testausta ja tarjoamalla monia muita hyödyllisiä toimintoja. Generatiivisten tekoälytyökalujen kasvava saatavuus

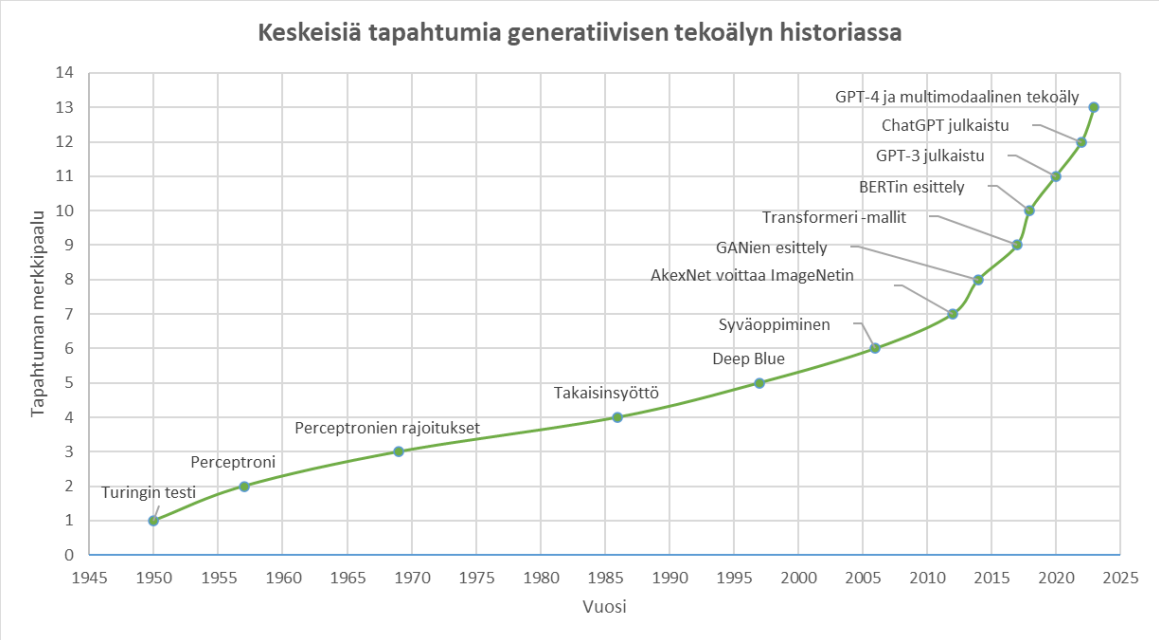
lisää entisestään niiden hyötyä ohjelmistokehittäjien päivittäisessä työssä (Petrovska ja muut, 2024, s. 1–2; Ghai ja muut, 2024, s. 1–2).

Generatiivinen tekoäly viittaa tekoälyn osa-alueeseen, joka pystyy luomaan uutta sisältöä sen sijaan, että se vain analysoisi ja toimisi olemassa olevan datan pohjalta, kuten asiantuntijajärjestelmät. Tekoälymallit, jotka on varustettu laajoilla tietoaineistoilla ja monimutkaisilla rakenteilla, omaavat poikkeuksellisen kyvyn luoda uutta ja monimutkaista sisältöä käyttäjien syötteiden pohjalta (Gozalo-Brizuela & Merchan, 2024, s. 1–2). Generatiivinen tekoäly on saanut merkittävää huomiota tutkimuksissa ja teollisuudessa viime vuosina. Viimeisimmät tutkimukset osoittavat, kuinka tekoäly tulee olemaan merkittävä tekijä tulevaisuuden talouden vauhdittamisessa eri liiketoiminta-alueilla personoinnin, tiivistämisen ja viestinnän kautta (Simaremare & Edison, 2024, s. 1).

Aihetta on tärkeä tutkia, sillä tekoälyn käyttöönotto ohjelmistotekniikassa tuo merkittäviä mahdollisuuksia, mutta myös samalla haasteita ohjelmistokehitykseen. Teknologian nopean kehittymisen vuoksi yritysten tulee pysyä kilpailussa mukana ja edellä, mikä edellyttää jatkuvan kehityksen ja uusien käytäntöjen haltuunottoa, kuten tekoälyn hyödyntämistä (Nguyen-Duc ja muut, 2023, s. 2). Ghai ja muut (2024) vertasivat tutkimuksessaan ohjelmointitehokkuutta tekoälytyökaluja hyödyntäen ja eroa manuaalisen ohjelmoinnin välillä. Tutkimus osoitti, että tekoälyn avulla tuotettu koodi parantaisi kehitysaikoja ja lisäisi koodirivien määrää verrattuna manuaalisesti kirjoitettuun koodiin, samalla kun tekoälytyökalua käyttäen toteutettu ohjelmointi mahdollisesti tuottaisi pienempää virhemäärää verrattuna käsin kirjoitettuun koodiin. Vastaavasti Pangavhane ja muut (2024) totesivat tutkimuksessaan, että ohjelmistokehittäjien hyödyntäessä tekoälytyökaluja, kuten ChatGPT:tä ja GitHub Copilotia, työkalut toivat paljon hyötyjä ohjelmistokehityksen vaiheissa kuten koodin generoinnissa, testauksessa ja virheenkorjauksissa lisäten ohjelmoijien tehokkuutta huomattavasti.

Generatiivinen tekoäly on levinnyt nopeasti yleiseen käyttöön. OpenAI:n kehittämä ja vuonna 2022 julkaisema ChatGPT saavutti 100 miljoonaa käyttäjää vain kahdessa kuu-

kaudessa, vertailun vuoksi mobiilisovellus TikTok sai vastaavan määrän yhdeksässä kuukaudessa (Ebert & Louridas, 2023, s. 2). Deloitten Digital Consumer Trends 2023 -kyselyn mukaan 52 % Yhdistyneen kuningaskunnan väestöstä tuntevat generatiivista sisältöä tuottavia tekoälytyökaluja, ja heistä puolet on käyttänyt niitä (Petrovska ja muut, 2024, s. 1).



Kuva 1 Tapahtumia generatiivisen tekoälyn historiassa (pohjautuu Chopra, 2024)

1.2 Tutkimuksen tavoite

Tutkimuksen tavoitteena on selvittää kuinka generatiivinen tekoäly vaikuttaa nykypäivänä ohjelmistokehittäjien työkuvaan ja mitkä sen käytön vaikutukset ovat. Tutkimuskysymysten määrittely on keskeinen osa sekä kvantitatiivisesta että kvalitatiivisesta tutkimusprosessia, sillä ne rajaavat tutkimuksen tavoitetta ja rakennetta siten, että tutkijat pystyvät tarkemmin keskittyä tiettyyn tutkimusmenetelmään. Tutkimuskysymykset määritellään ja muotoillaan yleensä tutkimuksen tarkoituksen tai tavoitteiden perusteella

(Onwuegbuzie & Leech, 2006, s. 1–2). Tutkimuskysymykset antavat myös ennakkokatsauksen tutkimuksen eri osista ja muuttujista, jotka on tarkoitettu käsittelemään tutkimuskysymyksessä esitettyä ongelmaa (Barroga & Matanguihan, 2022, s. 2).

TK 1: Kuinka hyödyllistä on hyödyntää generatiivisen tekoälyn työkaluja ohjelmistokehityksessä?

TK 2: Miten generatiivisen tekoälyn käyttöönotto muuttaa perinteisiä ohjelmistokehitysprosesseja ja mitä vaatimuksia se asettaa ohjelmistokehittäjälle?

Ensimmäinen tutkimuskysymys on asetettu siten, että sen myötä voidaan selvittää, kuinka hyödyllistä on hyödyntää tekoälytyökaluja ohjelmistokehityksessä. Tutkimuskysymys on tärkeä, sillä tekoälytyökalujen käyttö lupaa tutkimusten perusteella tehostavan ohjelmistokehitysprosesseja eri tavoin: esimerkiksi parantaen ohjelmistokehittäjien tehokkuutta, mikä jättää aikaa uusille luovuutta vaativille työtehtäville, kuten suunnittelulle ja ohjelmistojen laadunvarmistukselle. Samalla voidaan selvittää millaisia muutoksia ne tuovat ohjelmistokehittäjien perinteisiin rooleihin ja osaamistavoitteisiin. Ilman tutkimusta aiheesta, organisaatiot ja ohjelmistokehittäjät eivät pysty täysimääräisesti hyödyntämään tekoälyn etuja ja vaihtoehtoisesti sortuvat ratkaisuihin ja tapoihin, jotka eivät ole kannattavia pitkällä aikavälillä. Tästä näkökulmasta tämä pro gradu -työ ei anna ainoastaan kokonaiskuvaa tekoälyn teknisistä hyödyistä, vaan myös varmistaa, että tekniikka voidaan integroida koko ohjelmistokehityksen elinkaaren prosesseihin.

Tekoälyn käyttöönotto voi mullistaa perinteisiä ohjelmistokehitystapoja, sillä se mahdollistaa esimerkiksi vaatimusmäärittelyn, suunnittelun, testauksen sekä koodin tuottamisen nopeammin ja luovemmin. Toisen tutkimuskysymyksen perusteella selvitetään, millaisia taitoja ohjelmistokehittäjiltä vaaditaan, kuten tekoälyn ymmärtämistä ja sen tuomia rajoitteita ohjelmistokehityksen prosesseissa. Lisäksi on tarpeen tarkastella, miten kehittäjien vastuu, päätöksenteko ja virheidenhallinta muuttuvat, kun tekoäly suorittaa ja avustaa ohjelmistokehityksen vaiheita.

1.3 Tutkimuksen rakenne ja rajaus

Tutkielma on laaja ja tämä pro gradu -tutkielma keskittyy tähän mennessä toteutettuihin tutkimuksiin generatiivisen tekoälyn hyödyntämisestä ohjelmistokehityksen prosesseissa. Tutkielma koostuu kuudesta eri luvusta, joiden tavoitteena on herättää lukijan mielenkiinto ja tarjota selkeä ymmärrys käsiteltävästä aiheesta. Luvut on rakennettu edettävän loogisesti ja johdonmukaiseksi kokonaisuudeksi. Jokainen luku sisältää alilukuja, jotka vaihtelevat aihepiirin mukaan ja auttavat lukijaa hahmottamaan kokonaisuuden sekä saamaan kattavan kuvan käsiteltävästä aiheesta.

Toisessa luvussa lukija johdatetaan tekoälyn käytäntöihin, kuten koneoppimiseen ja syväoppimiseen. Lisäksi käsitellään luonnollisen kielen käsittelyä sekä suuria kielimalleja, jotka muodostavat generatiivisen tekoälyn perustan ja antavat laajemman kuvan sen mahdollisuuksista ja teknisen ymmärryksen niiden toimintaperiaatteista.

Lukujen tavoitteena on perehdyttää lukija tutkimuksen pääkonsepteihin ja syventää ymmärrystä käsiteltävistä aiheista. Esitetyt aiheet perustuvat tieteellisiin artikkeleihin ja kirjoihin, joiden tekijät ovat alansa asiantuntijoita ja tutkijoita. Kirjallisuuskatsauksessa hyödynnetyt materiaalit on hankittu Vaasan yliopiston www.tritonia.finna.fi -sivuston kautta saatavista tietokannoista, kuten Scopus, IEEE Xplore, Ebook Central ja ScienceDirect. Lisäksi lisämateriaalin hankinnassa on käytetty Google Scholaria. Tutkimusaineisto rajautuu käsiteltävän aiheen ajankohtaisuuden mukaan, mutta pääsääntöisesti se kattaa viimeisen viiden vuoden aikana tuotetun aineiston. Tekoälyn ja sen osa-alueiden nopean kehityksen vuoksi lähdeaineisto keskittyy pääasiassa vuodesta 2020 eteenpäin, ja generatiivisen tekoälyn osalta vuodesta 2023 lähtien.

Kolmannessa luvussa käsitellään generatiivisen tekoälyn hyödyntämisestä ohjelmistokehityksen eri vaiheissa. Generatiivisen tekoälyn käsite esitellään kattavasti, jotta lukijalle muodostuu selkeä ymmärrys aiheesta ja sen menetelmistä, jotka ovat keskeisiä tulevissa luvuissa. Lukujen aliluvut on jaoteltu teemoittain, jotka vastaavat ohjelmistokehitysprosessin eri vaiheita.

Neljäs luku esittelee tutkimuksessa käytetyt menetelmät sekä tiedonkeruu- että analyysimenetelmät. Viidennessä luvussa esitellään empiirisen tutkimuksen tulokset. Viimeisessä luvussa tehdään yhteenveto ja johtopäätökset tutkimuksen päätelmistä, tämän lisäksi pohditaan ennalta määriteltyjen tutkimuskysymysten löydöksiä. Lopuksi annetaan suositukset tulevaisuuden jatkotutkimuksille ja käytännön sovelluksille sekä tarkastellaan tutkimuksen rajoituksia ja käsitellään pohdintaa tutkimuksen tuloksista.

Tässä tutkielmassa termillä ”tekoäly” viitataan aina generatiiviseen tekoälyyn turhan termien toiston välttämiseksi. Poikkeuksen muodostavat toisen luvun teoreettisen viitekehyksen aliluvut, joissa käsitellään generatiivisen tekoälyn teknistä taustaa ja käyttötarkoituksia ja sanalla tekoäly viitataan tekoälyn ylä- ja aliluokkiin.

1.4 Tutkimusmenetelmä ja tutkimusaineisto

Tutkimus toteutetaan kvalitatiivisella tutkimusmenetelmällä. Kvalitatiivinen eli laadullinen tutkimusmenetelmässä on suositeltavaa tilanteissa, kun halutaan ymmärtää ilmiön laadullista ja ihmislähtöisiä puolia sekä syvällisempiä merkityksiä. Tämä tapahtuu laadullisen tutkimuksen tutkimusstrategiassa siten, että sanoja painotetaan enemmän kuin määrällistä mittaamista aineiston keruussa ja analyysissä (Hammersley, 2012, s. 12). Tekoälytyökalujen hyödyntäminen ohjelmistokehityksessä on monimuotoinen prosessi, johon liittyy tekijöitä, joita ei voida täysin ymmärtää pelkkien numeeristen mittareiden tai määrällisen datan perusteella. Esimerkiksi kehittäjien kokemukset, motivaatiot, asenteet ja mahdolliset huolenaiheet sekä niiden vaikutukset organisaatiokulttuuriin ja työyhteisöön edellyttävät laadullista tutkimusotetta.

Tutkimusaineisto on kerätty tietojärjestelmätieteen sekä tietotekniikan tietokannoista hyödyntämällä akateemisia lähteitä ja tutkimuksia. Tutkimusaineisto on kerätty huolellisesti ajankohtaisista lähteistä, jotta saadaan vertailukelpoista materiaalia useista eri näkökohdista ja mahdollisimman tarkka kuva käsitellyistä tutkimusaiheista.

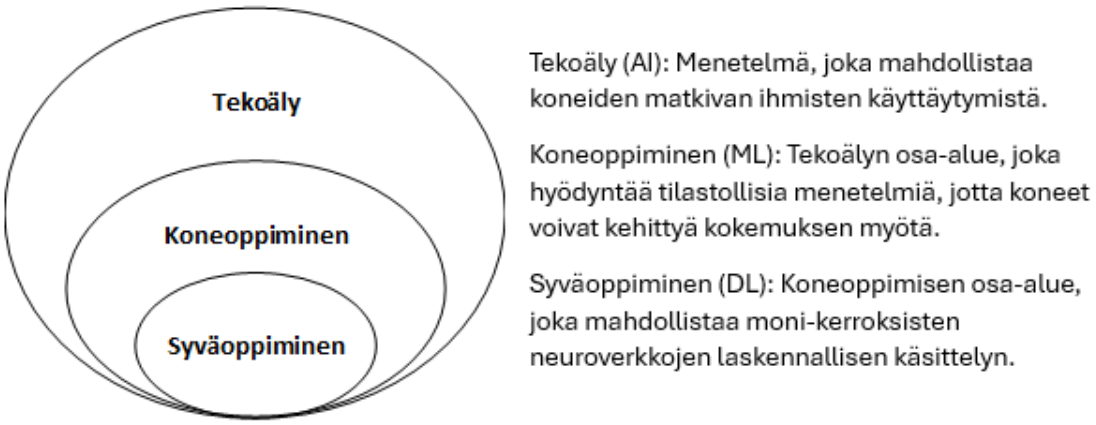
2 Tekoäly

Tekoäly (Artificial Intelligence) on laaja käsite tietotekniikan osa-alueella, joka voidaan ymmärtää tarkoittavan tietokoneohjelmoinnin keskittymistä koneiden kouluttamiseen ja tehtävien suorittamiseen. Tekoälyä voidaan käyttää testaamaan päättelyn teorioita, kuten kognitiivista päättelyä ja tietoisuutta (Harika ja muut, 2022, s. 1). Tekoälyn käsite ehdotettiin ensimmäisenä Yhdysvalloissa 1950-luvulla tavoitteena, että koneet saavat tekoälykomentoja ja voivat siten korvata ihmisisen suorittamalla mekaanisia töitä arjessa ja työelämässä sekä toteuttamaan vaikeita ja vaarallisia tehtäviä. Täten pyritään parantamaan työn tehokkuutta ja tarkkuutta (Ma, 2023, s. 1; Li ja muut, 2021, s. 2).

Tekoälyn avulla voidaan yhdistää useita tieteenaloja, kuten psykologian, biologian, tietojenkäsittelyn ja matematiikan tarjoten tarkan tiedonkäsittelyn, erinomaisen oppimiskyvyn sekä tehokkaan laskentakapasiteetin. Nykyään tekoälyä hyödynnetään laajasti eri aloilla kuten verkkokaupoissa, itseohjautuvissa ajoneuvoissa, koulutuksessa, terveydenhuollossa sekä rahoituksessa (Ma, 2023, s. 1–2). 2000-luvulle siirryttäessä tekoälylle hyödynnettävien laitteistokapasiteetti on kehittynyt merkittävästi. Itsenäisen laskentatehon ja verkkoyhteyksien parantuminen on lisännyt tekoälyn kykyä käsitellä entistä enemmän tietoa tehokkaammin. Muun muassa Big data (suuret datamäärät) -teknologia on luonut vankan pohjan koneoppimiselle, kun taas pilvilaskennan ja verkkoteknologian edistysaskeleet ovat nopeuttaneet tekoälyn kehitystä erityisesti koneoppimisen osa-alueella (Li ja muut, 2021, s. 2).

Tekoälyn alle kuuluu useita eri osa-alueita. Yksi näistä on koneoppiminen, joka saatetaan usein sekoittaa tekoälyyn (Ma, 2023, s. 4). Koneoppiminen hyödyntää algoritmeja löytääkseen malleja ja tuottaakseen oivalluksia käsittelemästään datasta. Koneoppimisen osa-alueen alle kuuluva syväoppiminen vie taas tekoälyä lähemmäs tavoitetta saada koneet ajattelemaan ja toimimaan mahdollisimman ihmismäisesti (Harkut, 2019, s. 15). Tekoälypohjaisia järjestelmiä kehitetään ja otetaan käyttöön monissa ympäristöissä, ja

tekoälyjärjestelmiltä odotetaan yhä enemmän itsenäistä toimintaa. Erityisesti koneoppimista on hyödynnetty monenlaisissa tehtävissä, ja siitä on tullut olennainen osa arkipäivää (Zhang ja muut, 2021, s. 10–11).



Kuva 2 Tekoälyn osa-alueet (pohjautuu Walker, 2025)

2.1 Koneoppiminen

Koneoppiminen (Machine Learning, ML) ja muut edistyneet laskentamallit ovat muodostaneet tekoälyn perustan, mahdollistaen koneiden jäljittelemään ihmisen älykkyyttä ja jopa ylittää tietyissä tehtävissä (Ghai ja muut, 2024, s. 1). Koneoppiminen on tiedon analysointiin liittyvä menetelmä ja osa tekoälyä, jossa järjestelmä pystyy oppimaan aikaisemmasta datasta, tunnistamaan kaavoja tai jakaumia aineistosta ja tekemään sen pohjalta päätöksiä. Toisin sanoen se on järjestelmä, joka automatisoi analyysimallien rakentamisen minimoiden ihmisen puuttumisen prosessiin (Wang ja muut, 2020, s. 2). Nykyiset tutkimukset koneoppimisessa keskittyvät varsinkin luonnollisen kielen käsittelyyn, konenäköön, kuvioiden tunnistamiseen, kognitiiviseen laskentaan ja tiedon esittämiseen (Rincy & Gupta, 2020, s. 1). Berradan ja muiden (2022) mukaan koneoppiminen voidaan jakaa neljään eri oppimispääkategoriaan: valvottuun, valvomattomaan, puolivalvottuun ja vahvistettuun. Kaikille koneoppimisen osa-alueille on omat käyttötarkoituksensa ja algoritminsa.

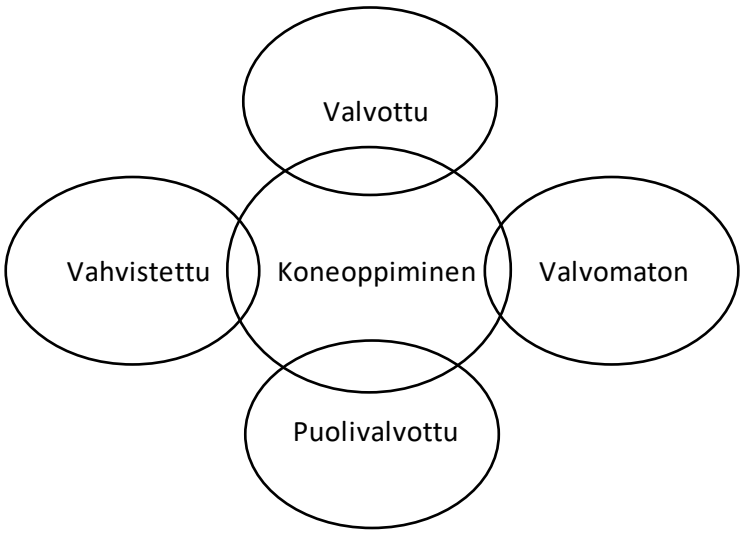
Valvottua koneoppimista käytetään ensisijaisesti regressio- ja luokittelutehtävissä ennalta määrätyn harjoitteluaineiston perusteella, joissa sekä syöte että haluttu lopputulos ovat ennalta määritettyjä. Malli oppii tunnistamaan syötteen ja odotetun tuloksen välisiä kaavoja (Sun ja muut, 2023, s. 1–2). Valvottua koneoppimista pidetään perustuvan erilaisiin koneoppimisalgoritmeihin, kuten neuroverkkoihin (Neural Network), päätöspuihin (Decision Tree), tukivektorikoneeseen (Support Vector Machine) ja päätöspöytiin (Decision Table). Syedin ja Lokhanden (2024) mukaan valvotun koneoppimisen tavoitteena on ymmärtää dataa tietyn kysymyksen kontekstissa.

Valvottamassa koneoppimisessa data on täysin merkitsemätöntä, ja malli pyrkii tunnistamaan rakenteita ilman valmiita vastauksia (Rincy & Gupta, 2020, s. 3). Valvomattomassa oppimisessa ei ole ennustettavaa tavoitemuuttujaa. Tätä menetelmää hyödynnetään ensisijaisesti datan ryhmittelyssä, mikä mahdollistaa uusien havaintojen tekemisen eri ryhmien välillä (Teles ja muut, 2020, s. 3). Valvomaton koneoppiminen puolestaan soveltuu erityisesti klusterointiin ja ulottuvuuksien vähentämiseen. Siinä lopputulosta ei määritellä etukäteen, vaan menetelmä pyrkii tunnistamaan syötedatan välisiä yhteyksiä ja löytämään piileviä rakenteita tai kuvioita (Sun ja muut, 2023, s. 2).

Puolivalvottu koneoppiminen yhdistää sekä merkittyä että merkitsemätöntä dataa hyödyntäen molempien etuja, vaikka merkittyä dataa on vähemmän merkitsemättömään dataan nähden (Rincy & Gupta, 2020, s. 4). Puolivalvottu oppiminen käsittelee luokittelua tilanteissa, joissa vain osa havainnoista on varustettu luokkatunnisteilla. Tämä lähestymistapa on erityisen hyödyllinen sovelluksissa, kuten kuvahakujärjestelmissä, genomiikassa, luonnollisen kielen jäsentämisessä ja puheanalyysissä. Näillä aloilla merkitsemätöntä dataa on runsaasti, mutta koko aineiston luokittelu voi olla kallista tai jopa mahdotonta (Kingma ja muut, 2014, s. 1–2).

Vahvistetussa koneoppimisessa ohjelmistoagentti oppii vuorovaikuttamalla ympäristönsä kanssa ja tekemällä päätöksiä, jotka maksimoivat saadun palkkion (Rincy & Gupta,

2020, s. 4–5). Vahvistetussa koneoppimisessa aktiivinen oppiminen tapahtuu vuorovaikuttamalla, jossa malli tunnistaa varmat sekä epävarmat tiedot ja pyytää asiantuntijoita merkitsemään ne oikein. Näin algoritmi parantaa ennustustarkkuuttaan koulutuksen aikana. Menetelmää käytetään erityisesti tietoturvassa, jossa luotettavan ja riittävästi merkityn datan saatavuus on usein rajallista (Kachynskyi & Tsebrinska, 2020, s. 2).

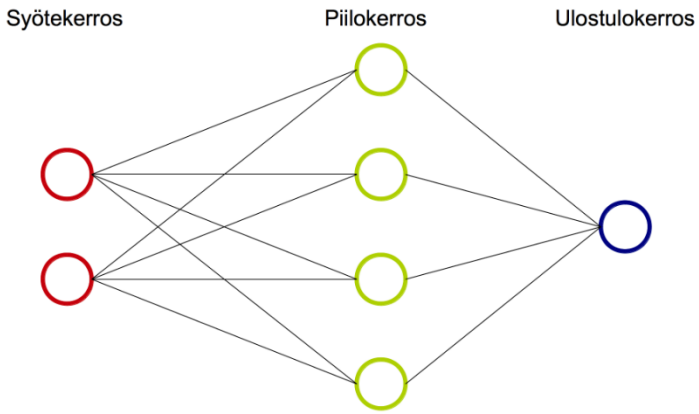


Kuva 3 Koneoppimisen kategoriat (pohjautuu Preis, 2022)

2.2 Syväoppiminen

Syväoppimisen juuret ulottuvat 1980-luvulle asti, jolloin tietokoneelle alettiin opettamaan aiemman tiedon hyödyntämistä. Syväoppiminen on koneoppimisen alaryhmä, joka on saanut inspiraationsa ihmisistä ja käyttää neuroverkkoja mallien rakentamiseen sovelluksissa, kuten konenäössä, puheentunnistuksessa ja robotiikassa (Berrada ja muut, 2022, s. 1). Syväoppiminen käsittää keinotekoiset neuroverkot, jotka jäljittelevät ihmisaivojen rakennetta ja toimintaa. Neuroverkko muodostuu nimensä mukaisesti solmuista eli neuroneista, jotka ovat yhteydessä toisiinsa kerroksittain. Dataa käsitellään toistuvasti näiden kerrosten läpi, jotta voidaan tehdä korkean todennäköisyyden ennusteita (Dias & Laueretta, 2024, s. 2).

Dongaren ja muiden (2012) mukaan neuroverkkojen tieto kulkee ensiksi syötekerroksesta piilokerroksen kautta ulostulokerrokseen. Piilokerroksissa tapahtuu tietojen laskennallinen prosessointi, jossa painotettuja yhteyksiä ja aktivointifunktioita hyödyntämällä neuroverkko oppii tunnistamaan monimutkaisia kaavoja ja rakenteita syötteestä. Tällaiset mallit ovat keskeisiä monilla tekoälyn osa-alueilla, kuten kuvantunnistuksessa, luonnollisen kielen käsittelyssä ja puheentunnistuksessa, joissa tarvitaan suuria määriä dataa ja tehokasta laskentaa ennusteiden tekemiseen (Wright ja muut, 2022, s. 1). Neuroverkon rakenne on kuvattu alla olevassa kuvassa.



Kuva 4 Neuroverkon rakenne (Tuominen & Neittaanmäki, 2019)

Syväoppimisella on merkittävä rooli generatiivisessa tekoälyssä ja suurissa kielimalleissa. Syväoppimismallit oppivat kielen rakenteita ja sääntöjä analysoimalla laajoja tietoaineistoja, minkä ansiosta ne voivat luoda luonnollisia ja johdonmukaista tekstiä. Erityisesti transformer-arkkitehtuuriin pohjautuvat esikoulutetut kielimallit, kuten GPT, ovat tehokkaita luonnollisen kielen käsittelyssä ja tarjoavat vankan pohjan tekoälyn tuottamalle tekstille (Mo ja muut, 2024, s. 1; Rane ja muut, 2024, s. 10–11).

2.3 Luonnollisen kielen käsittely

Luonnollisen kielen käsittely (Natural Language Processing) yhdistää ohjelmistokehitystä, semanttisuutta ja tekoälyä. Luonnollisen kielen käsittely keskittyy siihen, miten tietokoneet ja ihmiskieli voivat toimia yhdessä ymmärtääkseen, tuottaakseen ja tulkitakseen inhimillisen kaltaista tekstiä tai puhetta hyödyntäen tekoälyä ja koneoppimista. Käsittelytekniikat ovat kehittyneet nopeasti ovat muodostuneet keskeiseksi osaksi monia sovelluksia, varsinkin ihmisen ja tietokoneen välisen yhteistyön edistämisessä (Das & Das, 2024, s. 1–2).

Luonnollisen kielen käsittelyssä tekstiä yksinkertaistetaan, yleistetään ja tarkennetaan siten, että alkuperäinen informaatio ja merkitys säilyvät, jotta teksti olisi helpompi ymmärtää ja on luettavampaa. Tämä voi sisältää pitkien lauseiden lyhentämistä, yksinkertaistaa sanastoa, poistaa tarpeettomia yksityiskohta tai uudelleen järjestää rakenteita loogisen selkeyden parantamiseksi (Huo ja muut, 2024, s. 1–2). Lukuisat luonnollisen kielen käsittelyn sovellukset perustuvat laadukkaaseen, merkittyyn dataan, mikä usein vaatii suuria määriä annotoitua tekstiä koneoppimismallien koulutukseen, optimointiin tai arviointiin (Nasution & Onan, 2024, s. 2).

Viime vuosina on tapahtunut merkittävä muutos suurten kielimallien, kuten GPT-3:n ja sen vastineiden nousun myötä. Nämä kehittyneet mallit, jotka perustuvat neuroverkkorakenteisiin ja laajaan esikoulutukseen, edustavat kielen ymmärtämisen huippua (Nasution & Onan, 2024, s. 2). Ala jatkaa kehittymistään ja on hyvin todennäköistä, että tekoälyn uudet innovaatiot vievät luonnollisen kielen käsittelyä yhä pidemmälle. Tämä kehitys luo uusia mahdollisuuksia muun muassa automaattiseen sisällöntuotantoon, monikieliseen viestintään ja entistä älykkäämpiin vuorovaikutusjärjestelmiin, jotka mukautuvat käyttäjän tarpeisiin reaaliajassa (Sengar ja muut, 2024, s. 32).

Luonnollisen kielen käsittelyssä esiintyy myös paljon erilaisia haasteita, kuten kielen monimutkaisuus, eri kielet, datan saatavuus, tekstin skaalautuvuus ja tietoturvariskit teke-

vät ihmiskielistä luonnostaan haastavia luonnollisen kielen käsittelyn järjestelmille ymmärtää täysin. Tämä monimutkaisuus aiheuttaa usein vaikeuksia tunnistaa esimerkiksi ironiaa, huumoria tai sarkasmia tekstissä (Das & Das, 2024, s. 4).

Myös datan eheys ja saatavuus ovat merkittäviä haasteita, sillä luonnollisen kielen käsittelyn järjestelmät tarvitsevat korkealaatuista koulutusdataa, jota voi olla vaikea hankkia erityisesti vähemmistökielille tai erikoistuneille aloille. Lisäksi datan vinoumat voivat johtaa epäoikeudenmukaisiin tai vääristyneisiin tuloksiin, mikä tekee datan laadusta kriittisen huolenaiheen (Nasution & Onan, 2024, s. 21). Luonnollisen kielen käsittelyn kehitys on ollut niin nopeaa, että tekniset edistysaskeleet ovat alkaneet ylittää niitä mittaavat vertailuarvot (Zhang ja muut, 2023, s. 11). Luonnollisen kielen käsittelyn ala kehittyy nopeasti ja vuodelle 2025 sen ennustetaan olevan 43 miljardia Yhdysvaltain dollaria, joka on taas 14-kertainen vuoden 2017 kolmeen miljardiin (Das & Das, 2024, s. 4).

2.4 Suuret kielimallit

Suuret kielimallit (Large Language Model, LLM) ovat tekoälyjärjestelmiä, jotka on koulutettu ymmärtämään ja käsittelemään luonnollista kieltä. Ne hyödyntävät syviä neuroverkkoja (Deep Neural Networks) ja pystyvät analysoimaan valtavia määriä kielidataa. Suuret kielimallit koulutetaan laajoilla aineistoilla, jotka sisältävät miljoonia sanoja ja lauseita eri konteksteista. Koulutusprosessi perustuu seuraavan sanan ennustamiseen aiempien sanojen perusteella, minkä ansiosta malli oppii kielen rakenteet, kieliopin ja syntaksin (Nguyen-Duc ja muut, 2023, s. 7–11). Suuret kielimallit kykenevät tuottamaan korkealaatuista tekstiä, joka vastaa ihmisen tuottamaa sisältöä, kuten kielten käännöksissä, tekstin tiivistämisessä, kysymyksiin vastaamisessa ja sisällön generoinnissa (Almarie ja muut, 2023, s. 1).

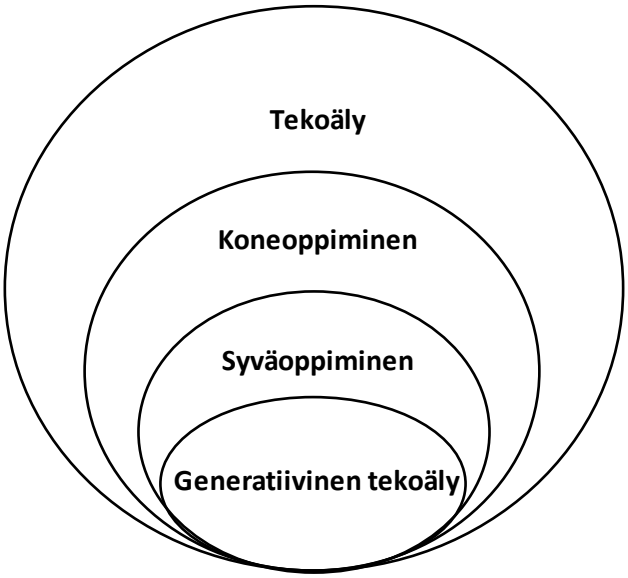
Suuret kielimallit eivät ainoastaan auta ihmismäisen tekstin tuotossa, mutta lisäksi ovat hyödyllisiä ohjelmistokehityksessä muun muassa tarjoamaan koodiehdotuksia, auttamaan dokumentaation laadinnassa, tukemaan vaatimusmäärittelyä ja paljon muuta

(Nguyen-Duc ja muut, 2023, s. 7). Tekoälyn kehitys on edennyt entisestään transformer -arkkitehtuurin integroinnin myötä, mikä parantaa niiden kykyä ymmärtää ja tuottaa kontekstia (Kaplan ja muut, 2020, s. 1). Transformer -arkkitehtuuri parantaa suurten kielimallien tehokkuutta huomattavasti kasvavan laskentatehon ja suuren koulutusdatan saatavuuden avulla (Naveed ja muut, 2024, s. 1).

Suurten kielimallien koulutus perustuu harjoitusdatan lisäksi tokenisaatioon, jossa teksti jaetaan hajoamattomiin yksiköihin. Tokenit voivat olla merkkejä, alisanayksiköitä, symboleja tai sanoja riippuen tokenisointiprosessista. Joitakin yleisesti käytettyjä tokenisointimenetelmiä suurissa kielimalleissa ovat muun muassa wordpiece, byte pair encoding (BPE) ja unigramLM (Naveed ja muut, 2024, s. 4). Tokenisaation prosessi mahdollistaa suurten kielimallien tehokkaan kyvyn käsitellä suuria tekstimääriä, optimoida kielen rakenteiden oppimista ja parantaa ennustusten tarkkuutta sekä luonnollisen tekstin tuottamista (Das & Das, 2024, s. 2).

3 Generatiivinen tekoäly

Generatiivisen tekoälyn juuret ulottuvat 1950-luvulle asti, jolloin tieteilijät alkoivat tutkia, pystyisivätkö tietokoneet luomaan uutta sisältöä tekoälyn sääntöpohjaisen päätöksenteon ja kuvioiden tunnistamisen lisäksi. 1980- ja 1990-luvulla tekoälystä tuli monimuotoisempi tilastotieteellisten mallien, kuten Bayes-verkot ja Markovin mallin myötä, ja näin tekoälyjärjestelmät pystyivät tekemään monimutkaisia päätöksiä ja tuottamaan monipuolisia tuloksia (Chakraborty ja muut, 2023, s. 7–8). Nykyisin generatiivinen tekoäly kuuluu tarkemmin ottaen syväoppimisen piiriin. Syväoppimisen mallit, kuten generatiiviset vastakkaiset verkot (Generative Adversarial Networks) ja variotionaaliset autoenkooderit (Variational Autoencoders) ovat mahdollistaneet tekoälyjärjestelmien luomaan realistisia ja monimutkaisia tuloksia, kuten luonnollista kieltä ja kuvien luontia. Generatiivinen tekoäly keskittyy uuden sisällön tai datan luomiseen annettujen syötteiden pohjalta, hyödyntäen syväoppimista ja neuroverkkoja (Chakraborty ja muut, 2023, s. 7; Chakraborty ja muut, 2025, s. 4; Nguyen-Duc ja muut, 2023, s. 2; Rane ja muut, 2024, s. 1).



Kuva 5 Generatiivisen tekoälyn kategoria (pohjautuu Rane ja muut, 2024)

Suurin kielimalleihin pohjautuvat tekoälymallit, kuten valmiiksi opetettuun transformer-malliin (Generative Pretrained Transformer, GPT), kuuluu transformer-arkkitehtuuriin perustuvia malleja, jotka kykenevät tuottamaan johdonmukaista sekä kontekstiin sopivaa tekstiä muistuttaen ihmisen kirjoitusta. Ne toimivat analysoimalla syötteitä, hyödyntämällä laajoja koulutusaineistojaan ja tuottamalla tarkoituksenmukaisia tekstivastauksia. Näitä malleja voidaan hyödyntää monipuolisesti esimerkiksi kysymyksiin vastaamisessa ja sisällön generoinnissa (Ferrara, 2024, s. 2). GPT-pohjaiset suuret kielimallit on hienosäädetty internetkeskusteluista koostuvalla aineistolla sekä ihmispalautetta hyödyntävällä vahvistusoppimisella (Reinforcement Learning from Human Feedback) (Gamieldien ja muut, 2023, s. 3). Koulutustapansa mukaisesti ne ovat monimutkaisia ja laajamittaisia rakenteita, jotka on koulutettu mittavilla kielellisillä aineistoilla. Ne hyödyntävät kehittyneitä algoritmeja ja syväoppimistekniikoita tuottaakseen kontekstiin sopivaa ja johdonmukaista tekstiä eri sovelluksissa (Aleti, 2023, s. 2).

Transformer-arkkitehtuuri liittyy vahvasti suurien kielimallien koulutusprosessiin, jonka taustalla ovat toistuvat neuroverkot (Recurrent Neural Network), jotka pystyvät ymmärtämään toistuvia tietoja (Rane ja muut, 2024, s. 5–6). Samankaltaiset sanat sijoittuvat tässä tilassa lähemmäs toisiaan (Ebert ja muut, 2024, s. 4). Transformer-arkkitehtuurin vahvuus on sen kyvyssä valikoivasti kiinnittää huomiota sanoihin, jotka muodostavat kontekstuaalisen kehyksen peitettyjen segmenttien ympärillä (Aleti, 2023, s. 8). Tämän ansiosta transformer-pohjaiset kielimallit pystyvät käsittelemään erilaisia luonnollisen kielen tehtäviä yhtenäisellä menetelmällä. Ne on koulutettu tekstistä tekstiin-muodossa, missä sekä syöte että tuloste ovat tekstijonoja. Koulutusprosessi hyödyntää sekä valvomatonta ja valvottua koneoppimista sekä esikoulutuksessa että hienosäädössä (Bandi ja muut, 2023, s. 20). Toisin kuin perinteiset neuroverkot, transformerit ovat erinomaisia laajojen tekstiaineistojen käsittelyssä tarkasti ja tehokkaasti, mikä on olennaista luonnollisen kielen tehtävissä (Warudkar & Jalit, 2024, s. 2–3).

Tekoälymallien, kuten GPT:n koulutus tapahtuu kahdessa päävaiheessa: esikoulutus ja hienosäätö. Esikoulutuksessa malli opetetaan ennustamaan seuraavaa sanaa lauseessa

aiempien sanojen perusteella. Tämä prosessi hyödyntää ohjaamatonta oppimista ja perustuu laajoihin merkitsemättömiin tekstiaineistoihin (Rane ja muut, 2024, s. 12). Hienosäätövaiheessa mallia muokataan tarkemmin käyttämällä ohjattua oppimista, jossa se koulutetaan tiettyihin tehtäviin tai erikoistuneisiin aineistoihin sen suorituskyvyn parantamiseksi tietyissä sovelluksissa (Nguyen-Duc ja muut, 2023, s. 5).

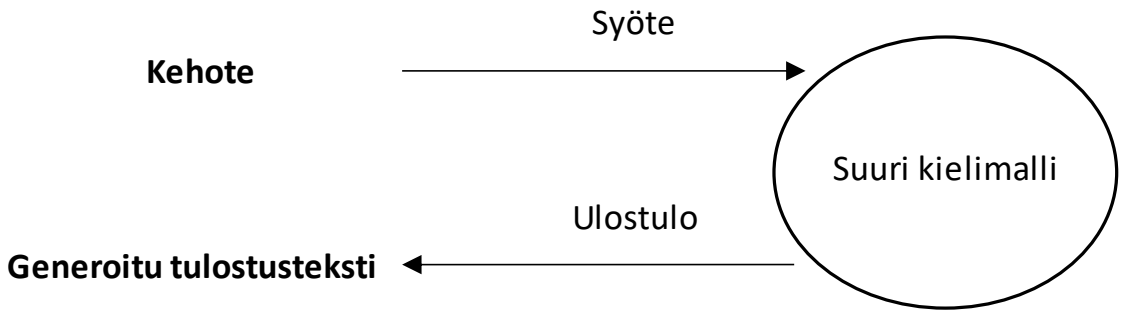
Merkittävä edistysaskel tekoälymallien parantamisessa on ihmispalautteeseen perustuvan vahvistusoppimisen käyttöönotto. Vahvistusoppiminen sisältää mallin kouluttamisen ihmisarvioijien antamien palautteiden avulla, joissa tuloksia arvioidaan niiden laadun perusteella. Tämä toistuva prosessi auttaa mukauttamaan mallin vastauksia paremmin ihmisten mieltymyksiin, vähentäen puolueellisia tai sopimattomia vastauksia (Rane ja muut, 2024, s. 12). Ihmispalautteeseen perustuvan vahvistusoppimisen lisääminen mahdollistaa sen, että GPT-pohjaiset mallit tuottavat luotettavampaa ja käyttäjän tarpeisiin räätälöityä tekstiä, parantaen niiden soveltuvuutta todellisiin käyttötarkoituksiin (Ebert ja muut, 2024, s. 2).

Generatiiviset vastakkaiset verkot (Generative Adversarial Networks, GAN) ovat koneoppimisen kehys ja merkittävä lähestymistapa tekoälyyn. Niiden uutuusarvo piilee siinä, ettei niiden toiminta ole voimakkaasti merkatun koulutusdatasta varassa. Lisäksi sen arkkitehtuuri poikkeaa huomattavasti perinteisistä syvistä neuroverkoista (Sengar ja muut, 2024, s. 4). GAN-verkot koostuvat kahdesta neuroverkosta: generaattorista ja diskriminaattorista, joita koulutetaan samanaikaisesti vastakkainasetteluun perustuvassa prosessissa. Generaattori opetetaan tuottamaan satunnaisesta kohinasta synteettistä dataa, joka muistuttaa korkealaatuista oikeaa dataa, kun taas diskriminaattorin tehtävänä on erottaa toisistaan generoidut ”väärennökset” ja todellinen data (Fu ja muut, 2025, s. 2). GANien etu tekoälyn sovelluksille on muun muassa sen kyky tuottaa realistista ja luotettavaa tekstiä (Rane ja muut, 2024, s. 2).

Variationaaliset autoenkooderit (Variational Autoencoders, VAE) ovat tekoälyssä käytettyjä syväoppimisen malleja luomaan uutta dataa alkuperäisestä syöttödatasta johdetuilla variaatioilla. Lisäksi ne pystyvät suorittamaan tavallisiin autoenkoodereihin liittyviä

tehtäviä, kuten kohinan poistamista. Kuten muissa autoenkoodereissa, variationaaliset autoenkoodereissa on kooderi ja dekooderi. Kooderin tehtävänä on oppia tunnistamaan ja eristämään tärkeät ja piilevät muuttujat koulutusdatasta, kun taas dekooderi puolestaan hyödyntää näitä piileviä muuttujia rakentaakseen alkuperäisen kaltaista dataa uudestaan (Ren, 2022, s. 1–2; Warudkar & Jalit, 2024, s. 2). VAEn koulutus on vakaampaa ja niiden tuottama latenttitila on helpommin tulkittavissa verrattuna GANeihin. Näitä malleja on sovellettu useilla aloilla, kuten tekoälyn kuvantuotannossa, paikkaavuuksien havaitsemisessa ja attribuutioppimisessa (Fu ja muut, 2025, s. 2).

Yleinen tapa vuoro vaikuttaa suurten kielimallien kanssa on kehotteiden käyttö (Prompt Engineering), jossa käyttäjät suunnittelevat tarkkoja kehotteita ohjatakseen kielimalleja tuottamaan haluttuja vastauksia tai suorittamaan erilaisia tehtäviä (Feuerriegel ja muut, 2024, s. 4). Tämä lähestymistapa mahdollistaa tekoälyjärjestelmän suorittaa tehtäviä, joihin sitä ei ole erikseen koulutettu, tuottaen usein arvokkaita ratkaisuja monenlaisiin tilanteisiin (Fischer & Lanquillon, 2024, s. 6–7). Tekoälymallien, kuten ChatGPT:n, musta laatikko (Black-Box) -luonne on herättänyt huolta, sillä niiden tarkka toteutus ja koulutusdata eivät ole julkisesti saatavilla. Tämä läpinäkyvyyden puute herättää kysymyksiä mallien luotettavuudesta, mahdollisista harhoista sekä siitä, miten ne käsittelevät ja tuottavat tietoa eri konteksteissa (Zubiaga, 2024, s. 3).



Kuva 6 Suuren kielimallin tuottama kehotesyöte (pohjautuu Belagatti, 2023)

3.1 Generatiiviset tekoälytyökalut ohjelmistokehityksen eri vaiheisissa

Tekoälytyökalut ovat nykyään laajasti käytössä eri ohjelmistokehityksen tehtävissä. Ohjelmistokehittäjät ovat pitkään hyödyntäneet jakamisalustoja, kuten Stack Overflow:ta, sekä hakukoneita, kuten Googlea, etsiessään koodausratkaisuja. Suuriin kielimalleihin perustuvat tekoälymallit indeksoivat valtavia määriä aiemmin kirjoitettua koodia ja yhteisön tuottamaa neuvontaa harjoitusaineiston perusteella, tarjoten vaihtoehtoisia ratkaisuja ohjelmointikehitykseen perinteisten jakamisalusten lisäksi (Ebert ja muut, 2024, s. 2). Ottaen huomioon tekoälyn kehityksen, uudet teknologiat, kuten OpenAI:n ChatGPT ja GitHub Copilot, ovat herättäneet huomattavaa mielenkiintoa ja saaneet nopeasti jalansijaa sekä käyttöä ohjelmistokehityksen parissa (Lima ja muut, 2024, s. 1). Liman ja muiden (2024) tutkimuksessa verrattiin automaatiota ohjelmistokehityksen prosesseissa ChatGPT:n ja ihmisen tekemän tuotosten välillä ja tutkimuksen tuloksena 65 % ohjelmistokehittäjistä koki ChatGPT:n vaikuttavan työtehtäviin, joissa on korkea automaatiotaso. Tekoälyn ekosysteemin kehittyessä yhä useammat ohjelmistokehittäjät luottavat näihin työkaluihin koodin tuottamisessa luonnollisen kielen kehotteiden avulla, vaikka perinteisillä hakukoneilla on oma asemansa (Sikand ja muut, 2024, s. 1–2).

Ohjelmistokehityksen prosessit seuraavat vakiintunutta prosessia ennen hyväksyntää ja ohjelmiston käyttöönottoa. Kehittäjät korjaavat virheitä, parantavat ohjelmistoa tai lisäävät uusia toiminnallisuuksia tuottamalla korjauspäivityksiä, jotka harkiten tarkastetaan ja hyväksynnän jälkeen otetaan osaksi koodipohjaa (Gonzalez-Barahona, 2024, s. 1–2). Ohjelmistotekniikan projekteissa on käytetty erilaisia ohjelmistokehitysmenetelmiä. Ketterä kehitys, scrum- ja vesiputousmalli ovat esimerkkejä yleisesti käytetyistä menetelmistä. Jokaisella ohjelmistokehitysmallilla on omat ainutlaatuiset ominaisuutensa, kuten kestävyys, luotettavuus, riippuvuus ja skaalautuvuus (Yahya & Maidin, 2022, s. 1–2).

Sauvola ja muut (2024) esittävät, että eri ohjelmistokehityksen elinkaaren prosessit voidaan hahmottaa eri skenaariona, jossa ihmisillä ja tekoälyllä on kukin omat tehtävänsä. Ensimmäisessä skenaariossa työkaluja on vain käytetty automatisointiin. Toisessa ske-

naariossa tekoälyä hyödynnettäisiin automatisoimaan toistuvat tehtävät ja auttaen ihmistä suunnittelussa, ongelmanratkaisussa ja päätöksenteossa. Kolmannessa vaiheessa tekoäly ottaa haltuunsa tietyt roolit, esimerkiksi testauksen ja ylläpidon, jättäen ihmiselle vastuun valvoa kokonaisprosessia. Neljännessä ja viimeisessä skenaariossa tekoäly on osana koko prosessissa ja ihmisen rooli rajoittuu valvomaan operatiivisia prosesseja kuten laadunvalvontaa ja tietoturvaa.

Gonzalez-Barahona (2024) esitti tulevaisuuden ohjelmistokehityksen siten, että tekoäly voisi automatisoida koodimuutokset, jolloin ihmisen tehtäväksi jäisi vain kuvata haluttu toiminnallisuus ja tarkistaa koodi. Myös testit voidaan luoda automaattisesti jättäen tekoälylle laadunvarmistamisen. Lopulta ihmiset määrittelisivät ohjelmiston vaatimukset antaen tekoälylle koodintuottamisen vastuun ja tehdä testaus. Tämän jälkeen ihmiset tarkastaisivat testien kattavuuden ennen hyväksyntää. Tämä sijoittuisi Sauvolan ja muiden (2024, s. 4) edellisessä kappaleessa esittämän neljännen vaiheen alle.

Sauvola ja muut (2024) viittaavat McKinseyn (2023) tutkimukseen, jonka mukaan tekoälyn hyödyntäminen parantaa ohjelmistokehittäjien tehokkuutta 20–50 % toistuvissa työtehtävissä, kuten dokumentoinnissa ja nopeiden prototyyppien toteutuksessa. Toisaalta tehtävissä, jotka vaativat syvempää ajattelua ja luovuutta, tekoälyn vaikutukset jäävät pieneksi. Pengin ja muiden (2023) tutkimuksessa tutkittiin Github Copilotin käyttöä ohjelmoinnissa, ja havaittiin, että Copilotin käyttö ohjelmistokehitystehtävissä osoitti merkittävää tuottavuuden kasvua, sillä tehtävien suorittamisaika nopeutui 55,8 %. Koehenkilöt myös saavuttivat myös paremman onnistumisprosentin ohjelmointitehtävissä aikaisempaan nähden.

Tekoälytyökalujen käyttöönoton myötä ohjelmistokehitys on muuttunut entistä nopeammaksi ja tehokkaammaksi, mutta samalla se tuo mukanaan haasteita, kuten liiallinen riippuvuus tekoälystä, tietoturvariskit ja mallien läpinäkyvyyden puute (Nasution & Onan, 2024, s. 22–23). Lisäksi tekoälyn tuottaman koodin laatu ja turvallisuus voivat vaihdella,

mikä korostaa ihmisten suorittaman tarkastuksen ja laadunvalvonnan merkitystä (Donvir ja muut, 2024, s. 7–8).

Tekoälyn kyky tuottaa haittaohjelmia ja viruksia, edellyttäen uusia ennakoivia ja ehkäiseviä lähestymistapoja, jotka on integroitava ohjelmistojen suunnitteluvaiheeseen ja sisäänrakennetuksi ominaisuudeksi kaikissa ohjelmistokehitysorganisaatioissa (Sauvola ja muut, 2024, s. 7). Tekoälyn käyttäjien pitää olla varuillaan syöttäessään sensitiivistä tietoa varsinkin yrityksen, sillä tietovuodet voivat olla mahdollisia ja arkaluontoisen tiedon päätyminen vääriin käsiin on mahdollista. Tämän vuoksi tekoälylle määrätyt säädökset ja politiikka on hyvä määritellä, mutta loppujen lopuksi vastuu on tekoälyn käyttäjällä, joka antaa komentoja (Nah ja muut, 2023, s. 10). Generoidun koodin riskeihin kuuluu herkkien, virheellisten tai vaarallisten osien kopioituminen koulutusaineistosta sekä virheiden syntyminen koodin siirrossa projekteista toiseen. Tämä voi johtaa tietoturvaongelmiin, kuten tunnettuihin virheisiin ja huonoihin käytäntöihin. Lisäksi avoimeksi jää kysymys tekoälyn tuottaman koodin tekijänoikeudellisesta asemasta (Negri-Ribalta ja muut, 2024, s. 17).

3.2 Generatiivisten tekoälytyökalujen omaksuminen

Tekoälyn nopean kehityksen myötä, erityisesti tekoälyavusteisessa ohjelmoinnissa, monet ohjelmistokehittäjät käyttävät jo näitä työkaluja tai harkitsevat niiden integroimista päivittäiseen koodaustyöhönsä. Näihin työkaluihin kuuluvat erikoistuneet koodin generointi- ja täydennystyökalut kuten GitHub Copilot, Tabninen ja muut, sekä yleisen suurikielimalli-pohjaiset työkalut, joilla on koodin generointikykyjä, kuten ChatGPT, Google BARD, Meta Llama ja muut (Sikand ja muut, 2024, s. 1). Näiden työkalujen avulla kehittäjät voivat nopeuttaa koodin kirjoittamista, vähentää rutiininomaista työtä ja parantaa ohjelmiston laatua tarjoamalla älykkäitä ehdotuksia ja tunnistamalla virheitä. Lisäksi tekoälyavusteiset työkalut voivat auttaa yksikkötestien generoinnissa ja dokumentaation luomisessa, mikä tehostaa ohjelmistokehitysprosessia. Tästä huolimatta haasteita, kuten tietoturva, mallien taipumus hallusinaatioihin sekä integroinnin käytännön haasteet, on

vielä ratkaistava ennen kuin nämä työkalut voivat saavuttaa täyden potentiaalinsa ohjelmistokehityksessä (Wang ja muut, 2024, s. 20–21).

Teknologian hyväksymismalli (Technology Acceptance Model, TAM) on laajasti käytetty viitekehys teknologian omaksumisen tutkimuksessa, ja sitä on sovellettu eri aloilla, kuten verkkokaupassa, mobiilipankkitoiminnassa ja terveydenhuollossa (Kanont ja muut, 2024, s. 3–4). Viimeaikaisissa tutkimuksissa on keskitytty tekoälyn hyväksyntään, erityisesti siihen, kuinka koettu hyödyllisyys (Perceived Usefulness, PU) ja helppokäyttöisyys (Perceived Ease of Use, PEOU) vaikuttavat käyttäjien sitoutumiseen tekoälytyökaluihin (Singh, 2024, s. 1; Kanont ja muut, 2024, s. 3–4). Singhin (2024) tutkimuksen mukaan, mikäli käyttäjät kokevat tekoälytyökalut hyödyllisiksi ja helppokäyttöisiksi, niiden käyttöönotto todennäköisesti kasvaa. Tämä viittaa siihen, että ohjelmistokehittäjät ja muut ammattilaiset ovat valmiimpia integroimaan tekoälyratkaisuja päivittäisiin työprosesseihinsa, mikä voi tehostaa tuottavuutta ja vähentää rutiininomaista työkuormaa. Samalla kuitenkin korostuu tarve ymmärtää tekoälyn rajoitukset, kuten tietoturvariskit ja mahdolliset virheelliset vastaukset, jotka voivat vaikuttaa sen laajempaan hyväksyntään.

Simamaren ja Edisonin (2024) tutkimuksessa todettiin, että tekoälyn käyttöönottoa rajoittavat huolet tietoturvasta, epätarkat ja harhaanjohtavat vastaukset sekä liiallinen kontekstiriippuvuus. Lisäksi sen tarjoamat ratkaisut voivat olla vanhentuneita, eikä sitä aina nähdä hyödyllisenä tietyissä työprosesseissa, kuten DevOps-automaatiossa. Yritykset saattavat myös epäröidä investointeja teknologiaan, jos sen tuomat hyödyt eivät ole selkeästi osoitettavissa. Näiden mallien musta laatikkoluonne aiheuttaa haasteita niiden tulkinnassa, mikä vaikeuttaa päätöksentekoprosessin ymmärtämistä ja virheiden paikantamista. Ohjelmistokehityksessä läpinäkyvyys on kuitenkin olennaista, sillä jokaisen koodirivin taustalla oleva logiikka on usein tärkeää hahmottaa. Lisäksi tekoälyn käyttö reaalimaailman sovelluksissa herättää eettisiä kysymyksiä. Koska mallit oppivat koulutusdatastaan, ne voivat huomaamattaan omaksua ja toistaa siinä esiintyviä vinoumia, mikä voi johtaa epäoikeudenmukaiseen tai syrjivään ohjelmistoon (Nguyen-Duc ja muut, 2023, s. 59).

Tulevaisuudessa tekoälyn kehitys suurissa kielimalleissa on mullistamassa tekoälyn ja luonnollisen kielen käsittelyn eri osa-alueita. Yksi keskeinen kehityssuunta on mallien tulkittavuuden parantaminen, jotta tekoälyjärjestelmistä tulisi luotettavampia ja läpinäkyvämpiä. Tutkijat työskentelevät aktiivisesti kehittääkseen menetelmiä, jotka selittävät, miten tekoälymallit tuottavat sisältöä. Tämä auttaa käyttäjiä ymmärtämään mallin päätöksentekoa sekä havaitsemaan mahdolliset vinoumat ja virheet (Warudkar & Jalit, 2024. s. 4–5). Fischerin ja Lanquillonin (2024) mukaan tekoälyn selittämättömyys ja läpinäkyvyys vaikuttavat järjestelmien luotettavuuteen ja käyttäjien luottamukseen. Koska mallien tuottama sisältö ei ole determinististä eikä aina täysin ymmärrettävissä, tämä voi aiheuttaa haasteita niiden käytössä kriittisissä sovelluksissa.

3.3 Vaatimusmäärittely ja suunnittelu

Termillä vaatimusmäärittely (Requirements Analysis, RE) tarkoitetaan järjestelmällistä prosessia, jossa vaatimuksia kehitetään iteratiivisesti ja yhteistyössä. Prosessissa analysoidaan ongelmaa, dokumentoidaan saadut havainnot useissa eri esitysmuodoissa ja varmistetaan saavutetun ymmärryksen oikeellisuus (Al-Msie’deen ja muut, 2021, s. 1). Vaatimusmäärittely on tärkeä osa ohjelmistokehityksen elinkaarta, koska se keskittyy järjestelmälliseen vaatimusten keräämiseen, analysointiin, määrittelyyn, validointiin ja hallintaan sekä toiminnallisten että ei-toiminnallisten vaatimusten osalta (Cheng ja muut, 2025, s. 2).

Ohjelmiston suunnittelu tapahtuu vaatimusmäärittelyn jälkeen. Suunnitteluvaihe alkaa ideoiden kehittämisellä kyseiseen ongelmaan, parhaiden ideoiden valinnalla tiiminä sekä valittujen ideoiden prototyyppien luomisella, jotta ne saadaan konkreettisempia. Tämä auttaa viestimään konseptin paljon nopeammin sidosryhmille ja potentiaalisille käyttäjille. Näin voidaan myös arvioida ideaa paremmin aiemmin määriteltyjen vaatimusten mukaisesti (Fischer & Lanquillon, 2024, s. 4). Suunnittelun tärkeys korostuu siten,

että ohjelmistosuunnittelu ja redundantti koodi voivat heikentää ohjelmiston suorituskykyä (Velaga, 2020, s. 3).

Aroran ja muiden (2023) mukaan tekoäly auttaa tunnistamaan tuntemattomia tekijöitä analysoimalla olemassa olevaa dokumentaatiota ja korostamalla epäselviä tai epävarmoja kohtia vaatimusmäärittelyn vaiheessa. Ne voivat auttaa vaatimusten täydentämisessä tai ehdottaa vaihtoehtoisia ideoita, joita vaatimusanalyytikot eivät ehkä muuten huomaisi, hyödyntäen laajaa koulutusdataansa ja yhteyksiä eri aiheiden välillä. Kehotteita käyttäen tekoäly jakaa monimutkaiset vaiheet pienempiin osiin, mikä tekee kielimallista tehokkaamman ja vaikuttavamman monimutkaisten ohjelmistokehitystehtävien suorittamisessa. Tämä antaa käyttäjälle mahdollisuuden tarkastaa tekoälyn tuottaman sisällön ja antaa lisäohjeita toteutusta varten (Wei, 2024, s. 3–4).

Arora ja muut (2023) painottivat, että oikeilla kehotteilla on suuri painoarvo saada kielimalli tuottamaan halutun lopputuloksen, vaatien kehotteiden tekijältä jo kyvyn tiedostaa haluttu lopputulos ja miten tekoälymalli sen luo. Kehotteiden hyöty tulee ilmi siinä tilanteessa, kun käyttäjä antaa mallin tuottaa välituotoksia ohjaten sitä oikeaan suuntaan noudattaen vesiputousmallin mukaista ohjelmistokehitysprosessia. Erityisesti ohjelmistoinsinööri kehottaa tekoälyä ensin tarkentamaan vaatimukset yksityiskohtaisiksi toiminnallisiksi vaatimuksiksi (Wei, 2024, s. 7).

Şimşek ja muut (2024) korostivat tutkimuksessaan, että tekoälyn vaikutus ohjelmistoarkkitehtien ja käyttöliittymäsuunnittelijoiden työhön auttaa tunnistamaan käyttäjien tarpeet, suunnittelemaan ja luomaan visuaalisia prototyyppejä sekä parantamaan käytettävyyttä. Ohjelmoijien osalta se puolestaan mahdollistaa koodipätkien luomisen kehotteiden avulla. Weiszin ja muiden (2024) tutkimuksessa havaittiin, että tekoälymallit voivat tuottaa useita erilaisia lopputuloksia, mutta samalla niihin voi sisältyä virheitä tai epätäydellisyyksiä, mikä edellyttää kriittistä tarkastelua ja käyttäjien vaikutusmahdollisuuksia lopputulosten hallintaan. Lisäksi tekoäly voi auttaa laajentamaan suunnitte-

lunäkökulmia ja mahdollistaa uusien ratkaisuvaihtoehtojen tutkimisen osana luovaa prosessia. Tekoälyn hyödyntämisessä ohjelmistosuunnittelussa korostuu myös osallistavien menetelmien merkitys, jotka voivat tukea tekoälyavusteista suunnitteluprosessia ja varmistaa, että syntyvät ratkaisut palvelevat käyttäjien tarpeita mahdollisimman hyvin. Vaatimusmäärittely ja suunnittelu ovat kuitenkin vahvasti ihmiskeskeisiä tehtäviä, joissa yksilön taidolla on merkittävä rooli. Tästä syystä tekoälyn vaikutus näihin tehtäviin on todennäköisesti pienempi kuin ohjelmoinnissa ja testauksessa (Sauvola ja muut, 2024, s. 7).

Nguyen-Ducin ja muiden (2023) tutkimuksessa arvioitiin, kuinka hyvin ChatGPT pystyy vastaamaan kysymyksiin, joissa kuvataan konteksti ja pyydetään suosittelemaan sopivaa suunnittelumallia. Tutkimus osoitti, että tällaisella työkalulla on potentiaalia toimia arvokkaana resurssina kehittäjien tukena. Kuitenkin tapauksissa, joissa ChatGPT antoi virheellisen vastauksen, laadullinen arviointi osoitti, että vastaus saattoi johtaa kehittäjiä harhaan. Tekoälymallien tuottamiin vääriin ja harhaanjohtaviin vastauksiin vaikuttaa käyttäjän antamat kehotteet ja sanalliset luokitukset, mikä herättää huolta niiden kyvykkyyksistä vaatimustenmäärittelyssä (Fazelnia ja muut, 2024, s. 2).

3.4 Ohjelmointi ja ohjelmistokehitys

Tekoälyn hyödyntämisellä pystytään tuottamaan suuria määriä koodia ja tämän lisäksi voi auttaa muun muassa luomaan tietorakenteita, API-integraatioita ja paljon muuta. Hyödyntämällä luonnollisen kielen käsittelyä, työkalut voivat tulkita ihmiskielen tekstisyötteitä ja tuottaa suoritettavaa koodia, mikä mahdollistaa kehittäjien keskittymisen enemmän koodin arkkitehtuuriin ja ominaisuuksien validointiin (Donvir ja muut, 2024, s. 3). Vuodesta 2020 lähtien on tehty laajaa tutkimusta suurten kielimallien käytöstä erilaisiin koodaukseen liittyviin tehtäviin, kuten koodin generointiin, täydennykseen, tiivistämiseen, hakuun ja kommenttien luomiseen (Nguyen-Duc ja muut, 2023, s. 23). Ohjelmoijat voivat käyttää olemassa olevia koodipätkiä luodakseen uusia toimintoja, joilla on

tietty käyttäytyminen, ja tekoälyavusteiset työkalut voivat järjestää ehdotukset niiden kehittäjälle sopivuuden perusteella (Velaga, 2020, s. 7–8).

Useita generatiivisia ohjelmistoalustoja on saatavilla markkinoilla, ja ne mahdollistavat yksinkertaisten ohjeiden muuntamisen tietokonekoodiksi. GitHub Copilot on saatavilla laajennuksina ohjelmistokehityksen kehitysalustoihin ja editoreihin, kuten Visual Studio Codeen, jotka tarjoavat koodin automaattisen täydennyksen käyttäjän tekstikomentojen perusteella. Koodin täydennyksen lisäksi Copilot auttaa muissa kehitystehtävissä, kuten koodin ymmärtämisessä, muutospyyntöjen parantamisessa sekä skriptauksen ja komentorivityökalujen tukemisessa (Ebert & Louridas, 2023, s. 3). Koodauksesta tulee tehtävä, jonka suorittavat tekoälyagentit, kun taas kehittäjät keskittyvät ilmaisemaan ohjelmien toiminnan luonnollisella kielellä (Gonzalez-Barahona, 2024, s. 2).

Sergeyukin ja muiden (2025) tutkimuksessa esitellyssä GitHubin kyselyssä, johon osallistui 500 kehittäjää ilman esihenkilöasemaa, todettiin, että 67 % oli käyttänyt tekoälytyökaluja sekä työssään että henkilökohtaisissa projekteissaan. Lisäksi 70 % vastaajista uskoi, että tekoälypohjaiset koodaustyökalut hyödyttävät heidän työtään, erityisesti taitojen kehittämisessä ja tuottavuuden parantamisessa. Lisäksi 81 % odotti näiden työkalujen parantavan tiimityöskentelyä, erityisesti tietoturvatarkastuksissa, suunnittelussa ja pariohjelmoinnissa. Koetun hyödyllisyyden lisäksi Paavilaisen ja muiden (2025) tutkimuksessa havaittiin, että tekoälyn kanssa työskenneltäessä on yleensä parempi antaa kehotteita pienemmissä vaiheissa monimutkaisen funktion kehittämisessä kuin kirjoittaa kaikki vaatimukset kerralla. Lisäksi tutkimuksessa todettiin, että suuret kielimallipohjaiset tekoälyt, kuten ChatGPT, auttaa koodin ymmärtämisessä selittämällä sen jokaisen iteraation jälkeen, mikä puolestaan lisäsi käyttäjien luottamusta koodin laatuun. Francen (2024) mukaan tekoälyä kehotteilla ohjaavaa ohjelmoijaa voidaan tulevaisuudessa pitää jopa omana työnimikkeenään ohjelmistokehityksessä.

Donvirin ja muiden (2024) tutkimuksen mukaan, vaikka tekoälyn työkalut tarjoavat monia etuja, ne tuovat mukanaan myös riskejä nykyiselle ohjelmistokehityksen alalle. Kehittäjät saattavat tulla liian riippuvaisiksi näistä työkaluista, mikä voi johtaa perustavanlaatuisten koodaustaitojen heikkenemiseen ja kyvyttömyyteen arvioida ratkaisuja kriittisesti. Tämä ongelma on erityisen merkittävä aloitteleville kehittäjille, jotka eivät ole vielä vahvaa osaamista alalta. Rippuvaisuus tekoälyn käytöstä voi myös demokratisoida koodausta ja madaltaa aloittamiskynnystä aloittaville ohjelmoijille tai uusien ohjelmointikielien oppijoilla, ne voivat myös johtaa siihen, että käyttäjiltä puuttuu kriittiset taidot koodin arviointiin (Hosseini ja muut, 2025, s. 9).

Paavilaisen ja muiden (2025) tutkimuksen tuloksena tekoälymallit eivät riittävän luotettavia, jotta loppukäyttäjät voisivat itsenäisesti luoda monimutaisia ohjelmistoja, erityisesti erikoissovelluksissa. Kuitenkin ohjelmointiosaamista omaavat henkilöt voivat hyödyntää suuria kielimalleja luodakseen suhteellisen monimutkaisia ohjelmia myös oman asiantuntemuksensa ulkopuolella. Viitaten edellisessä kappaleessa mainittuihin havaintoihin, tekoälyn hyödyntäminen lisää aloittavien ohjelmistokehittäjien kyvykkyyttä ja tehokkuutta lisätä koodin määrää, vaikka saattaisi johtaa riippuvaiseksi sen käytöstä. Gambacortan ja muiden (2024) tutkimuksen tuloksena aloittelevat ohjelmistokehittäjät paransivat 67 % tuottamansa koodin määrässä, kun taas kokeneimmilla työntekijöillä tekoälyn hyödyntämisestä ei ollut tilastollisesti merkittävä. Petrovskan ja muiden (2024) tutkimuksen mukaan 62,5 % vastaajista koki, että tekoälyn oikeanlainen hyödyntäminen työpaikoilla voi lisätä tuottavuutta ja helpottaa koodin luomista.

3.5 Testaus ja laadunvarmistus

Testaus- ja virheenjäljitysaputyökalut tukevat sovelluskehityksen laadunvarmistus- ja virheenkorjausvaiheita. Ne auttavat tunnistamaan ohjelmistokehityksen aikana virheet ja reunatapaukset sekä voivat ohjata ohjelmiston rakenteen tuotantoa varten. Nämä työkalut pystyvät oppimaan projektin kontekstista ja automatisoimaan yksikkö- ja integraa-

tiotestien luomisen uusille ominaisuuksille, mikä parantaa koodikannan testauksen kattavuutta ja vähentää kehittäjän työmäärää. Tukemalla testausta ne edistävät ohjelmiston koodin laatua ja luotettavuutta (Donvir ja muut, 2024, s. 2). Ohjelmiston laadunvarmistus on olennaista sen varmistamiseksi, että ohjelmistotuotteet täyttävät korkeat toiminnallisuuden, suorituskyvyn ja luotettavuuden vaatimukset. Ohjelmistojärjestelmien monimutkaistuessa testausmenetelmiä on kehitetty vastaamaan näihin haasteisiin (Pochu & Katham, 2022, s. 2). Khaliqin ja muiden (2022) tuottaman tutkimuksen mukaan ohjelmiston testaus ja laadunvarmistusvaihe kattaa 30–40 % kehittämiseen käytetyistä työtunneista ja kustantaa yli 50 % projektin kokonaiskustannuksista.

Tekoäly voi automatisoida testitapausten luomisen hyödyntämällä aiempaa testikoodia, vaatimuksia ja trendejä, mikä tehostaa prosessia ja parantaa kattavuutta. Lisäksi se voi tuottaa tarkempaa ja monipuolisempaa testidataa, mikä vahvistaa ohjelmiston luotettavuutta erilaisissa tilanteissa. Tekoäly voi myös tunnistaa virheitä analysoimalla koodia, testituloksia ja lokitietoja sekä oppien jatkuvasti ja parantaen virheiden havaitsemisen tarkkuutta (Garg & Sharma, 2023, s. 1). Manuaalinen yksikkötestien kirjoittaminen on työlästä, joten on kehitetty automaattisia testigenerointimenetelmiä, kuten hakupohjaisia, rajoitepohjaisia ja satunnaispohjaisia strategioita kattavuuden maksimoimiseksi. Vaikka nämä testit saavuttavat hyvän kattavuuden, niiden luettavuus ja merkityksellisyys jäävät heikommiksi kuin käsin kirjoitetuilla testeillä, minkä vuoksi kehittäjät eivät useimmiten käytä niitä sellaisenaan (Yuan ja muut, 2024, s. 20).

Yksi keskeisistä haasteista testaukseen on oraakkeliongelma, joka tarkoittaa tilanteita, joissa ei ole yksiselitteistä, oikeaa vastausta tai vertailukohtaa generoitujen tulosten arvioimiseksi. Tekoälyn järjestelmät tuottavat usein subjektiivisia ja monimuotoisia tuloksia (Aleti, 2023, s. 9). Wangin ja muiden (2024) tutkimuksessa havaittiin, että tekoälyn hyödyntäminen toi hyviä testituloksia varsinkin yksikkötestitapausten generoinnissa, testioraakkelin luomisessa, järjestelmätestisyötteiden generoinnissa, ohjelman virheenjäljityksessä ja ohjelman korjauksissa. Kuitenkin haasteita edelleen muun muassa korkean

testauskattavuuden saavuttamisessa, testioraakkeliongelman ratkaisemisessa, perusteellisten arviointien suorittamisessa sekä suurten kielimallien soveltamisessa reaalimaailman tilanteisiin.

Yuanin ja muiden (2024) tutkimuksessa tarkasteltiin ChatGPT:n kyvykkyyttä luoda yksikkötestejä keskittyen niiden oikeellisuuteen, riittävyyteen, luettavuuteen ja käytettävyyteen. Tutkimus osoitti, että ChatGPT:n luomat testit kärsivät yhä kääntövirheistä ja suoritushäiriöistä, mutta onnistuneet testit vastaavat käsin kirjoitettuja testejä kattavuuden ja luettavuuden osalta, ja toisinaan ne jopa kehittäjät suosivat niitä. Myös Wangin ja muiden (2024) tutkimuksessa havaittiin paljon kääntövirheitä yksikkötestauksessa, mikä saattaa johtua siitä, että ChatGPT ei ymmärrä koodin syvällistä rakennetta täysin. Vaikka laajalla koodiaineistolla toteutettu esikoulutus auttaa ChatGPT:tä hahmottamaan koodin syntaktisia sääntöjä, se on pohjimmiltaan todennäköisyyspohjainen merkkijonogeneraattori, minkä vuoksi ChatGPT:lle on haastavaa täysin ymmärtää koodin syvällisiä sääntöjä. Yaacovin ja muiden (2024) tutkimuksessa ehdotettiin ratkaisuksi vaikeasti havaittaviin virheisiin ja mallin oikeellisuuden varmistamiseen eristämällä ja pitämällä kunkin vaatimuksen toteuttavan koodin erillisenä ja ytimekkäänä. Tällä tavoin voidaan helpottaa virheiden havaitsemista ja varmistaa koodin laatu.

Simaremaren & Edisonin (2024) tutkimuksen haastateltava käytti tekoälyä apuna ohjelmistotestauksessa luomalla näennäisdataa testejä varten, joka on aikaa vievää ja varsinkin jos data on monimutkaista, mutta tärkeä vaihe ohjelmiston testauksessa. Tähän tarkoitukseen tekoälyn työkalut soveltuvat hyvin. On todettu, että tekoälyn kasvava autonomia voi tuoda merkittäviä etuja kehittäjille käsittelemällä monimutkaisempia testaustehtäviä, mikä vähentää manuaalista työtä ja asiantuntemuksen tarvetta (Paavilainen ja muut, 2025, s. 12).

4 Tutkimusmenetelmä- ja aineisto

Tutkimusta, joka toteutetaan tieteen edistämiseksi järjestelmällisesti keräämällä, tulkitsemalla ja arvioimalla tietoa suunnitelmallisella tavalla, kutsutaan tieteelliseksi tutkimukseksi (Çaparlar & Dönmez, 2016, s. 1). Tutkimuksia voi tehdä eri tavoin. Laadulliset menetelmät ovat nousseet keskeisiksi työkaluiksi syvällisen ymmärryksen saavuttamisessa ja monimutkaisten ilmiöiden analysoinnissa. Niiden avulla voidaan saavuttaa laaja näkökulmia sekä käytännönläheisiä keinoja sen eri ulottuvuuksien hallintaan. Toisin kuin määrällinen tutkimus, joka keskittyy numeeriseen dataan ja tilastolliseen analyysiin, laadullinen tutkimus syventyy ihmisten kokemuksiin ja näkökulmiin, tuoden esiin kontekstit ja vivahteet, jotka saattavat jäädä tilastollisessa tarkastelussa huomaamatta (Lim, 2024, s. 2).

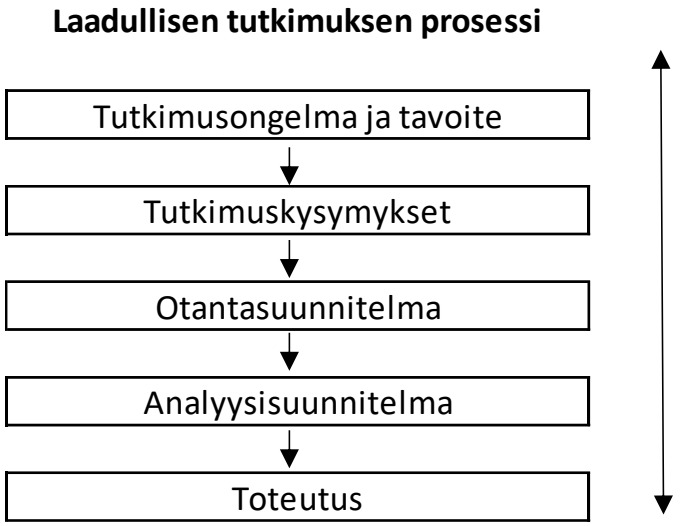
Tässä luvussa käsitellään tutkimuksen toteutuksessa käytettyä menetelmää sekä aineiston keruuta ja analysointia. Lisäksi perustellaan valittujen menetelmien soveltuvuus tutkimusaiheeseen, sekä tarkastellaan aineiston käsittelyä ja tulkintaa. Tutkielma toteutetaan puolistrukturoidun teemahaastattelun avulla, jonka tarkoituksena on tuoda esiin haastateltavien ohjelmistokehittäjien henkilökohtaisia kokemuksia tekoälyn hyödyntämisestä ohjelmistokehityksessä. Tulevissa luvuissa käsitellään tutkimusmenetelmiä tarkemmin, joiden avulla pyritään vastaamaan asetettuihin tutkimuskysymyksiin ja saavuttamaan tutkimuksen tavoitteet.

4.1 Laadullinen tutkimus

Tämä pro gradu -tutkielma toteutetaan kvalitatiivisena eli laadullisena tutkimuksena. Tutkimusmenetelmä antaa ymmärryksen ohjelmistokehittäjien kokemuksista ja työkuvaansa vaikuttavien muutoksien ilmiöistä. Aikaisemmissa luvuissa käydyn teoriaosuuden tarkoitus on tukea tutkimusta muiden tieteellisten tutkimusten pohjalta ja saada johtopäätös, jonka perusteella voidaan arvioida, tukeeko tämän pro gradu -työn tutkimuksen tulokset teoriaosuuden tutkimusten havaintoja.

Laadullinen tutkimus viittaa yleisesti tutkimusmenetelmiin, joissa tuloksia ei ilman määrällistä mittausta tai tilastollisia analyysejä (Hamilton & Finley, 2019, s. 1). Kyse ei ole pelkästään metodologisesta valinnasta, vaan sitoutumisesta yhteiskunnallisten ilmiöiden syvälliseen tarkasteluun, mikä mahdollistaa tutkijoille yhteyden luomisen tutkittavien subjektiivisiin kokemuksiin (Lim, 2024, s. 2). Laadullisen tutkimuksen ytimessä ovat avoimet kysymykset, joiden vastauksia ei ole helppo ilmaista numeroina, kuten "miten" ja "miksi." Tämän avoimen luonteensa vuoksi laadullisen tutkimuksen suunnittelu ei usein ole yhtä lineaarista kuin määrällisen tutkimuksen suunnittelu (Tenny ja muut, 2022, s. 1).

Tennyn ja muiden (2022) mukaan laadullinen tutkimus auttaa tutkijoita saamaan syvällisempää ymmärrystä reaalimaailman ongelmista. Sen sijaan, että kerättäisiin numeerisia datapisteitä, puututtaisiin tilanteisiin tai sovellettaisiin interventioita kuten määrällisessä tutkimuksessa, laadullinen tutkimus auttaa muodostamaan hypoteeseja määrällisen datan tarkempaa tutkimista ja ymmärtämistä varten. Subjektiivisten kokemusten, näkökulmien ja merkitysten tutkimisessa, joita yksilöt liittävät sosiaaliseen maailmaansa, laadullinen tutkimus korostaa avoimuutta ja hyödyntää monipuolisia menetelmiä, kuten avoimia kysymyksiä, syvähaastatteluja, fokusryhmähaastatteluja ja osallistuvaa havainnointia, varmistaen ilmiöiden kokonaisvaltaisen tarkastelun (Lim, 2024, s. 3). Haastattelun etuna on sen joustavuus, ja sitä voidaan toteuttaa monin eri tavoin. Tämä monipuolisuus tekee siitä soveltuvan erilaisiin tutkimustilanteisiin ja mahdollistaa sen hyödyntämisen monenlaisiin tarkoituksiin (Puusa ja muut, 2020).



Kuva 7 Laadullisen tutkimuksen prosessi (pohjautuu Hossain, 2011)

4.2 Teemahaastattelu

Eskolan ja Suorannan (1998) mukaan laadullisen tutkimuksen haastattelutyypit voidaan jakaa neljään eri kategoriaan: avoin haastattelu, strukturoidut ja strukturoimattomat haastattelut sekä teemahaastattelu. Avoimessa haastattelussa keskustellaan vapaasti ilman tarkkaa rakennetta. Strukturoidussa haastattelussa kysymykset ja vastausvaihtoehdot ovat haastattelijalle valmiiksi määrätty. Puolistrukturoitu haastattelu on lähellä strukturoitua haastattelua, mutta haastattelussa kysymykset ovat valmiiksi määrättyjä ja haastateltavat voivat vapaasti vastata kysymyksiin. Teemahaastattelussa eteneminen nojaa ennalta määriteltyihin, keskeisiin teemoihin ja niihin liittyviin tarkentaviin kysymyksiin. Sen etuna on se, että haastattelussa voidaan tarkentaa ja syventää kysymyksiä haastateltavien vastauksiin perustuen (Jowsey ja muut, 2021, s. 1).

Tuomen ja Sarajärven (2018) mukaan teemahaastattelussa korostuvat erityisesti ihmisten omat tulkinnat, heidän antamat merkitykset sekä se, miten merkitykset syntyvät vuorovaikutuksessa. Teemahaastattelussakaan ei voi kysellä ihan mitä tahansa, vaan tavoitteena on etsiä tutkimuksen tarkoituksen, ongelmanasettelun tai tutkimustehtävän asettamien kysymysten mukaisia merkityksellisiä vastauksia. Näin varmistetaan, että

haastattelussa esitetyt kysymykset ovat tutkimuksen tavoitteiden kannalta merkityksellisiä

Tuomen ja Sarajärven (2018) mukaan teemallisen haastattelun analyysin tavoitteena on tunnistaa teemoja, eli aineistosta nousevia merkityksellisiä tai kiinnostavia toistuvia kuvioita, ja käyttää näitä teemoja tutkimuskysymykseen vastaamiseen tai jonkin ilmiön tarkasteluun. Kyse ei ole pelkästä aineiston tiivistämisestä, vaan hyvä teemallinen analyysi tulkitsee ja jäsentää aineistoa, jotta siitä voidaan saada syvällisempää ymmärrystä (Maguire & Delahunt, 2017, s. 3).

Tuomi ja Sarajärvi (2018) ilmaisevat, että teemahaastattelun avoimuus heijastuu siihen, kuinka teemoihin sisältyvien kysymysten osuus suhteessa tutkimuksen teoreettiseen viitekehykseen vaihtelee. Toisin sanoen, havaintojen perustaminen voi liukua intuitiivisten ja kokemusperäisten näkökulmien hyväksymisestä hyvin tiukasti ennalta määrättyjen kysymysten käyttöön.

4.3 Aineiston hankinta ja haastateltavien valinta

Tämän pro gradu -tutkielman aineisto on kerätty puolistrukturoiduilla teemahaastatteluilla, joissa kaikille osallistujille esitettiin samat kysymykset ennalta määrätyssä järjestyksessä ennalta määrättyjen teemojen pohjalta. Puolistrukturoidussa haastattelussa on ominaista, että kaikille haastateltaville esitetään valmiiksi laaditut kysymykset, kuitenkin vastaajilla oli vapaus muotoilla vastauksensa omin sanoin ilman tiukkoja rajoitteita (Eskola & Suoranta, 1998). Tämä menetelmä valittiin, koska sen katsottiin tukevan parhaiten tutkimuksen tavoitteita. Haastattelukysymykset laadittiin huolellisesti siten, että ne ohjasivat keskustelua tutkimusongelman kannalta esitettyjen teemojen pohjalta, mutta jättivät samalla tilaa vastaajien omille kokemuksille ja tulkinnoille. Näin varmistettiin, että kerätty aineisto heijastaa monipuolisesti ohjelmistokehittäjien näkemyksiä ja yksilöllisiä havaintoja aiheesta.

Tavoitteena on valikoida haastateltaviksi henkilöitä, joilla on ohjelmistokehityksestä kokemusta työelämässä vähintään kolmen vuoden ajan. Näin varmistetaan, että he ovat työskennelleet ajanjaksona, jolloin tekoälyn työkalut eivät olleet vielä yleisesti saatavilla ja ohjelmistokehittäjien apuvälineinä oli perinteiset välineet kuten Google ja kehittäjäfoorumit (esim. Stack Overflow). Haastateltavat tulee valita harkinnanvaraisesti ja ideana on, että tutkija asettaa ennalta määrätyt kriteerit, joiden perusteella valikoidaan haastateltavat henkilöt. Tätä voidaan kutsua tarkoituksenmukaiseksi ja harkinnanvaraiseksi näytteeksi (Saaranen-Kauppinen & Puusniekka, 2006). Tuomen ja Sarajärven (2018) mukaan on tärkeää, että haastateltaviksi valikoituvat henkilöt tietävät tutkittavasta ilmiöstä mahdollisimman paljon ja heillä on kokemusta asiasta. Näin haastatteluista on mahdollista saada käsityksen, miten tekoälymallit ovat vaikuttaneet työkuvaansa.

Laadullisessa tutkimuksessa haastateltavien määrän ei ole tarkoitus olla ennalta määrätty luku, vaan olennaista on se, että haastateltavilla on riittävästi tietämystä tutkittavasta ilmiöstä. Tuomen ja Sarajärven (2018) mukaan haastateltavien määrä ei ole kriittinen tekijä vaan se, että haastateltavilla on tietämystä tutkivasta asiasta. Eskola ja Suoranta (1998) ehdottavat ratkaisuksi käsitettä saturaatio eli kyllästyminen. Saturaatio tarkoittaa tilannetta, jossa aineistosta ei enää nouse uutta merkittävää tietoa ja samat teemat sekä havainnot alkavat toistua tiedonantajien kertomuksissa. Tällöin lisäaineisto ei tuo tutkimusongelman kannalta olennaista lisäarvoa. Tämän lisäksi Eskolan ja Suorannan (1998) mukaan riippuen tutkimuksesta ja haastateltavien tietämyksestä saturaation määrä vaihtelee, eikä sille ole määrätty tiettyä lukumäärää tai pistettä.

Haastateltaviksi valikoitui ohjelmistokehittäjät, jotka ovat tällä hetkellä ohjelmistokehittäjinä työelämässä ja joilla on vähintään kolme vuotta työkokemusta sekä riittävä tietämys käsitellyistä aiheista. Haastateltavien kokemus ohjelmistokehityksestä ulottuu pidemmälle kuin asetettu kriteeri työkokemusvuosien suhteen muun muassa harrastuneisuuden ja koulutuksen puolesta lisäten tietämystä ja kokemusta esiteltyjen teemojen

pohjalta. Tämä parantaa tutkimuksen luotettavuutta, sillä haastateltavat pystyvät tarjoamaan syvällisempiä ja monipuolisempia näkökulmia käsiteltyihin aiheisiin. Lisäksi heidän laajempi osaamisensa vahvistaa aineiston kattavuutta ja varmistaa, että tutkimuksessa huomioidaan myös kokemuksen kautta hankitut tiedot ja taidot.

Haastattelut toteutettiin Microsoft Teams -sovelluksen välityksellä. Haastateltavat on nimetty taulukon yksi mukaisesti H1, H2 jne., ja taulukossa on kerrottu haastateltavien nykyinen työnimike sekä työkokemusvuodet ohjelmistokehityksen parissa työelämässä. Videohaastatteluiden vastaukset litteroitiin kirjalliseen muotoon, mikä mahdollisti syvällisemmän analyysin litterointia varten helpoittain vastausten vertailua keskenään. Haastattelut kestivät keskimäärin 15–20 minuuttia ja haastatteluissa pyrittiin pysymään ennalta määrättyjen teemojen rajoissa ilman, että keskustelu johtaisi muualle.

Haastateltavien henkilöllisyydet ja yrityksen nimet pidetään salassa haastateltavien suostumuksella. Haastateltavat ovat tietoisia, että heidän työnimikettänsä ja työkokemusvuosia käytetään tutkimuksessa.

Taulukko 1 Haastateltavat

Haastateltava	Työnimike	Työvuosia
H1	Ohjelmistokehittäjä	4
H2	Data-insinööri	7
H3	Koneoppimis-insinööri	6
H4	Ohjelmistosuunnittelija	4
H5	Ohjelmistokehittäjä	4
H6	Data-insinööri	6
H7	Ohjelmistokehittäjä	5

Haastateltavista työnimikkeellä ohjelmistokehittäjä työskentelee 42 %, data-insinöörinä 28 % ja ohjelmistosuunnittelijana 14 % sekä koneoppimisinsinöörinä 14 % vastanneista. Yhteensä haastateltavilla on 36-vuotta työkokemusta, ja heidän keskimääräinen kokemuksensa on viisi vuotta.

4.4 Aineiston analyysi

Tämän pro gradu -tutkielman aineiston analyysin lähtökohtana on määritetyt tutkimuskysymykset, jotka ohjaavat sitä, millaisia vastauksia etsitään ja mihin tekijöihin aineistossa kiinnitetään huomiota. Sisällönanalyysin päämääränä on esittää tutkimusmateriaali pelkistetyssä, tiiviissä ja yleistettävässä muodossa. Analyysin keskeinen askel on muodostaa luokkia, joiden lopputuloksena syntyy tyypillisesti luokkia tai kategorioita, jotka kuvaavat tutkittavaa ilmiötä ja sen keskeisiä teemoja (Elo ja muut, 2022, s. 3).

Tässä tutkimuksessa analyysimenetelmäksi valittiin teemoittelu, joka on sisällönanalyysin muoto, jossa laadullinen aineisto jaotellaan keskeisten aiheiden mukaisesti. Saaranen-Kauppisen ja Puusniekan (2006) mukaan menetelmä auttaa tunnistamaan tutkimusongelman kannalta merkitykselliset teemat ja jäsentämään aineistoa selkeämpään muotoon. Tässä tutkimuksessa keskeiset teemat ovat luvussa kolme käsitellyt teemat:

- Generatiivisen tekoälyn omaksuminen
- Vaatimusmäärittely ja suunnittelu
- Ohjelmointi ja ohjelmistokehitys
- Testaus ja laadunvarmistus
- Tietoturva- ja luottamusriskit

Edellä esitettyjen teemojen lisäksi haastateltavilta kysyttiin generatiivisen tekoälyn liittyvistä riskeistä, jotta saataisiin kattavampi ymmärrys heidän näkemyksistään ja mahdollisista huolenaiheistaan. Tämä kysymys auttoi kartoittamaan, miten haastateltavat kokevat tekoälytyökalujen hyödyntämisen, sekä mikä saattaa olla yrityksen politiikka ja näkemys niiden käyttöön.

Laadullisessa tutkimuksessa aineiston hankinta ja analysointi ovat sidoksissa toisiinsa. Koska tutkija toimii tutkimusinstrumenttina, hänen osallistumisensa aineiston keruuseen käynnistää samalla analyysiprosessin (Puusa ja muut, 2020). Aineiston tarkaste-

lussa keskitytään ainoastaan siihen, mikä on teoreettisen viitekehyksen kannalta merkityksellistä, vaikka samaa tutkimusta voitaisiin lähestyä useista eri näkökulmista. Tästä syystä laadullisessa analyysissä on olennaista tiivistää raakahavainnot mahdollisimman ytimekkääksi havaintojoukoksi, minkä pohjalta voidaan tehdä laajempia tulkintoja keskeisten teemojen ympärillä (Alasuutari, 2011). Tämän pro gradu -tutkimuksen analyysi keskittyy ensisijaisesti haastatteluista nouseviin havaintoihin, joita tuetaan teemoihin liittyvillä tutkimuksilla ja niissä todettujen tutkimustulosten valossa.

5 Tutkimuksen tulokset

Tämän luvun tarkoituksena on käydä läpi tutkimuksen tulokset ja verrata tuloksia muihin valmiiksi toteutettuihin tutkimuksiin alueaiheesta. Seuraavassa luvussa käydään tutkimuksen yhteenveto ja johtopäätökset. Tulokset esitetään luvussa 4.4 käytyjen teemojen mukaisesti, ja havainnot on tuettu aihealueeseen liittyvän kirjallisuuden tuella.

Tutkimusten tulosten esittely aloitetaan teemojen järjestyksen mukaisesti. Teemaan generatiivisten tekoälytyökalujen omaksumiseen on lisätty aliteema sen koetusta hyödyllisyydestä, mikä mahdollistaa kokonaiskuvan tekoälytyökalujen hyödyllisyydestä haastateltavien kesken. Luvussa 4.4 mainittu aliteema, joka käsittelee riskeihin liittyvistä kysymyksistä, on tässä luvussa käsitelty omana kokonaisuutenaan, vaikka luvussa 3 käsitellyt teemat sisältävät myös riskeihin liittyviä näkökulmia.

5.1 Omaksumiseen vaikuttavat tekijät ja koettu hyödyllisyys

Tässä luvussa tarkastellaan haastateltavien kokemusta tekoälytyökalujen käyttöönotosta ja niistä koetusta hyödyllisyydestä. Haastateltavilta kysyttiin, miten ovat käyttäneet ja ottaneet käyttöön tekoälytyökaluja sekä millaisia haasteita he ovat kokeneet omaksumisessa. Lisäksi selvitettiin, kuinka hyödyllistä työkalut on koettu ohjelmistokehityksessä ja miten tekoälytyökalut tukevat työskentelyä. Haasteltaessa nousi esiin vahvasti esiin se, että työkalujen koettu hyödyllisyys liittyy vahvasti rutiininomaisten työtehtävien tehokkuuden parantumiseen ja tietoisuuteen, että työkalut auttavat tietyissä tehtävissä.

Generatiivisen tekoälyn perustuvien teknologioiden nopean kehityksen ja yleistyvän käytön myötä ohjelmistotekniikan akateeminen ja teollisuusyhteisö ovat käymässä läpi merkittäviä muutoksia. Monet kokevat tekoälyn olevan yhtä mullistava tekijä ohjelmistokehitykselle kuin internetin yleistyminen (Treude & Storey, 2025, s. 1). Ohjelmistokehitysyhteisön tulisi suhtautua tekoälyyn perustuvien työkalujen käyttöönottoon siten, että keskitytään erityisesti siihen, miten tekoäly voi tukea kehitysprosessia ilman, että se lisää

tarpeetonta monimutkaisuutta tai haittaa työnkulkua (Yabaku, 2023, s. 16). Torka ja Albayrak (2024) toteavat, että 70 % ammattikehittäjistä käyttää tai aikoo käyttää tekoälyteknologioita koko ohjelmistokehityksen elinkaaren vaiheissa. Tekoälypohjaisia työkaluja käytetään jo nykyään muun muassa projektisuunnittelussa, vaatimusmäärittelyssä, vaatimusanalyysissä, koodin generoinnissa, testauksessa, muutostenhallinnassa, optimoinnissa, virheenjäljityksessä ja dokumentoinnissa sekä koodin selittämisessä. Sergeyukin ja muiden (2025) mukaan 77 % ohjelmistokehittäjistä on myönteinen näkemys tekoälyn hyödyntämisestä ohjelmoinnissa.

Haastatteluissa kävi ilmi, kuinka paljon haastateltavat hyödyntävät tekoälytyökaluja, erityisesti OpenAI:n ChatGPT ja IDE-ympäristöön integroitua GitHub Copilot olivat suosittuja. Pienten koodipätkien luonti ja funktioiden selitys olivat yleistä. Tekoälytyökalut toimivat tukena, mutta haastavammissa ongelmatilanteissa niiden antamat ehdotukset eivät ole yleensä avuliaita, vaan saattavat johtaa vääriin ja toimimattomiin ratkaisuihin. Tekoälyn käyttö on siis hyödyllinen työväline, mutta niitä ei voi käyttää täysin kritiikittömästi tai korvaamaan asiantuntevaa kehittäjää.

> *“Käytän nykyään generatiivista tekoälyä muun muassa dataputkissa käsittelyssä datan siivoamiseen ja kategorisointiin. Työskentelyn tukena käytän generatiivisen tekoälyn työkaluja suhteellisen varovaisesti, sen tuottama koodi on usein suhteellisen huonolaatuista eikä kauhean ns. production valmista tavaraa varsinkin, jos pyyntö tai ratkaisu on suhteellisen uniikki tai monimutkainen. Idean sparrailu kaveriksi generatiivinen tekoäly sopii ihan hyvin, kunhan muistaa myös itse kyseenalaistaa sen vastauksia.” (H6)*

Kuten luvussa 3.2 on kuvattu, tekoälytyökalujen omaksumisen hyöty näkyy rutiininomaisten työtehtävien tehostamisessa ja avussa koodin generoinnissa. Kehittäjien, työkalujen ja loppukäyttäjien roolit ovat keskeisenä muutoksen kohteena. Nykyään kehittäjät käyttävät tekoälytyökaluja koodin tuottamiseen, mikä siirtää heidän keskittymisensä koodin kirjoittamisesta ratkaisujen suunnitteluun ja hienosäätöön (Treude & Storey,

2025, s. 2). Ulfsnesin ja muiden (2024) tutkimuksessa tarkasteltiin, kuinka tekoälytyökalut ovat muuttaneet ohjelmistokehittäjien työkuvaa. Tutkimuksessa korostettiin kehotesyötteiden merkitystä, sillä niiden avulla ohjeistetaan tekoälyä saavuttamaan halutun tuloksen ilman, että se antaa vääriä vastauksia. Selkeiden ohjeiden myötä ohjelmistokehittäjät voivat keskittyä tuotoksen arviointiin ja itse vaikeiden ongelmien ratkaisuun (Ulfsnes ja muut, 2024, s. 11–12).

Tekoälyä hyödynnetään laajasti ideointiin, ongelmanratkaisuun ja luovaan ajatteluun. Näissä tekoäly auttaa ohjelmistokehittäjiä paljon jättäen aikaa muihin tehtäviin ja ajatustyöhön (Jackson ja muut, 2024, s. 9). Aartin (2024) tutkimuksessa tekoälypohjaisten työkalujen koettiin parantavat koodin laatua älykkäiden ehdotusten, varmistaen parhaiden käytäntöjen noudattamisen. Oikean työkalun valinta kuitenkin riippuu projektin erityistarpeista ja kehitysympäristöstä. Haitaksi on todettu, että tekoälytyökalut tuottavat virheellistä tai optimaalisuudeltaan puutteellista koodia, mikä voi johtaa luotettavuusongelmiin. Kehittäjien on edelleen tarkistettava ja testattava tekoälyn luoma koodi huolellisesti varmistaakseen sen oikeellisuuden ja ylläpitääkseen koodin laatua.

Luvussa 3.2 esitelty teknologian hyväksymismalli tarjoaa näkökulman uusien teknologioiden käyttöönottoon. Singhin (2024) mukaan koettu helppokäyttöisyys on keskeinen tekijä, sillä tekoälyn omaksuminen ja sitoutuminen on todennäköistä, jos käyttäjä tiedostaa ja uskoo sen parantavan lopputuloksia. Tämän lisäksi hyödyllisyys ja helppokäyttöisyys antamalla konkreettisia ratkaisuja sekä kontekstuaalisia ehdotuksia ja selityksiä, mikä helpottaa monimutkaisten ongelmien ratkaisun tekoälyn avulla.

Tekoälyn hyödyllisyydestä vastaajat ovat yhteneväisiä. Se nähdään ohjelmistokehityksessä erittäin hyödyllisenä, erityisesti tehokkuuden parantamisessa, rutiinitehtävien automatisoinnissa ja oppimisen tukemisessa. Sen suurimmat edut liittyvät syntaksin muistamisen helpottamiseen, koodin kirjoittamisen nopeuttamiseen ja ongelmanratkaisun tehostamiseen. Vastaajien mukaan kehittäjät voivat keskittyä vaativampiin tehtäviin eikä

tarvitse käyttää aikaa rutiininomaisiin koodipätkien luomiseen tai syntaksivirheiden korjaamiseen. Teknologian hyväksymismallin mukaisesti vastaajat olivat samaa mieltä tekoälytyökalujen helppokäyttöisyydestä ja hyödyllisyydestä.

> *“Auttaa yksinkertaisten, mutta aikaa vievien asioiden hoitamisessa yllättävän paljon.” (H6)*

> *”Voisin melkeinpä heittää, että generatiivisen tekoälyn hyödyntäminen tuplaa tehokkuuteni eikä varmasti mene pahasti metsään.” (H2)*

> *“Koen, että tekoäly tehostaa työskentelyäni merkittävästi, ehkä jopa noin 10–30 %.” (H7)*

> *”Se nopeuttaa minua yksinkertaisen ja toistuvan ohjelmiston kirjoittamisessa huomattavasti. Ajatustyötä tai arkkitehtuuria vaativassa työssä ei juurikaan.” (H3)*

Gambacortan ja muiden (2024) tutkimuksen mukaan tekoälytyökalujen hyödyntäminen kasvatti ohjelmistokehittäjien tehokkuutta 55 % generoidun koodin määrässä ja kolmasosa tästä oli suoraan suurien kielimallien generoimaa koodia, jättäen kehittäjille kaksi kolmasosaa ajastaan muihin tehtäviin. Tätä tukevat haastatteluissa ilmenneet merkittävät tehokkuuden parantumiset, jotka näkyvät paitsi pelkästään tuotetun koodin määrässä, myös tehokkuudessa ja ajattelua sekä suunnittelua vaativissa tehtävissä. Kuten luvussa 3.4 todettiin, tekoälytyökalujen käyttö tarjoaa merkittäviä etuja eri osa-alueilla, kuten tuottavuuden parantamisessa, oppimisessa ja yhteistyön tehostamisessa, vaikka sen käyttöön liittyykin erilaisia näkökulmia ja kokemuksia. Yhden haastateltavan mielestä tekoälytyökalut tekevät hänestä ”superkoodaajan”, vaikka luvussa 3.4 todettiin työkaluista olevan suurimpia hyötyjä kokemattomille ohjelmoijille, toisin kuin kokeneemmille.

> *“Olen koodannut jo 10-vuotta. Matkan varrella on tullut opittua kaikenlaista. Tekoäly on tehnyt minusta ns. superkoodarin. Se helpottaa ja nopeuttaa koodaamista huomattavasti.” (H5)*

Haastateltavien kokemusten perusteella tekoälyn käytöstä ei ole haittaa, vaan sen vaikutus on positiivinen, sillä se parantaa tuottavuutta, helpottaa oppimista ja tukee ongelmanratkaisua erityisesti rutiinitehtävissä.

5.2 Rooli vaatimusmäärittelyssä ja suunnittelussa

Vaatimusmäärittelyprosessi pyrkii määrittämään ohjelmistopalvelut ja -rajoitukset, mikä on olennainen prosessi asiakkaan tarpeiden ja odotusten täyttämiseksi (Al-Msie’deen ja muut, 2021, s. 1). Ohjelmistosuunnittelu puolestaan edustaa uudelleenkäytettäviä ratkaisuja toistuviin ohjelmistosuunnittelun ongelmiin, ja niiden oikea soveltaminen on keskeistä vankkojen ja ylläpidettävien järjestelmien kehittämisessä. Tekoälyteknologiat voivat auttaa kehittäjiä tunnistamalla automaattisesti sopivia suunnittelumalleja järjestelmän vaatimusten ja olemassa olevan koodipohjan perusteella (Alenezi & Akour, 2025, s. 6–7).

Haastateltavilta kysyttiin, miten tekoäly on vaikuttanut vaatimusmäärittelyssä ja suunnittelun prosesseihin. Vastausten perusteella yhteiseksi teemaksi muodostui se, että tekoälyn hyödyntäminen vaatimusmäärittelyn ja suunnittelun tukena, erityisesti ideoiden tuottamisessa, vaihtoehtojen arvioinnissa ja teknisten ratkaisujen hahmottamisessa. Vaikka osa vastaajista ei käytä tekoälyä vaatimusmäärittelyssä, sitä pidetään yleisesti hyödyllisenä työkaluna ideoinnissa, suunnittelussa ja ratkaisuvaihtoehtojen arvioinnissa.

> *“Generatiivisen tekoälyn työkalut tuottavat nopeita vaatimusmäärittely ja suunnitteluarvioita, jotka pitkälti perustuvat netistä eniten osumia löytyneisiin artikkeleihin ja kaavioihin. Tekoäly on oiva apu suunnittelussa ja vaatimusmäärittelyssä, mutta sen tuotoksia pitää aina tarkastella kriittisesti.” (H4)*

> *"Tekoäly auttaa näkemään muita ratkaisuja. Välillä ideat voivat loppua kesken, jolloin on hyvä pallotella tekoälyn kanssa, ja saada täysin uusia näkemyksiä. Se auttaa myös arvioimaan paremmin erilaisia ratkaisuja." (H5)*

Chengin ja muiden (2023) mukaan vaatimusmäärittely on keskeinen osa ohjelmistokehityksen elinkaarta, sillä 37 % ohjelmistoprojektien epäonnistumisista kehitysvaiheissa johtuu huonosta vaatimusmäärittelystä ja 68 % virheistä huomataan vasta projektien loppuvaiheessa. Kuten luvussa 3.3 käytiin läpi, vaatimusmäärittelyvaiheessa ohjelmistokehittäjän tulisi jo tietää lopputulos ja hyödyntää tekoälyä sen tukena. Aroran ja muiden (2023) tutkimuksessa todettiin, että tekoälytyökalut voivat parantaa useita vaatimusmäärittelyn tehtäviä automatisoimalla, tehostamalla ja tukemalla ihmisten kykyjä. Kuitenkin kehotesyötteillä on suuri merkitys johdattaa kielimalli oikeaan tulokseen. Saleemin ja muiden (2024) tutkimuksessa havaittiin, että tekoälyn tuotokset paranevat vaatimusmäärittelyssä sitä mukaan, mitä tarkempia ja asiantuntijatietoisempia kehotesyötteitä annetaan. Kuitenkaan niiden hyöty ei ole yhtä merkittävä kuin koodin generoinnissa.

> *"En ole käyttänyt tekoälyä vaatimusmäärittelyyn, mutta ideoinnissa ja vaihtoehtojen arvioinnissa se on ollut hyödyllinen työkalu. Tämä pätee erityisesti silloin, kun koen ettei minulla ole riittävästi pohjatietoa erilaisten ideoiden muodostamiseen tai valistuneiden päätösten tekoon. Toki silloin pyrin vahvistamaan myös jostain toisesta lähteestä, ettei tekoäly puhu ihan puuta heinää." (H7)*

> *"Itse hyödynnän paljon erilaisiin kehityksen aikaisiin teknisten toteutusvaihtoehtojen generoimiseen ja arvioimiseen. Lisäksi uutta kokonaisuutta aloittaessa monesti käyn konseptin läpi ja pyydän kehitysehdotuksia." (H2)*

Sauvolan ja muiden (2024) tutkimuksen mukaan vaatimusmäärittely ja suunnittelu ovat erittäin ihmiskeskeisiä työtehtäviä ohjelmistokehityksen elinkaaressa, jolloin tekoälyn

rooli jää pienemmäksi. Tämä käy ilmi haastateltavien vastauksissa, joissa tekoälyn rooli on pieni ja soveltuu enemmän havainnointiin ja ideointiin. Haastateltavat suhtautuvat kriittisesti tekoälytyökalujen tuottamiin sisältöihin ja mieluummin käyttävät perinteisiä menetelmiä, kuten verkkoselaimen kautta haettua faktapohjaista tietoa varmistaakseen sisällön luotettavuuden.

5.3 Vaikutus ohjelmointiin ja ohjelmistokehitykseen

Perinteisesti ohjelmointi ja ohjelmistokehitys ovat nojanneet pitkälti manuaaliseen koodaamiseen ja ihmiskeskeisiin tarkistusprosesseihin, mutta tekoälyn integrointi ohjelmistokehitykseen on avannut ennennäkemättömiä mahdollisuuksia automaattiseen koodin luomiseen (Modi, 2024, s. 2). Ohjelmistosovellusten kasvavan monimutkaisuuden ja lyhyempiin kehityssykleihin kohdistuvan paineen vuoksi kehittäjien tuottavuuden parantamiseen ja laadukkaamman koodin tuottamiseen tarvitaan yhä enemmän uusia työkaluja (Sajja ja muut, 2024, s. 1). Vaikka koodin generointityökalut vähentävät merkittävästi manuaalista ohjelmointia ja siihen liittyviä mahdollisia virheitä, tekoälyn ohjaus voi toimia turvaverkkona samalla tarjoten joustavaa, tehokasta ja mukautettavaa automaatiota (Velaga, 2020, s. 4).

Haastateltavilta kysyttiin, miten tekoälytyökalut ovat konkreettisesti vaikuttaneet ohjelmointiin ja ohjelmistokehitykseen. Haastateltavien vastausten perusteella tekoäly auttaa koodipätkien tuottamisessa, tehostaa ohjelmointia ja toimii ohjelmistokehityksessä tukena.

> *”Generatiivinen tekoäly on nopeuttanut ohjelmointia huomattavasti juuri yksinkertaisten koodinpätkien ehdottamisella, yleisten algoritmien automaattisella generoimisella, tyyppien ja syntaxin ehdottamisella ja edelleen.” (H3)*

> *”Generatiivinen tekoäly auttaa minua koodin generoinnissa paljon.” (H1)*

> *”Välillä ne pystyvät toteuttamaan ominaisuuden/ominaisuuksia hyvin lyhyessä ajassa. Featureen, johon ennen saattoi mennä aikaa useampi tunti, voi nyt tehdä muutamassa minuutissa.” (H5)*

Kuten edellä mainituissa lainauksissa todetaan, tekoäly tukee merkittävästi koodin generoinnissa ja tukee ohjelmointia konkreettisesti, erityisesti koodipätkien ja yleisten algoritmien luomisessa. Työkalut, kuten GitHub Copilot, voivat luoda valmista koodipohjaa, ehdottaa täydennyksiä ja jopa kirjoittaa kokonaisia funktioita luonnollisen kielen kuvausten perusteella. Tämä nopeuttaa merkittävästi kehitysprosessia ja vähentää kehittäjien kognitiivista kuormitusta (Aarti, 2024, s. 2). Kehitysprosessin tehostumisen lisäksi tekoälytyökalujen käyttö edistää kehittäjien oppimista. Francen (2024) mukaan sopivilla kehotteilla käytettynä tekoälymallit voivat parantaa kehittäjien koodin ymmärrystä ja parantavat oppimista. Hänen mukaansa yleinen näkemys on, että käyttäessään tekoälymallien luomaa koodia kehittäjien tulisi varmistaa, että he ymmärtävät sen täysin ennen sen käyttöönottoa, sillä koodin sitouttava kehittäjä on vastuussa siitä, mukaan lukien tekoälymallin tuottamat virheet.

> *”Käytän säännöllisesti ja päivittäin OpenAI:n chattia kehityksen apuna. Tällä hetkellä pääasiallisesti GPT-4o:ta. Hyödynnän sitä paljon koodin selittämiseen sekä uuden koodin luontiin. Se on erittäin suuri apu datan työstöön varsinkin Pythonilla liittyvissä jutuissa. Erityisesti monimutkaisten datastruktuureihin liittyvien transformaatioiden tekemisessä se on suorastaan yllättävän hyvä.” (H2)*

> *”Käytän pääasiassa ChatGPT-4:ää ja GitHub Copilotia. Copilot on joskus hyvä ehdottamaan esimerkiksi yleisiä apufunktioita, lajittelualgoritmeja jne. Se on myös hyvä välillä tyypittämään esimerkiksi TypeScript-koodia ja tyypitettyä Pythonia. Kuitenkin täysin uuden koodin kehittämisessä harvoin on paljon hyötyä, varsinkin kun koodi on monimutkaista.” (H3)*

Tekoälyn käyttö tukee ohjelmointia merkittävästi. Esimerkiksi ChatGPT ja GitHub Copilot voivat luoda valmista koodipohjaa kehotesyötteiden avulla ja kirjottaa kokonaisia koodipätkiä. Luvun 5.1 H2- ja H6 vastausten perusteella suurimmat konkreettiset hyödyt näkyvät monimutkaisten tietorakenteiden käsittelyssä ja datan muokkaamisessa, jossa se voi tarjota tarkkoja ja hyödyllisiä ratkaisuja. Ohjelmistokehitys on luovaa ja ajattelua vaativaa, joten tekoälyn avulla tuotettu koodi vähentää kehittäjien kognitiivista kuormitusta jättäen aikaa muille tehtäville.

Sergeyukin ja muiden (2025) mukaan kehittäjät kohtaavat myös paljon ongelmia ohjelmoinnissa tekoälyä hyödyntäen. Näihin haasteisiin kuuluvat muun muassa tekoälyn tuottaman koodin epätarkkuus, virheiden huomaamatta jääminen sekä koodin luettavuuden ja ylläpidettävyyden heikkeneminen. Kehittäjät käyttävät lähes 50 % koodausajastaan vuorovaikutuksessa suurten kielimallien kanssa, ja tästä ajasta 35 % kuluu tekoälyn antamien ehdotusten tarkastamiseen. Tästä huolimatta kehittäjät arvostavat manuaalisen koodaamisen työmäärän vähenemistä ja monet ohjelmistokehittäjät ovat sitä mieltä, että tekoälytyökalujen ansiosta he pystyivät keskittymään enemmän korkean tason suunnittelupäätöksiin ja laadunvarmistukseen sen sijaan, että aikaa olisi kulunut matalan tason koodaustehtäviin (Raghi ja muut, 2024, s. 6).

> *”Joskus tosin generatiivisen tekoälyn käyttö on myös saattanut hidastaa projektia, jos tekoälyn luoma koodi on luonut huomaamattomia bugeja ohjelmistoon. Lisäksi täysin uuden koodin kehittämisessä harvoin on paljon hyötyä, varsinkin kun koodi on monimutkaista. Liiallinen GitHub Copilotin ehdotuksiin luottaminen johtaa usein vääriin ratkaisuihin ja rikkinäiseen tai bugiseen koodiin. Se on hyvä työkalu esimerkiksi toistuvien tai hyvin samanlaisten koodinpätkien "kopioimiseen" tai esimerkiksi kokoelmien rakentamiseen.” (H3)*

Raghin ja muiden (2024) mukaan tekoälytyökalujen tarjoamaa tietoa ei aina tule ottaa sellaisenaan, vaan se tulisi mahdollisuuksien mukaan varmistaa toisesta lähteestä, sillä ne voivat ajoittain keksiä virheellisiä tietoja. Serguykin ja muiden (2025) tutkimuksessa

ollaan samaa mieltä virheellisen tiedon antamisesta, 7,8 % tutkimuksen vastaajista oli sitä mieltä, että tekoälyn vastaukset eivät ole hyödyllisiä. Kuitenkin luvussa 5.1 käytyjen vastausten perusteella hyödyt ylittävät haitat. Tekoälyn vastausten epätarkkuus ja niiden käyttäminen koodikannassa voi ilmetä monimutkaisiksi virheiksi verrattuna siihen, että kehittäjät itse rakentaisivat ja testaisivat toimivia ratkaisuja alusta asti (Sajja ja muut, 2024, s. 5).

Tekoälyn yleistyminen saattaa vaikuttaa paljon harjoittelijatason työpaikkoihin. Modin (2024) mukaan McKinsleyn tuottaman tutkimuksen perusteella yritysten panostaminen tekoälyn hyödyntämiseen ja nykyisten työntekijöiden kouluttamisessa on johtanut siihen, että aloitustason työtehtävistä 35 % on vähentynyt, mutta samalla tekoälyasiantuntijoiden roolit ovat kasvaneet 85 %. Tekoälyn yleistyminen on siis lisännyt kehittäjien osaamisvaatimuksia. Osaamiseen liittyen, Gambacortan ja muiden (2024) tutkimuksen yhteenvetona ohjelmointi parani yli 50 % aloittelevien tai junioritason työntekijöiden keskuudessa. Kokeneempiin työntekijöihin vaikutus ei ollut tilastollisesti merkittävä, pääasiassa siksi, että seniorikehittäjät käyttivät tekoälyä harvemmin kuin juniorit.

> ”*Generatiivisen tekoälyn tuottama koodaustaso vastaa noin pääpiirteittäin osaavaa ja pätevää junior tason ohjelmoijaa ja se soveltuu niin sanotusti ”rivikoodaajaksi” oikein hyvin.” (H6)*

5.4 Vaikutus testauksessa ja laadunvarmistuksessa

Ohjelmistotestaus on olennainen osa ohjelmistokehityksen elinkaarta, ja sen tavoitteena on varmistaa, että kehitettävä ohjelmistotuote toimii ja täyttää sille asetetut käyttövaatimukset. Historiallisesti testausvaihe ohjelmistokehityksessä on ollut aikaa vievää ja intensiivinen prosessi, joka kuluttaa paljon resursseja (Thakur ja muut, 2024, s. 1). Hyödyntämällä kehittyneitä koneoppimisalgoritmeja tekoälytyökalut voivat automaattisesti luoda korkealaatuisia koodinpätkiä, suorittamaan älykästä testausta virheiden tunnistamiseksi varhaisessa vaiheessa sekä tehostamaan jatkuvan integraation työnkulkuja.

Nämä edistysaskeleet eivät ainoastaan paranna tehokkuutta, vaan myös mahdollistavat sen, että kehittäjät voivat keskittyä monimutkaisten ongelmien ratkaisuun ja luovien tehtävien hoitamiseen (Mary & Hoover, 2024, s. 2).

Ohjelmistotuotteiden toiminnallisuuden, suorituskyvyn ja luotettavuuden varmistamiseksi laadunvarmistus on olennaista, että järjestelmien monimutkaistuessa testaustekniikat ovat kehittyneet vastaamaan näihin haasteisiin. Laadunvarmistuksessa automaattisella ja manuaalisella testauksella voidaan ratkaista monimutkaiset ohjelmiston laadunvarmistuksen haasteet (Pochu & Katham, 2022, s. 2). Tekoäly tarjoaa uuden lähestymistavan testitapausten suunnitteluun ja testikoodin generointiin, ja sen avulla voidaan saavuttaa perinteisten automatisoitujen testaustyökalujen kanssa saman laadun (Nazari, 2024, s. 11). Thakur ja muut (2024) totesivat, että tekoälyn hyödyt ilmenisivät automaattisten testitapausten luomisessa, jossa tekoäly analysoisi koko koodipohjaa ja loisi testiskenaarioita sen pohjalta.

Haastateltavilta kysyttiin, kuinka tekoälytyökalut ovat tukeneet laadunvarmistus- ja testausvaiheessa sekä mitä etuja ja riskejä he ovat kohdanneet niiden hyödyntämisessä omassa työssään. Haastateltavat kokivat, että tekoälyn käyttö on hyödyllistä etenkin automaattisten testiskenaarioiden luomisessa ja testausvaiheen nopeutumisessa. Toisaalta sen hyöty testauksessa ei ole yhtä suuri kuin esimerkiksi koodin generoinnissa, sillä testitulokset vaativat jatkuvaa tarkistusta, eikä luottamus testituloksiin ei ole suuri.

> *”Käytän ChatGPT:tä automaattisien testiskenaarioiden luomisessa, ei oikeastaan muuten.” (H1)*

> *“Monesti testiskenaarioiden kirjoittaminen käsin on työlästä, koska siihen liittyy paljon toistuvaa boilerplate-koodia ja esimerkiksi testausdatan alustusta, luontia ja siivoamista testin ajettua. Tässä generatiivinen tekoäly monesti nopeuttaa tekemistä ja vapauttaa kaistaa kehittäjältä testikoodin puuduttavasta kirjoittamisesta itse testien suunnitteluun.” (H7)*

> *”Generatiivinen tekoäly osaa luoda yksikkötestauksia kohtuullisen hyvin pelkän pohjalogiikan pohjalta, mutta näissäkin on ollut usein vajavaisuuksia ja kehittämisen aihetta. On silti erittäin hyödyllistä, että testipohjat luodaan automaattisesti.” (H4)*

Haastatteluiden vastausten perusteella tekoälyn hyödyntäminen testauksessa ja laadunvarmistuksessa on hyödyllistä erityisesti testiskenaarioiden, testipohjien ja yksikkötestien automaattisessa luomisessa, mikä puolestaan nopeuttaa testausprosessia ja vähentää manuaalista kirjoittamista. Lisäksi se auttaa testidatan generoinnissa ja hallinnassa, mikä nopeuttaa testisettien muodostamista. Tekoälyn avulla voi myös nopeuttaa testien suunnittelua ja toteutusta, vapauttaen kehittäjien aikaa muihin tehtäviin. Hnatushenkon ja Pavlenkon (2024) tutkimuksen mukaan nykyisellä tekoälyn käytöllä ohjelmistokehityksessä nykyisellä kehitystasolla on täysin mahdotonta luoda kattavia ja hyväksyttäviä testejä. Tavoitteellisempaa on testien luominen yhteistyössä kehittäjän kanssa avustavassa tilassa. Avustavana roolina Wang ja muut (2024) esittivät tekoälyn käytettävyyden yleensä vain tietyissä testauksen elinkaaren osissa, erityisesti testauksen keskivaiheilla ja myöhemmissä vaiheissa, kuten yksikkö- ja järjestelmätestauksessa, ja pääasiassa vain funktionaaliseen testaukseen.

Haastatteluiden perusteella tekoälyn käyttöä testauksessa havaitut haitat ylittävät hyödyt toisin kuin ohjelmoinnissa koettiin erittäin hyödylliseksi ja hyödyt ylittivät haitat. On kuitenkin huomattava, että haastateltavilla ei ole yhtä paljon kokemusta testitapausten luomisesta verrattuna ohjelmointiin, joten vastaukset eivät anna täydellistä kuvaa sen todellisesta hyödyntämisestä testausprosesseissa.

> *”Olen käyttänyt generatiivista tekoälyä labeloimaan testidataa erillisille tekoälymalleille. Tämä on nopeuttanut testisettien luomista, ja siten myös mallien testausta, mutta tietenkin labelointi voi joskus olla vajanaista, eikä esimerkiksi saman tasoista, kun ihmisen luoma.” (H3)*

> *“Generatiivisen tekoälyn tuottamat testitapaukset täytyy kuitenkin lukea huolella läpi, jotta voi varmistua testin testaavan oikeita asioita ja välillä virheitä pääsee omastakin seulasta läpi.” (H7)*

Aletin (2023) mukaan ohjelmistotestauksen ala kohtaa uusia haasteita kehittyvien tekoälyjärjestelmien myötä, mikä tekee uusien testaustapojen kehittämisestä välttämätöntä. Keskeisenä haasteena on oraakkeliongelma, joka liittyy tekoälyjärjestelmien luomien tulosten oikeellisuuden määrittämiseen. Luvussa 3.5 esitelty oraakkeliongelman käsite tulee ilmi monissa tekoälyn käyttöä ohjelmistotestauksen testausskenaarioissa. Fan ja muut (2023) puolestaan esittävät, että oraakkeliongelmaa voitaisiin välttää hyvin suunnitelluilla kehotteilla, jotka parantavat ohjelmistotestien laatua ennustamalla ja vähentämällä testien epäluotettavuutta sekä paljastamalla todennäköisiä virheitä.

> *“Generoidut dokumentoinnit ja testit parantavat ohjelmiston laatua. Olen nähnyt paljon puutteita ohjelmistokehityksen dokumentoinnissa ja testauksessa. Tekoälyllä voi generoida hyvinkin nopeasti koko koodin kattavia testejä. Etuna tässä on, että ohjelmisto on laadukkaampaa, mutta samalla herää huoli, että kuinka hyviä testit lopulta ovat. Sillä lopputulos riippuu täysin henkilöstä, joka tekoälylle kehotteen on luonut.” (H5)*

Vaikka tekoäly voi parantaa testauksen tehokkuutta, ihmisten asiantuntemus on edelleen kriittistä testauksen tavoitteiden määrittelyssä ja tulosten arvioinnissa (Hnatushenko & Pavlenko, 2024, s. 8). Tekoälyn kehittyessä sen sovellusalue ohjelmistotestauksessa laajenee ja laadunvarmistusprosessit kehittyvät entistä edistyneemmiksi. Yritykset, jotka ottavat käyttöön tekoälypohjaisen testauksen, ovat paremmassa asemassa kehittämään korkealaatuista ohjelmistoa nopeammin ja kustannustehokkaammin (Thakur ja muut, 2024, s. 10).

Vastausten perusteella tehokkuuden lisääntyminen ja ajansäästö ovat ilmeisiä, mutta testituloksiin suhtaudutaan varovaisesti. Tekoälyn tuottamien tulosten tarkistaminen ja ohjaaminen oikeiden kehotteiden avulla kulkevat käsikädessä saavuttaakseen riittävän tarkkuuden ja luotettavuuden.

5.5 Tietoturva- ja luottamusriskit

Tekoälytyökalut on koulutettu suurilla tietomäärillä, jotka ovat usein kerätty suodattamattomista verkkolähteistä, esimerkiksi koodausfoorumeilta kuten GitHub, Stack Overflow ja HuggingFace, joissa saattaa olla virheellistä ja puutteellista koodia. Tämän seurauksena tekoälymallit ovat alttiita datan myrkyttämiselle, eli hyökkäykselle, jossa haitallisia näytteitä syötetään opetusaineistoon, mikä voi johtaa haavoittuvan koodin luomiseen. Koska tekoälypohjaisten koodigeneraattoreiden käyttö laajenee jatkuvasti, on yhä tärkeämpää ymmärtää niiden mahdollinen vaikutus ohjelmistojen haavoittuvuuksiin. Tekoälytyökalut voivat tahattomasti toistaa samoja kaavoja, jotka alun perin aiheuttivat nämä haavoittuvuudet (Cotroneo ja muut, 2024, s. 1; Tihanyi ja muut, 2023, s. 1). Tekoälymallit tarjotaan usein palveluna internetin kautta, mikä tuo mukanaan paitsi riskin syötteiden ja tulosteiden tahattomasta vuotamisesta tiedonsiirron aikana, myös mahdollisuuden, että palveluntarjoaja pääsee käsiksi dataan ja mahdollisesti käyttää sitä mallin jatkokoulutukseen (Alt ja muut, 2025, s. 13).

Muissa edellä esitellyissä luvuissa on noussut esiin epävarmuus ja luottamus tekoälyn tuottamia vastauksia kohtaan. Tämän myötä on aiheellista tutkia, miten haastateltavat kokevat nämä haasteet sekä millaisia keinoja he käyttävät tekoälyn tuottaman tiedon arviointiin, varmentamiseen ja hyödyntämiseen omassa työssään. Haastateltavilta kysyttiin millaisia tietoturvaan ja luottamukseen liittyviä riskejä he kokevat tekoälytyökalujen käytössä ja miten näitä riskejä pyritään ehkäisemään.

Haastateltavien vastausten perusteella tekoälyn käyttö yritysdatan kanssa nähdään merkittäviä tietoturvariskinä, erityisesti luottamuksellisen koodin ja liiketoiminnalle kriittisten algoritmien suojaamisen osalta. Tietoisuus, että tekoälytyökalujen käyttämät koodiehdotukset ovat peräisin julkisista tietolähteistä ja suoraan kopioidun koodin kooditietokantaan sisältää riskejä. Lisäksi työntekijöiden koulutusta mahdollisista riskeistä pidetään tärkeänä.

> *"Näen kyllä riskejä esimerkiksi, kun Copilotilla on pääsy yksityisiin repositoreihin GitHubissa. Jos yrityksillä on repositorioissaan yrityksen liiketoiminnalle kriittistä ohjelmistoa, voi Copilot teoriassa ehdottaa kilpailijalle vastaavaa koodia. Kuitenkin Copilot ymmärtääkseni toimii statistiikan mukaan valitsemalla normaalijakauman keskikohdalle osuvaa koodia, ja niin sanotusti todennäköisintä seuraavaa koodipätkää, on hyvin epätodennäköistä, että uniikki kriittinen ohjelmisto päätyy Copilotin ehdotukseksi. Tietenkin on ongelmallista copy-pasteta koodia myös esim. GPT-malliin, sillä emme voi olla täysin varmoja, mitä mallia tarjoava palvelu tallentaa tietokantaansa. Usein kopioidussa koodissa saattaa olla salausavaimia, tai muuta kriittistä dataa, joka voi siten päätyä vääriin käsiin. Siksi on tärkeää käyttää vain luotettavia käyttöliittymiä, ja ymmärrän hyvin huolen esimerkiksi DeepSeek mallin mahdollisista riskeistä." (H3)*

> *"Se riippuu siitä mitä yritys tekee. Jos yritys käsittelee henkilötietoja, silloin GDPR voi aiheuttaa haasteita. Varsinkin kun suurin osa tekoälyistä pyörii Yhdysvalloissa palvelimilla. En myöskään päästäisi tekoälyä käsiksi yrityksen keskeisiin algoritmeihin, jotta ne eivät vuotaisi (jos kyseessä tekoäly, joka pyörii muulla kuin yrityksen omalla palvelimella). Hyvällä työntekijöiden koulutuksella voi toki joitain vuotoja estää, mutta paras tapa yritykselle on ajaa omaa tekoälyä omilla palvelimilla." (H5)*

Nguyen-Ducin ja muiden (2023) mukaan yrityksillä ei ole aina resursseja kouluttamaan omaa suurta kielimalli-pohjaista agenttia sisäiseen käyttöön, ja koulutuksen rajoittaminen vain tiettyihin projekteihin saattaa johtaa kielimallien tarjoaman yleisen tietämyksen menetykseen, joka taas voi heikentää generoidun koodin laatua. Humphreysin ja muiden (2024) mukaan yritysten itse kouluttama kielimalli ei ole turvassa hyökkäyksiltä. Yritys, joka kouluttaa suurta kielimalliaan luottamuksellisella tiedolla, saattaa altistua hyökkäyksille, joissa herkkää tietoa voi vuotaa ulkopuolisille. Eirasin ja muiden (2024) mukaan avoimen lähdekoodin tekoälymallit voivat lisätä turvallisuusriskejä antamalla haitallisille toimijoille paremmat mahdollisuudet hyödyntää niitä väärin. Toisaalta avoimen lähdekoodin mallit ovat olleet ja todennäköisesti pysyvät keskeisessä asemassa turvallisuuden, oikeudenmukaisuuden ja yhdenvertaisuuden tutkimuksessa, minkä vuoksi ne ovat välttämättömiä turvallisen kehityksen ja käyttöönoton kannalta.

> *“Riskit ovat olemassa ja niihin suhtaudutaan alalla vakavasti. Työnantajallani on käytössä firman sisäinen agentti, joka käyttää vastauksissaan julkista pohjadataa muttei vie käyttäjien syötteitä eteenpäin järjestelmän ulkopuolelle.” (H4)*

> *”Riippuu täysin mitä generatiivista tekoälyä käyttää ja pyöriikö malli yrityksen omassa infrassa vai ei. Mikäli malli pyörii yrityksen omilla koneilla, data ei liiku yrityksen oman infran ulkopuolelle kolmansille osapuolille, eikä näin ollen siihen liity suurempaa tietoturvariskiä kuin pilvipalvelujen käytöstä yleensäkin. Suurin osa yritysten käyttämistä generatiivisen tekoälyn ratkaisuista ei hyödynnä eikä tallenna sinne lähetettyä dataa, vaan data pysyy niin sanotusti siilossa, tästä esimerkkinä muun muassa OpenAI:n API:t yrityksille.” (H6)*

> *”Kyllähän se on aina riski, että tietoa vuotaa ulkopuolelle. Suoraan koodikantaan kytkettävien plugarien käytössä kannattaa varmistua, ettei laita koodikantaan mitään luottamuksellista tietoa, kuten käyttäjätunnuksia, salasanoja tai API-avaimia. Toki esimerkiksi Codeium väittää, ettei hyödynnä käyttäjien koodia tai muuta informaatiota millään lailla tai välitä sitä eteenpäin. Täysin varma ei*

tietysti käyttäjänä voi olla, ettei näin ole. Olisi myös mahdollista esimerkiksi hallinoida Codeium deployment itse, jolloin mitään dataa ei ulkopuolelle edes lähtisi.” (H7)

Haastateltavien vastausten perusteella tekoälyteknologian käyttöön liittyy paljon tietoturva- ja luottamusriskejä, mutta alalla suhtaudutaan niihin vakavasti ja harkinnanvaraisesti. Riskit riippuvat tapauskohtaisesti käytetystä mallista sekä siitä, pyöriikö se yrityksen omassa infrastruktuurissa ja paikallisesti toimiva malli ei siirrä dataa ulkopuolisille. Kuten edellisessä kappaleessa mainittiin, sisäinen ympäristö ei ole täysin turvassa tietomurroilta, ja on olemassa mahdollisuus, että tietoa vuotaa, varsinkin koodiin integroitavien työkalujen kautta. Kuitenkin käyttäjällä on loppupeleissä vastuu mihin käyttää tekoälyn tuottamaa sisältöä, eikä sille anneta arkaluontoista tietoa.

6 Yhteenveto ja johtopäätökset

Tutkimustulokset osoittivat selkeästi, millainen on tekoälyn vaikutus ohjelmistokehityksen prosesseihin nykyisten ohjelmistokäytäntöjen ja teknologioiden puitteissa. Teoreettisessa viitekehyksessä käsitellyt osiot tekoälystä ja sen vaikutuksesta ohjelmistokehityksessä muodostavat tutkimusosuuden perustan. Tekoälyn hyödyntäminen ohjelmistokehityksen prosesseissa on merkittävä, ja tutkimuksen tuloksista voidaan koota niihin liittyvät hyödyt ja haitat. Alla olevassa taulukossa on esitetty yhteenveto aiemmin käsiteltyjen teemojen pohjalta mitkä ovat haastateltavien kokemukset tekoälyn hyödyistä ja haitoista tietyissä ohjelmistokehityksen elinkaaren vaiheissa.

Taulukko 2 Koetut hyödyt ja haitat eri prosesseissa

Teemat	Koetut Hyödyt	Koetut haitat
Generatiivisen tekoälyn omaksumiseen vaikuttavat tekijät ja koettu hyödyllisyys	-Rutiininomaisten työtehtävien automatisointi ja ajan vapauttaminen muihin tehtäviin. -Helppokäyttöisyys ja konkreettinen hyöty.	-Virheellisen koodin generointi ja jatkuva tarkastus. -Liiallinen luottamus generoituun koodiin.
Rooli vaatimusmäärittelyssä ja suunnittelussa	-Ideoiden generointi ja suunnitteluprosessien tehostuminen. -Antaa laajemman näkökulman.	-Ihmiskeskisiä tehtäviä, jossa generatiivisen tekoälyn rooli jää pieneksi. -Syötteet voivat johtaa harhaanjohtaviin ratkaisuihin.
Vaikutus ohjelmointiin ja ohjelmistokehitykseen	-Ohjelmoinnin nopeutuminen ja tehostuminen. -Koodin selittäminen, oppimisen edistäminen ja manuaalisen koodaamisen kuormituksen vähentyminen.	-Virheellisen koodin liittäminen koodikantaan saattaa aiheuttaa virheitä jatkossa. -Liiallinen käyttö voi hidastaa projektin etenemistä.
Vaikutus testauksessa ja laadunvarmistuksessa	-Testiskenaarioiden luonnin nopeutuminen. -Auttaa kattavampien testien rakentamisessa ja idoinnissa.	-Kriittisyys testigeneraatioihin. -Hyöty jää pieneksi.
Tietoturva- ja luottamuusriskit	-Yrityksen sisäisten työkalujen ja rajattujen API-avainten käyttö estää tietovuotoriskiä. -Työntekijöiden koulutuksella voidaan ehkäistä riskejä.	-Luottamuksillisten tietojen antaminen avoimille malleille. -Oraakkeliongelma ja generatiivisten mallien "musta laatikko" -luonne tuovat epävarmuutta.

Tekoälyn omaksumiseen vaikuttavat tekijät ja koetut hyödyt tulivat tutkimuksessa esiin. Haastatteluiden perusteella tekoälytyökalut koetaan konkreettisesti hyödylliseksi, ja niiden tuomat hyödyt ovat merkittäviä. Haastateltavien mukaan käyttäjäystävällisyys ja helppokäyttöisyys tukevat työkalujen omaksumista. Vaikka tekoälymallien käytöllä on myös omat rajoitteensa, tutkimuksen perusteella sen tuomat hyödyt ylittävät haitat. Virheellisten generoitujen syötteiden antaminen ja liiallinen luottamus malleihin saattaa kuitenkin kostautua myöhemmin kehityksessä ilmenemissä virheissä ja vaatien kehittäjien jatkuvaa tarkastusta.

Omaksumisen ja hyödyllisyyteen liittyvät vastaukset kulkevat käsi kädessä muiden tutkimuksessa käytyjen teemojen kanssa. Ohjelmistokehityksen elinkaaressa on paljon aikaa vieviä, rutiininomaisia työtehtäviä, joissa tekoälyn käyttö vähentää aikaa manuaalista työtä vaativissa työtehtävissä ja vapauttaa aikaa muihin ajattelua vaativiin tehtäviin. Tekoälyä voi käyttää monissa eri merkeissä työskentelyn tukena. Raghi ja muut (2024) toteavat tutkimuksessaan, että tekoälymallien helppokäyttöisyys ja kyky vähentää käsin tehtävää työtä mahdollistavat sen, että kehittäjät voivat keskittyä korkean tason suunnitteluun. Vaikka tekoälytyökalut vaikuttavat positiivisesti ohjelmistokehityksessä, tutkimuksessa kävi ilmi, että haastavimmissa tilanteissa ohjelmistokehittäjien asiantuntemus on ratkaisevassa roolissa ja tekoälyn hyödyntäminen jää pienemmäksi.

Vaatimusmäärittelyssä ja suunnittelussa tekoälyä hyödynnetään erityisesti ideoinnin, vaihtoehtojen ja teknisten ratkaisujen tukena, mikä antaa ohjelmistokehittäjälle laajemman näkökulman hahmottaa kokonaistilannetta. Hnatushenkon ja Pavlenkon (2024) mukaan tekoäly ei korvaa testaajaa, vaan toimii enemmänkin testauksen tehostajana. Vaikka työkaluilla voidaan nopeuttaa vaatimusmäärittelyä ja suunnitteluprosesseja, tutkimuksen perusteella työtehtävät ovat edelleen ihmiskeskeisiä ja vaativat ihmisen ammattikokemusta ja ajattelua toteutuksissa. Tästä huolimatta tekoälyä hyödynnetään, ja oikeiden kehotteiden antaminen mallille on ensiarvoisen tärkeää.

Ohjelmistokehityksessä tutkimuksen tulokset antoivat lupaavia tuloksia tekoälyn hyödyntämisestä. Haastateltavat kokivat ohjelmointia ja ohjelmistokehitystä tehostavan tekoälyn tuoma etu, varsinkin koodipätkien ja algoritmien generoinnissa. Tällöin automaatio säästää ohjelmoijien aikaa vähentäen rutiiniomaista koodaamista ja vapauttaen aikaa keskittymään korkeamman tason suunnittelua vaativiin tehtäviin. Lima ja muut (2024) korostivat tutkimuksessaan, että tekoäly auttavaa kehittäjiä ajansäästössä automaatiota vaativissa tehtävissä sekä tarjoaa samalla suuren avun koodin korjaamisessa ja rakenteiden muuntamisessa. Lisäksi haastateltavat näkivät, että tekoälyn hyödyntäminen tehostava oppimista tilanteissa, joissa koodin ja sen funktioiden selittäminen on tarpeen.

Haasteita ilmenee hyötyjen lisäksi. Tutkimustulosten perusteella haastateltavat pitävät suurimpina riskeinä virheellisen koodin generoinnin ja mahdolliset ongelmat, jos koodin luotetaan sellaisena, kuin tekoäly tuottaa, ja sitä liitetään ohjelmistoon vähäisellä tarkastuksella. Hosseinin ja muiden (2025) tutkimuksessa todettiin, että tekoäly voi ehdottaa tarpeetonta ja huonosti soveltuvaa koodia tilanteissa, joissa samaan käyttäjän kehotteeseen voidaan ehdottaa hyvin erilaista koodia. Virheellisen koodin tuotanto saattaa hidastaa projektin etenemistä. Kokonaisuudessaan haastateltavat arvioivat, että oikeanlaisella käytöllä ja riittävällä varovaisuudella tekoälyn käyttö ohjelmoinnissa tuo merkittäviä etuja.

Vaikutus testauksessa ja laadun varmistuksessa tulosten perusteella tekoäly tuo etua erityisesti testiskenaarioiden, testipohjien ja yksikkötestien luomisessa. Lisäksi koodin tarkastus mahdollisten bugien kannalta on kätevää, mikä vähentää manuaalisen työn panostusta ja samalla nopeuttaa testausprosesseja. Kuitenkin testituloksia tulee tulkita kriittisesti, sillä liiallinen luotto testituloksiin saattaa johtaa virheellisiin testaus- ja laadunvarmistustuloksiin. Sergeyukin ja muiden (2025) tutkimuksessa havaittiin, että ohjelmistokehittäjien kynnys käyttää tekoälyä testauksessa on suuri, sillä se nähdään epämiellyttävänä. Tästä huolimatta, haastateltavien mukaan tekoälyn hyödyntäminen testauksessa jää pienemmäksi kuin ohjelmoinnissa osittain siksi, että testaukselle on laadittava selkeät reunaehdot ja vaatimukset, jolloin ihmiskeskeinen harkinta on suuressa roolissa.

Tutkimuksen tulosten perusteella tekoälyyn liittyy tietoturva- ja luottamusriskejä. Yleisimpänä huolena haastateltavien keskuudessa ilmeni se, että he eivät halua antaa luottamuksellista koodia kielimalleille. Näihin liittyviä yleisimpiä huolia ovat tekoälyn vuotavan tietoa, mallin koulutukseen käytettävät käyttäjäsyötteet sekä mallien harjoittaminen vapaasti saatavilla tietoaineistolla. Haastateltavien mukaan yritykset ovat varautuneita näihin riskeihin käyttämällä omaa infrastruktuuria ja API-ratkaisuja eri avainten suojaamana, jolloin tietovuodot ovat epätodennäköisiä. Joissakin tapauksissa yritykset käyttävät omia tekoälyn sovelluksia, jotka toimivat sisäisesti ilman ulkopuolista yhteyttä. Nguyen-Ducin ja muiden (2023) tutkimuksen tulosten perustella haastateltavilla on samat näkemykset kuin tässä tutkimuksessa. Lisäksi tietosuojan ja turvallisuuden säännölliset tarkastukset ovat tarpeen mahdollisten heikkouksien tunnistamiseksi ja korjaamiseksi.

Kehittäjien koulutuksella voidaan lisätä tietoisuutta tekoälyn käyttöön liittyvistä haitoista ja puutteista (Sergeyuk ja muut, 2025, s. 10). Haastateltavat tiedostavat mahdolliset riskit, mutta se ei estä mallien käyttöä. He korostivat, että riskit riippuvat merkittävästi siitä, miten mallia käyttää ja mitä sille syöttää. Huolellisella käytöllä ja riskien tiedostamisella voidaan vähentää tietoturva- ja luottamusriskejä, vaikka mikään ympäristö ei ole täysin immuuni tietomurroille.

6.1 Pohdinta

Tutkimuksen tavoitteena oli vastata kahteen määritettyyn tutkimuskysymykseen ja selittää, miten tutkimus tukee laadittuja tutkimuskysymyksiä. Kirjallisuuden ja aiempien tutkimustulosten pohjalta löydettiin paljon yhteneväisiä tuloksia.

TK 1: Kuinka hyödyllistä on hyödyntää generatiivisen tekoälyn työkaluja ohjelmistokehityksessä?

Ensimmäiseen tutkimuskysymykseen vastaten tekoälytyökalut koetaan erittäin hyödyllisiksi ohjelmistokehityksessä. Ne tukevat ohjelmistokehittäjien työtehtäviä etenkin rutiiniomaisten työtehtävien parissa, vapauttaen ajattelun ja ajankäytön muihin tehtäville. Työkalut toimivat ohjelmistokehittäjien apuna ongelmien ratkomisessa eri näkökulmista ja tarjoamaan vaihtoehtoisia lähestymistapoja. Suurin konkreettinen hyöty ilmenee ohjelmoinnissa ja datan käsittelyä vaativissa tehtävissä.

Samalla tekoäly ei välttämättä ole hyödyllinen, vaan sen käyttö vaatii käyttäjältä kriittistä asennetta. Liiallinen luottamus saattaa johtaa virheisiin ja riippuvaiseksi mallin tuottamiin syötteisiin. Jos virheet jää huomaamatta, sillä saattaa olla suuret seuraukset ohjelmistokehityksen myöhemmissä elinkaaren vaiheissa ja tekoälyn tuoma hyöty tehokkuuden lisääntymisessä poistuu. Tästä huolimatta tutkimukset ja haastatteluiden vastauksista saadut havainnot osoittavat, että tekoälyn hyödyntäminen nopeuttaa kehitysprosessia ja antavat kehittäjille mahdollisuuden parantaa tehokkuutta.

TK 2: Miten generatiivisen tekoälyn käyttöönotto muuttaa perinteisiä ohjelmistokehitysprosesseja ja mitä vaatimuksia se asettaa ohjelmistokehittäjälle?

Toiseen tutkimuskysymykseen vastaten tekoälyn käyttöönotto muuttaa perinteisiä ohjelmistokehitysprosesseja siirtämällä painopistettä manuaalisesta ohjelmoinnista ja yksityiskohtien tarkastelusta laajemmalle skaalalle, kuten suunnitteluun, konseptien toteuttamiseen ja laadunvarmistukseen. Ohjelmistokehittäjiltä edellytetään pysymään jatkuvasti nopeasti kehittyvien teknologioiden mukana ja kehittääkseen uusia taitoja. Tekoälyn käyttöönotto edellyttää uudenlaista lähestymistapaa ohjelmistokehityksen elinkaarelle, jossa käyttäjät ovat vuorovaikutuksessa työkalujen kanssa jokaisessa vaiheessa. Tämä muuttaa työkuvaa siten, että kehittäjien työ keskittyy yhä enemmän ongelmien hahmottamiseen, kun tekoäly automatisoi rutiininomaiset ja toistuvat tehtävät.

6.2 Rajoitukset

Tällä tutkimuksella on rajoitteensa, jotka ovat vaikuttaneet tutkimustuloksiin ja niiden tulkintaan. Ensimmäisen rajoite liittyy haastateltavien työkuvaan ja tekoälyn kykyyn tuottaa sisältöä. Tutkimuksessa käsitellyt teemat ulottuvat koko ohjelmistokehityksen elinkaareen, joista monilla on kokemusta, mutta tekoälyn syötteiden luovuuden puute tuottaa osassa käsitellyissä teemoissa rajoitteisia tuloksia. Tämän lisäksi haastateltavista löytyi ohjelmistokehittäjien lisäksi datainsinöörejä, jotka hyödyntävät tekoälytyökalusovelluksia eri tehtävissä kuin ohjelmistokehittäjät. Tämä tekee haastateltavien vastausten vertailusta hankalampaa, mutta tutkimuksen laajuuden näkökulmasta monipuolisempaa.

Toinen rajoite liittyy haastateltavien kokemukseen ohjelmistokehityksen parissa. Tutkimustuloksia tekoälyn hyödyntämisestä olisi voinut täydentää haastateltavien vastaukset, joilla on vähemmän kokemusta ja sellaiset, joilla on jo vuosikymmeniä kokemusta ohjelmistokehityksestä ilman tekoälymallien avustuksia. Tämä olisi auttanut ymmärtämään, kuinka suurena apuna tekoäly koetaan eri kokemusasteilla. Kuitenkin haastateltavien valintakriteerit oli määritelty siten, että heillä oli työkokemusta ajalta, jolloin tekoäly oli jo saatavilla ja tutkimus on toteutettu sen mukaisesti. Teoreettisesta viitekehyksestä ja tutkimusosiosta esiin nousevat tulokset eri tutkimuksista antavat kuitenkin jo alustavan käsityksen eritasoisten ohjelmistokehittäjien kokemusten vaikutuksesta.

Valittu kirjallisuus tutkimukseen perustuu tutkijan valitsemiin lähteisiin ja artikkeleihin, jotka on valittu niiden sisällön yhteneväisyyden perusteella suhteessa käsiteltäviin teemoihin. Aiheesta on julkaistu paljon tutkimuksia viimeisen kahden vuoden aikana, mikä tekee valittavien artikkeleiden valinnasta haastavaa. Tämä voi osaltaan aiheuttaa vinoumaa tutkimukseen ja vaikuttaa tulosten yleistettävyyteen, sillä lähteiden valinta heijastaa tutkijan tekemää rajausta ja painotuksia.

6.3 Jatkotutkimusaiheet

Tulevaisuutta silmällä pitäen jatkotutkimusaiheet voisivat keskittyä enemmän tekoälyn ja ohjelmistokehityksen prosessien yhä saumattomampaan yhdistämiseen. Tekoälyn hyödyntäminen ohjelmistokehityksessä on suhteellisen tuore ilmiö, ja tekoälymallit kehittynyt nopeaa vauhtia. Vasta viimeisen kahden vuoden aikana on ilmennyt tutkimuksia aiheesta, mikä vaikuttaa siihen, että tekoälyavusteisen ohjelmistokehityksen luomisessa on yhä monia erilaisia ongelmia, jotka vaativat pitkäjänteistä tutkimusta. Erityisen tärkeäksi nähtäisiin kohdistaa huomio paitsi järjestelmien kehitykseen, mutta puuttuen tyypillisiin haasteisiin, kuten mallien tuottamien syötteiden laadun puutteeseen, laajemman kontekstin ymmärtämiseen sekä turvallisuuteen ja juridisten liittyvien vaatimusten noudattamiseen. Jatkotutkimusten tulisi sovittaa ohjelmistokehittäjien tarpeisiin ja hahmottaa syitä, jotka johtavat tiettyihin haasteisiin tekoälyn käytön toiminnan kannalta ohjelmistokehityksen prosesseissa.

Lähteet

Aarti. (2024). Generative AI in software development: An overview and evaluation of modern coding tools. *International Journal for Multidisciplinary Research (IJFMR), 6*(3). https://doi.org/10.36948/ijfmr.2024.v06i03.23271

Alasuutari, P., & Vastapaino. (2011). *Laadullinen tutkimus 2.0* (4. uud. p.). Osuuskunta Vastapaino. Noudettu 20.3.2025 osoitteesta https://tritonia.finna.fi/Record/tria.296164?sid=4990851857

Alenezi, M., & Akour, M. (2025). AI-Driven Innovations in Software Engineering: A Review of Current Practices and Future Directions. *Applied Sciences*, *15*(3), 1344. https://doi.org/10.3390/app15031344

Aleti, A. (2023). *Software testing of generative AI systems: Challenges and opportunities.* In *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)* (pp. 4-14). Melbourne, Australia: IEEE. https://doi.org/10.1109/ICSE-FoSE59343.2023.00009

Almarie, B., Teixeira, P. E. P., Pacheco-Barrios, K., Rossetti, C. A., & Fregni, F. (2023). Editorial – The use of large language models in science: Opportunities and challenges. *Princ Pract Clin Res, 9*(1), 1–4. https://doi.org/10.21801/ppcrj.2023.91.1

Al-Msie'deen, et al. (2021). Constructing a software requirements specification and design for electronic IT news magazine system. *International Journal of Advanced and Applied Sciences,* *8*(11), 104–118. https://doi.org/10.21833/ijaas.2021.11.014

Alt, T., Ibisch, A., Meiser, C., Wilhelm, A., Zimmer, R., Ditz, J., Dresen, D., Droste, C., Karschau, J., Laus, F., Müller, O., Neu, M., Plaga, R., Plesch, C., Sennewald, B., Thaeren, T., Unverricht, K., & Waurick, S. (2025). Generative AI models: Opportunities and risks for industry and authorities. arXiv. https://doi.org/10.48550/arXiv.2406.04734

Arora, C., Grundy, J., & Abdelrazek, M. (2023). Advancing requirements engineering through generative AI: Assessing the role of LLMs. *arXiv*. https://doi.org/10.48550/arXiv.2310.13976

Bandi, A., Adapa, P. V. S. R., & Kuchi, Y. E. V. P. K. (2023). The Power of Generative AI: A Review of Requirements, Models, Input–Output Formats, Evaluation Metrics, and Challenges. *Future Internet*, *15*(8), 260. https://doi.org/10.3390/fi15080260

Barroga, E., & Matanguihan, G. J. (2022). A practical guide to writing quantitative and qualitative research questions and hypotheses in scholarly articles. *Journal of Korean Medical Science, 37*(16), e121. https://doi.org/10.3346/jkms.2022.37.e121

Belagatti, P. (2023). *Prompt engineering: A beginners guide!* Noudettu 20.3.2025 osoitteesta https://levelup.gitconnected.com/prompt-engineering-a-beginners-guide-53ecd38eadf3

Berrada, I. R., Barramou, F. Z., & Alami, O. B. (2022). A review of Artificial Intelligence approach for credit risk assessment. In 2022 2nd International Conference on Artificial Intelligence and Signal Processing (AISP) (pp. 1-5). Vijayawada, India: IEEE. https://doi.org/10.1109/AISP53593.2022.9760655

Çaparlar, C. Ö., & Dönmez, A. (2016). What is scientific research and how can it be done? *Turkish Journal of Anaesthesiology and Reanimation, 44*(4), 212–218. https://doi.org/10.5152/TJAR.2016.34711

Chakraborty, C., Bhattacharya, M., Pal, S., Chatterjee, S., Das, A., & Lee, S.-S. (2025). AI-enabled language models (LMs) to large language models (LLMs) and multimodal large language models (MLLMs) in drug discovery and development. *Journal of Advanced Research*. https://doi.org/10.1016/j.jare.2025.02.011

Chakraborty, U., et al. (2023). *Rise of generative AI and ChatGPT: Understand how generative AI and ChatGPT are transforming and reshaping the business world (English edition).* BPB Publications. ProQuest Ebook Central. Noudettu 25.2.2025 osoitteesta https://ebookcentral-proquest-com.proxy.uwasa.fi/lib/tritonia-ebooks/detail.action?docID=30495009

Cheng, H., Husen, J. H., Lu, Y., Racharak, T., Yoshioka, N., Ubayashi, N., & Washizaki, H. (2025). Generative AI for requirements engineering: A systematic literature review. *arXiv*. https://doi.org/10.48550/arXiv.2409.06741

Chopra, K. (2024). *A brief history of generative AI*. Medium. Noudettu 26.3.2025 osoitteesta https://medium.com/@kamalmeet/a-brief-history-of-generative-ai-e91a95d5604c

Cotroneo, D., Improta, C., Liguori, P., & Natella, R. (2024). Vulnerabilities in AI code generators: Exploring targeted data poisoning attacks. *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension (ICPC '24)*, 280–292. Association for Computing Machinery. https://doi.org/10.1145/3643916.3644416

Das, S., & Das, D. (2024). *Natural language processing (NLP) techniques: Usability in human-computer interactions.* In *2024 6th International Conference on Natural Language Processing (ICNLP)* (pp. 783–787). Xi'an, China. IEEE. https://doi.org/10.1109/ICNLP60986.2024.10692776

Dias, F. S., & Lauretta, G. A. (2024). The transformative impact of AI and deep learning in business: A literature review. *arXiv preprint arXiv:2410.23443*. https://doi.org/10.48550/arXiv.2410.23443

Dongare, A. D., Kharde, R. R., & Kachare, A. D. (2012). *Introduction to artificial neural network*. Noudettu 29.3.2025 osoitteesta https://api.semanticscholar.org/CorpusID:212457035

Donvir, A., Panyam, S., Paliwal, G., & Gujar, P. (2024). The role of generative AI tools in application development: A comprehensive review of current technologies and practices. *2024 International Conference on Engineering Management of Communication and Technology (EMCTECH)*, 1-9. https://doi.org/10.1109/EMCTECH63049.2024.10741797

Ebert, C., & Louridas, P. (2023). Generative AI for software practitioners. *IEEE Software, 40*(4), 30–38. https://doi.org/10.1109/MS.2023.3265877

Ebert, C., Arockiasamy, J. P., Hettich, L., & Weyrich, M. (2024). Hints for generative AI software development. *IEEE Software, 41*(5), 24–33. https://doi.org/10.1109/MS.2024.3410641

Eiras, F., Petrov, A., Vidgen, B., Schroeder, C., Pizzati, F., Elkins, K., Mukhopadhyay, S., Bibi, A., Purewal, A., Botos, C., Steibel, F., Keshtkar, F., Barez, F., Smith, G., Guadagni, G., Chun, J., Cabot, J., Imperial, J., Nolazco, J. A., Landay, L., Jackson, M., Torr, P. H. S., Darrell, T., Lee, Y., & Foerster, J. (2024). Risks and opportunities of open-source generative AI. arXiv. https://doi.org/10.48550/arXiv.2405.08597

Eskola, J., & Suoranta, J. (1998). *Johdatus laadulliseen tutkimukseen*. Vastapaino. Noudettu 15.3.2025 osoitteesta https://tritonia.finna.fi/Record/tria.378715?sid=4986538708

Fan, A., et al. (2023). Large language models for software engineering: Survey and open problems. *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*, 31-53. https://doi.org/10.1109/ICSE-FoSE59343.2023.00008

Fazelnia, M., Mirakhorli, M., & Bagheri, H. (2024). *Translation titans, reasoning challenges: Satisfiability-aided language models for detecting conflicting requirements.* https://doi.org/10.1145/3691620.3695302

Ferrara, E. (2024). GenAI against humanity: Nefarious applications of generative artificial intelligence and large language models. *Journal of Computational Social Science, 7*, 549–569. https://doi.org/10.1007/s42001-024-00250-1

Feuerriegel, S., Hartmann, J., Janiesch, C., et al. (2024). Generative AI. *Business & Information Systems Engineering, 66*(2), 111–126. https://doi.org/10.1007/s12599-023-00834-7

Fischer, M., & Lanquillon, C. (2024). Evaluation of generative AI-assisted software design and engineering: A user-centered approach. In *Artificial Intelligence in HCI: 5th International Conference, AI-HCI 2024, Held as Part of the 26th HCI International Conference, HCII 2024, Washington, DC, USA, June 29 – July 4, 2024, Proceedings, Part I* (pp. 31–47). Springer-Verlag. https://doi.org/10.1007/978-3-031-60606-9_3

France, S. L. (2024). *Navigating software development in the ChatGPT and GitHub Copilot era. Business Horizons, 67(5)*, 649–661. https://doi.org/10.1016/j.bushor.2024.05.009

Fu, B., Hadid, A., & Damer, N. (2025). Generative AI in the context of assistive technologies: Trends, limitations and future directions. *Image and Vision Computing, 154*, 105347. https://doi.org/10.1016/j.imavis.2024.105347

Fui-Hoon Nah, F., Zheng, R., Cai, J., Siau, K., & Chen, L. (2023). Generative AI and ChatGPT: Applications, challenges, and AI-human collaboration. *Journal of Information Technology Case and Application Research, 25*(3), 277–304. https://doi.org/10.1080/15228053.2023.2233814

Gambacorta, L., Qiu, H., Shan, S., & Rees, D. M. (2024). *Generative AI and labour productivity: A field experiment on coding* (BIS Working Papers No. 1208). Bank for International Settlements. Noudettu 27.3.2025 osoitteesta https://ideas.repec.org/p/bis/biswps/1208.html

Gamieldien, Y., Case, J. M., & Katz, A. (2023). Advancing qualitative analysis: An exploration of the potential of generative AI and NLP in thematic coding. *SSRN*. https://doi.org/10.2139/ssrn.4487768

Garg, A., & Sharma, D. (2023). *Generative AI for software test modelling with a focus on ERP software.* In *2023 International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)* (pp. 187-193). Faridabad, India: IEEE. https://doi.org/10.1109/ICAICCIT60255.2023.10466102

Ghai, A. S., Rawat, V., Gupta, V. K., & Ghai, K. (2024). Artificial intelligence in system and software engineering for auto code generation. *2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT)*, 1–5. https://doi.org/10.1109/ICEECT61758.2024.10738945

Gonzalez-Barahona, J. M. (2024). *Software development in the age of LLMs and XR.* In *2024 IEEE/ACM First IDE Workshop (IDE)* (pp. 66–69). Lisbon, Portugal. https://doi.org/10.1145/3643796.3648457

Gozalo-Brizuela, R., & Merchan, E. E. G. (2024). A survey of generative AI applications. *Journal of Computer Science, 20*(8), 801–818. https://doi.org/10.3844/jcssp.2024.801.818

Hamilton, A. B., & Finley, E. P. (2019). Qualitative methods in implementation research: An introduction. *Psychiatry Research, 280*, 112516. https://doi.org/10.1016/j.psychres.2019.112516

Hammersley, M. (2012). *What is qualitative research?* Bloomsbury Academic. https://doi.org/10.5040/9781849666084

Harika, J., Baleeshwar, P., Navya, K., & Shanmugasundaram, H. (2022). A review on artificial intelligence with deep human reasoning. *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, 81–84. https://doi.org/10.1109/ICAAIC53929.2022.9793310

Harkut, D. G. (Ed.). (2019). *Artificial Intelligence – Scope and Limitations*. IntechOpen. https://doi.org/10.5772/intechopen.77611

Hnatushenko, V., & Pavlenko, I. V. (2024). The use of generative artificial intelligence in software testing. *System Technologies, 2*, 113-123. https://doi.org/10.34185/1562-9945-2-151-2024-10

Hossain, D. (2011). *Qualitative research process*. Postmodern Openings, 7. Noudettu 5.4.2025 osoitteesta https://www.researchgate.net/publication/267798343_Qualitative_Research_Process

Hosseini, M., Horbach, S. P. J. M., Holmes, K., & Ross-Hellauer, T. (2025). Open science at the generative AI turn: An exploratory analysis of challenges and opportunities. *Quantitative Science Studies, 6*, 22–45. https://doi.org/10.1162/qss_a_00337

Humphreys, D., Koay, A., Desmond, D., et al. (2024). AI hype as a cyber security risk: The moral responsibility of implementing generative AI in business. AI Ethics, 4, 791–804. https://doi.org/10.1007/s43681-024-00443-4

Huo, H., Fu, Z., Yu, H., Yang, S., & Tang, G. (2024). *Chinese text simplification based on large language models.* In *2024 International Conference on Computational Linguistics and Natural Language Processing (CLNLP)* (pp. 52–56). Yinchuan, China: IEEE. https://doi.org/10.1109/CLNLP64123.2024.00018

Jackson, V., Vasilescu, B., Russo, D., Ralph, P., Izadi, M., Prikladnicki, R., D'Angelo, S., Inman, S., Lisboa, A., & van der Hoek, A. (2024). Creativity, generative AI, and software development: A research agenda. *arXiv*. https://doi.org/10.48550/arXiv.2406.01966

Jowsey, T., Deng, C., & Weller, J. (2021). General-purpose thematic analysis: A useful qualitative method for anaesthesia research. BJA Education, 21(12), 472–478. https://doi.org/10.1016/j.bjae.2021.07.006

Kachynskyi, A. B., & Tsebrinska, N. A. (2020). Reinforced machine learning methods for testing quality of cyber threat prediction results. Theoretical and Applied Cybersecurity. https://doi.org/10.20535/tacs.2664-29132020.1.209432

Kanont, K., Pimgmuang, P., Simasathien, T., Wisnuwong, S., Wiwatsiripong, B., Poonpirome, K., Songkram, N., & Khlaisang, J. (2024). *Generative-AI, a learning assistant? Factors influencing higher-ed students' technology acceptance. The Electronic Journal of e-Learning, 22*(6). https://doi.org/10.34190/ejel.22.6.3196

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). *Scaling laws for neural language models*. *arXiv Preprint*, arXiv:2001.08361. https://doi.org/10.48550/arXiv.2001.08361

Khaliq, Z., Farooq, S. U., & Khan, D. A. (2022). *Artificial intelligence in software testing: Impact, problems, challenges, and prospect. arXiv preprint*, arXiv:2201.05371. https://doi.org/10.48550/arXiv.2201.05371

Kingma, D. P., Rezende, D. J., Mohamed, S., & Welling, M. (2014). Semi-supervised learning with deep generative models. *arXiv preprint arXiv:1406.5298.* https://doi.org/10.48550/arXiv.1406.5298

Li, X., Ma, Z., Tu, Y., & Du, Y. (2021). Study on the application of artificial intelligence technology in empowering education: Taking "Intelligent Learning Partner" as an

example. *2021 2nd International Conference on Information Science and Education (ICISE-IE)*, 1449-1453. Chongqing, China. IEEE. https://doi.org/10.1109/ICISE-IE53922.2021.00323

Lim, W. M. (2024). What is qualitative research? An overview and guidelines. *Australasian Marketing Journal, 0*(0). https://doi.org/10.1177/14413582241264619

Lima, C. D. C., et al. (2024). Generative AI impact on the future of work: Insights from software development. *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 3887-3892. https://doi.org/10.1109/SMC54092.2024.10831787

Ma, X. (2023). Application of artificial intelligence in computer network technology. *2023 2nd International Conference on Artificial Intelligence and Autonomous Robot Systems (AIARS)*, 182–186. https://doi.org/10.1109/AIARS59518.2023.00043

Maguire, M., & Delahunt, B. (2017). Doing a thematic analysis: A practical, step-by-step guide for learning and teaching scholars. AISHE-J: The All Ireland Journal of Teaching and Learning in Higher Education, 9(3), 3351–3364. https://doi.org/10.62707/aishej.v9i3.335

Mary, A., & Hoover, R. (2024). Revolutionizing software development with generative AI: Automated code creation, testing, and integration. *Automation, 10*. Noudettu 16.3.2025 osoitteesta https://www.researchgate.net/publication/387682552_Revolutionizing_Software_Development_with_Generative_AI_Automated_Code_Creation_Testing_and_Integration

Mo, Y., Qin, H., Dong, Y., Zhu, Z., & Li, Z. (2024). Large language model (LLM) AI text generation detection based on transformer deep learning algorithm. *arXiv preprint arXiv:2405.06652*. https://doi.org/10.48550/arXiv.2405.06652

Modi, M. D. B. (2024). *Transforming software development through generative AI: A systematic analysis of automated development practices*. Qualcomm, USA. https://doi.org/10.32628/CSEIT24106197

Nasution, A. H., & Onan, A. (2024). *ChatGPT Label: Comparing the quality of human-generated and LLM-generated annotations in low-resource language NLP tasks*. *IEEE Access, 12*, 71876–71900. https://doi.org/10.1109/ACCESS.2024.3402809

Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., & Mian, A. (2024). A comprehensive overview of large language models [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.2307.06435

Nazari, A. R., & Thunell, B. N. (2024). *Usage of generative AI-based plugin in unit testing: Evaluating the trustworthiness of generated test cases by Codiumate, an IDE plugin powered by GPT-3.5 & 4* (Dissertation). Noudettu 16.3.2025 osoitteesta https://urn.kb.se/resolve?urn=urn:nbn:se:bth-26473

Negri-Ribalta, C., Geraud-Stewart, R., Sergeeva, A., & Lenzini, G. (2024). A systematic literature review on the impact of AI models on the security of code generation. *Frontiers in Big Data, 7*. https://doi.org/10.3389/fdata.2024.1386720

Nguyen-Duc, A., Cabrero-Daniel, B., Przybylek, A., Arora, C., Khanna, D., Herda, T., Rafiq, U., Melegati, J., Guerra, E., Kemell, K.-K., Saari, M., Zhang, Z., Le, H., Quan, T., & Abrahamsson, P. (2023). Generative artificial intelligence for software engineering – A research agenda. *arXiv preprint arXiv:2310.18648*. https://doi.org/10.48550/arXiv.2310.18648

Onwuegbuzie, A. J., & Leech, N. L. (2006). Linking Research Questions to Mixed Methods Data Analysis Procedures 1. The Qualitative Report, 11(3), 474-498. https://doi.org/10.46743/2160-3715/2006.1663

Paavilainen, M., Ylihurulaa, A., Niinimäki, V., Kumar, A., Loven, L., Pirttikangas, S., & Tarkoma, S. (2025). Software Development for Scientific Computing using AI Code Generation. Noudettu 25.2.2025 osoitteesta https://researchportal.helsinki.fi/en/publications/software-development-for-scientific-computing-using-ai-code-gener-2

Pangavhane, S., Raktate, G., Pariane, P., Shelar, K., Wakchaure, R., & Kale, J. N. (2024). AI-augmented software development: Boosting efficiency and quality. *2024 International Conference on Decision Aid Sciences and Applications (DASA)*, 1–5. https://doi.org/10.1109/DASA63652.2024.10836523

Patil, K. P., & Pramod, D. (2024). Conversational artificial intelligence in the workplace: Analysing the impact of ChatGPT on users' perceived self-efficacy. *2024 2nd International Conference on Advancement in Computation & Computer Technologies (InCACCT)*, 766–770. https://doi.org/10.1109/InCACCT61598.2024.10551239

Peng, et al., (2023). "The Impact of AI on Developer Productivity: Evidence from GitHub Copilot." arXiv.org, https://doi.org/10.48550/arxiv.2302.06590

Petrovska, O., Clift, L., Moller, F., & Pearsall, R. (2024). Incorporating generative AI into software development education. *Proceedings of the 8th Conference on Computing Education Practice (CEP '24)*, 37–40. Association for Computing Machinery. https://doi.org/10.1145/3633053.3633057

Pochu, S., Kathram, S. R., & Engineer, S. D. (2022). Synergizing automation and human insight: A comprehensive approach to software testing for quality assurance. *Journal of Multidisciplinary Research, 8*(1), 51-62. Noudettu 20.2.2025 osoitteesta https://www.researchgate.net/publication/388497716_Synergizing_Automation_and_Human_Insight_A_Comprehensive_Approach_to_Software_Testing_for_Quality_Assurance

Preis, A. (2022). *The most important categories of machine learning*. Noudettu 16.3.2025 osoitteesta https://www.itwm.fraunhofer.de/en/departments/fm/latest-news/blog/categories_machine_learning.html

Puusa, A., Juuti, P., & Aaltio, I. (2020). *Laadullisen tutkimuksen näkökulmat ja menetelmät*. Gaudeamus. Noudettu 1.3.2025 osoitteesta https://tritonia.finna.fi/Record/tria.378779?sid=4947420041

Raghi, K. R., Sudha, K., A. M, S., & Joshua, S. S. (2024). *Software development automation using generative AI.* In *2024 International Conference on Emerging Research in*

Computational Science (ICERCS) (pp. 1–6). IEEE. https://doi.org/10.1109/ICERCS63125.2024.10894980

Rane, N. L., Mallick, S. K., Kaya, O., & Rane, J. (2024). Role of machine learning and deep learning in advancing generative artificial intelligence such as ChatGPT. In *Applied machine learning and deep learning: Architectures and techniques* (pp. 96-111). Deep Science Publishing. https://doi.org/10.70593/978-81-981271-4-3_5

Ren, Z. (2022). *The advance of generative model and variational autoencoder*. In *2022 IEEE Conference on Telecommunications, Optics and Computer Science (TOCS)* (pp. 268–271). Dalian, China: IEEE. https://doi.org/10.1109/TOCS56154.2022.10016057

Rincy. N., & Gupta, R. (2020). A survey on machine learning approaches and its techniques. *2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, 1-6. Bhopal, India. IEEE. https://doi.org/10.1109/SCEECS48394.2020.190

Saaranen-Kauppinen, A. & Puusniekka, A. (2006). Aineiston määrä ja tutkittavat. KvaliMOTV - Menetelmäopetuksen tietovaranto [verkkojulkaisu]. Tampere: Yhteiskuntatieteellinen tietoarkisto. Noudettu 1.3.2025 osoitteesta https://www.fsd.tuni.fi/menetelmaopetus/kvali/L6_2.html

Sajja, A., Thakur, D., & Mehra, A. (2024). Integrating generative AI into the software development lifecycle: Impacts on code quality and maintenance. *International Journal of Science and Research Archive, 13*, 1952-1960. https://doi.org/10.30574/ijsra.2024.13.1.1837

Saleem, S., Asim, M. N., Van Elst, L., & Dengel, A. (2024). Generative language models potential for requirement engineering applications: Insights into current strengths and limitations. *arXiv*. https://doi.org/10.48550/arXiv.2412.00959

Sauvola, J., Tarkoma, S., Klemettinen, M., et al. (2024). Future of software development with generative AI. *Automated Software Engineering, 31*(26). https://doi.org/10.1007/s10515-024-00426-z

Sengar, S. S., Hasan, A. B., Kumar, S., & others. (2024). *Generative artificial intelligence: A systematic review and applications*. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-024-20016-1

Sergeyuk, A., Golubev, Y., Bryksin, T., & Ahmed, I. (2025). Using AI-based coding assistants in practice: State of affairs, perceptions, and ways forward. *Information and Software Technology, 178*, 107610. https://doi.org/10.1016/j.infsof.2024.107610

Sikand, S., Mehra, R., Sharma, V. S., Kaulgud, V., Podder, S., & Burden, A. P. (2024). Do generative AI tools ensure green code? An investigative study. *2024 IEEE/ACM International Workshop on Responsible AI Engineering (RAIE)*, 52-55. https://doi.org/10.1145/3643691.3648588

Simaremare, M., & Edison, H. (2024). The state of generative AI adoption from software practitioners' perspective: An empirical study. *2024 50th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 106–113. https://doi.org/10.1109/SEAA64295.2024.00024

Şimşek, T., Gülşeni, Ç., & Olcay, G. A. (2024). The future of software development with GenAI: Evolving roles of software personas. *IEEE Engineering Management Review*. https://doi.org/10.1109/EMR.2024.3454112

Singh, P. D. (2024). *Generative AI through the lens of Technology Acceptance Model.* SSRN. https://doi.org/10.2139/ssrn.4953174

Sun, Z., Yuan, Y., Dong, X., Liu, Z., Cai, K., Cheng, W., Wu, J., Qiao, Z., & Chen, A. (2023). Supervised machine learning: A new method to predict the outcomes following exercise intervention in children with autism spectrum disorder. International Journal of Clinical and Health https://doi.org/10.1016/j.ijchp.2023.100409

Syed, I., & Lokhande, V. (2024). An overview of the supervised machine learning. *International Research Journal of Modernization in Engineering Technology and Science, 6*(3). https://doi.org/10.56726/IRJMETS51366

Teles, G., Rodrigues, J. J. P. C., Rabêlo, R. A. L., & Kozlov, S. A. (2021). Comparative study of support vector machines and random forests machine learning algorithms on credit operation. https://doi.org/10.1002/spe.2842

Tenny, S., Brannan, J. M., & Brannan, G. D. (2022). Qualitative study. In *StatPearls* [Internet]. StatPearls Publishing. Noudettu 25.3.2025 osoitteesta https://www.ncbi.nlm.nih.gov/books/NBK470395/

Thakur, D., Mehra, A., Choudhary, R., & Sarker, M. (2024). Generative AI in software engineering: Revolutionizing test case generation and validation techniques. *Iconic Research and Engineering Journals, 7*(5), 281-293. IRE Journals. Noudettu 16.3.2025 osoitteesta https://www.irejournals.com/paper-details/1705175

Tihanyi, N., Bisztray, T., Jain, R., Ferrag, M. A., Cordeiro, L. C., & Mavroeidis, V. (2023). The FormAI dataset: Generative AI in software security through the lens of formal verification. *Proceedings of the 19th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE 2023)*, 33–43. Association for Computing Machinery. https://doi.org/10.1145/3617555.3617874

Torka, S., & Albayrak, S. (2024). Optimizing AI-assisted code generation. *arXiv*. https://doi.org/10.48550/arXiv.2412.10953

Treude, C., & Storey, M.-A. (2025). *Generative AI and empirical software engineering: A paradigm shift*. arXiv. https://doi.org/10.48550/arXiv.2502.08108

Tuomi, J., Sarajärvi, A., & Tammi. (2018). *Laadullinen tutkimus ja sisällönanalyysi* (Uudistettu laitos.). Kustannusosakeyhtiö Tammi. Noudettu 20.3.2025 osoitteesta https://tritonia.finna.fi/Record/tria.360759?sid=4986555004

Tuominen, H., & Neittaanmäki, P. (2019). *Tekoälyn perusteita ja sovelluksia*. Noudettu 18.3.2025 osoitteesta https://tim.jyu.fi/view/kurssit/tie/tiep1000/tekoalyn-sovellukset/kirja#DKUvbnUuGytQ

Ulfsnes, R., Moe, N. B., Stray, V., & Skarpen, M. (2024). Transforming software development with generative AI: Empirical insights on collaboration and workflow. *arXiv*. https://doi.org/10.48550/arXiv.2405.01543

Velaga, S. P. (2020). *AI-assisted code generation and optimization: Leveraging machine learning to enhance software development processes*. *International Journal of Innovative Engineering and Research Technology (IJJERT), 7*(09), 177–186. https://doi.org/10.26662/ijiert.v7i09.pp177-186

Walker, S. (2025). *What does "artificial intelligence" mean?* Noudettu 15.3.2025 osoitteesta https://saifr.ai/blog/what-does-artificial-intelligence-mean

Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). *Software testing with large language models: Survey, landscape, and vision. IEEE Transactions on Software Engineering, 50*(4), 911-936. https://doi.org/10.1109/TSE.2024.3368208

Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering, 50*(4), 911-936. https://doi.org/10.1109/TSE.2024.3368208

Wang, Y., Zhang, Y., Lu, Y., & Yu, X. (2020). A Comparative Assessment of Credit Risk Model Based on Machine Learning — A case study of bank loan data. Procedia Computer Science, 174, 141-149. https://doi.org/10.1016/j.procs.2020.06.069

Warudkar, S., & Jalit, R. (2024). Unlocking the potential of generative AI in large language models. *2024 Parul International Conference on Engineering and Technology (PICET)*, 1–5. https://doi.org/10.1109/PICET60765.2024.10716156

Wei, B. (2024). Requirements are all you need: From requirements to code with LLMs. *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, Reykjavik, Iceland, 416–422. https://doi.org/10.1109/RE59067.2024.00049

Weisz, J. D., He, J., Muller, M., Hoefer, G., Miles, R., & Geyer, W. (2024). Design principles for generative AI applications. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Article 378, pp. 1–22). Association for Computing Machinery. https://doi.org/10.1145/3613904.3642466

Wright, L. G., Onodera, T., Stein, M. M., et al. (2022). Deep physical neural networks trained with backpropagation. *Nature, 601*(549–555). https://doi.org/10.1038/s41586-021-04223-6

Yaacov, T., Elyasaf, A., & Weiss, G. (2024). Boosting LLM-based software generation by aligning code with requirements. *2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW)*, Reykjavik, Iceland, 301–305. https://doi.org/10.1109/REW61692.2024.00045

Yabaku, M. (2023). *Generative AI tools for software engineering: An analysis of functionality and users' expectations* (Dissertation). Noudettu 25.3.2025 osoitteesta https://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-543009

Yahya, N., & Maidin, S. (2022). The waterfall model with agile Scrum as the hybrid agile model for the software engineering team. *Proceedings of the 2022 International Conference on Information Technology Systems and Management (CITSM)*, 1–5. https://doi.org/10.1109/CITSM56380.2022.9936036

Yuan, Z., Liu, M., Ding, S., Wang, K., Chen, Y., Peng, X., & Lou, Y. (2024). *Evaluating and improving ChatGPT for unit test generation. Proceedings of the ACM on Software Engineering, 1*(FSE), Article 76, 24 pages. Association for Computing Machinery. https://doi.org/10.1145/3660783

Zhang, D., Mishra, S., Brynjolfsson, E., Etchemendy, J., Ganguli, D., Grosz, B., Lyons, T., Manyika, J., Niebles, J., Sellitto, M., Shoham, Y., Clark, J., & Perrault, R. (2021). *The AI Index 2021 annual report.* arXiv. https://doi.org/10.48550/arXiv.2103.06312

Zhang, M., & Zhang, Y. (2023). *The study on NLP-based semantic analysis technology to improve the accuracy of English translation.* In *2023 IEEE International Conference on Control, Electronics and Computer Technology (ICCECT)* (pp. 1107–1112). Jilin, China: IEEE. https://doi.org/10.1109/ICCECT57938.2023.10140377

Zubiaga, A. (2024). *Natural language processing in the era of large language models*. *Frontiers in Artificial Intelligence, 6*. https://doi.org/10.3389/frai.2023.1350306

Liitteet

Liite 1: Haastattelukysymykset

1. Työtehtävä
    a. Mikä on nykyinen työtehtäväsi?
2. Työvuosia ohjelmistokehittäjänä
    a. Kuinka monta vuotta olet työskennellyt ohjelmistokehityksen parissa?
3. Omaksumiseen vaikuttavat tekijät ja koettu hyödyllisyys
    a. Miten olet hyödyntänyt ja ottanut käyttöön generatiivisia tekoälytyökaluja ja millaisia haasteita tai onnistumisia olet havainnut matkalla?
    b. Missä määrin uskot, että generatiivisen tekoäly käyttö tukee sinua ohjelmistokehittäjänä?
4. Rooli vaatimusmäärittelyssä ja suunnittelussa
    a. Miten generatiivinen tekoäly on vaikuttanut vaatimusmäärittelyn ja suunnittelun prosesseihin, esimerkiksi uusien ideoiden tuottamiseen tai olemassa olevien ratkaisuvaihtoehtojen arviointiin?
5. Vaikutus ohjelmointiin ja ohjelmistokehitykseen
    a. Miten generatiiviset tekoälymallit ovat konkreettisesti tukeneet ohjelmointia ja ohjelmistokehitystä, esimerkiksi koodin automaattisena generointina tai kehittäjänä koodin tuottavuuden parantajana?
6. Vaikutus testauksessa ja vaatimusmäärittelyssä
    a. Miten generatiiviset tekoälytyökalut ovat tukeneet testaus- ja laadunvarmistusvaiheita, ja mitä erityisiä etuja tai riskejä olet havainnut tekoälyn soveltamisessa näissä vaiheissa? Esimerkiksi automaattisten testiskenaarioiden luomisessa tai virheiden havaitsemisessa?
7. Tietoturva ja luottamusriskit
    a. Millaisia tietoturvaan ja luottamuksellisuuteen liittyviä riskejä näet syntyvän? Esimerkiksi kun generatiivisille tekoälymalleille annetaan pääsy yrityksen sisäiseen dataan, ja miten näitä riskejä pyritään hallitsemaan?