TURUN YLIOPISTO
Kauppakorkeakoulu

Tekoälyavusteinen ohjelmointi kehittäjän työn näkökulmasta

Tietojärjestelmätieteen kandidaatintutkielma

Laatija(t):
Olli Auranen

Ohjaaja(t):
KTM Tarja Matikka

12.4.2026
Turku

Opiskelijan lausunto tekoälyn käytöstä tähän tutkielmaan liittyen:

☐ **En ole käyttänyt tekoälyä hyödyntäviä työkaluja** tätä tutkielmaa kirjoittaessani.

☒ **Olen käyttänyt tekoälyä hyödyntäviä työkaluja** tätä tutkielmaa kirjoittaessani. Tämä käyttö on dokumentoitu tutkielman liitteessä. Vakuutan, että tekoälyä käytettiin yliopiston ohjeistuksen mukaisella tavalla.

Turun yliopiston laatujärjestelmän mukaisesti tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -järjestelmällä.

Kandidaatintutkielma

Oppiaine: Tietojärjestelmätiede
Tekijä(t): Olli Auranen
Otsikko: Tekoälyavusteinen ohjelmointi kehittäjän työn näkökulmasta
Ohjaaja(t): KTM Tarja Matikka
Sivumäärä: 28 sivua
Päivämäärä: 12.4.2026

Ohjelmistokehittäjien rooli on merkittävässä muutoksessa tekoälyn kehityksen myötä. Erityisesti suurten kielimallien yleistyminen on vaikuttanut laaja-alaisesti kehittäjien työnkuvaan, osaamisvaatimuksiin sekä työn haasteisiin. Tekoälyn roolin kasvaessa sen vaikutusten kokonaisvaltainen ymmärtäminen ohjelmistokehittäjien työssä on keskeistä.

Tämän tutkielman tavoitteena on tarkastella tekoälyä ohjelmistokehittäjien näkökulmasta. Tutkielmassa selvitetään, mihin tehtäviin tekoälyä hyödynnetään, millaisia hyötyjä ja haasteita sen käyttöön liittyy sekä miten se muuttaa kehittäjien roolia ja osaamisvaatimuksia. Tutkielma on toteutettu kirjallisuuskatsauksella, ja aineistona on hyödynnetty viime vuosien aikana julkaistuja tieteellisiä tutkimuksia.

Tulosten perusteella generatiivinen tekoäly ja erityisesti suuret kielimallit tukevat ohjelmistokehitystä esimerkiksi koodin tuottamisessa ja muokkaamisessa, dokumentoinnissa ja ongelmanratkaisussa. Tekoälyn käyttö voi lisätä työn tehokkuutta ja tukea oppimista, mutta siihen liittyy myös haasteita, kuten virheellisten ratkaisujen riski, lisääntynyt kognitiivinen kuormitus sekä mahdollinen osaamisen heikkeneminen.

Tekoäly ei korvaa ohjelmistokehittäjiä, vaan toimii heidän työparinaan. Tämä vapauttaa aikaa, jonka vuoksi kehittäjien rooli on siirtymässä kohti tekoälyn hyödyntämisen ohjaamista, arviointia ja laadunvarmistusta. Tämä korostaa päätöksentekoa, AI-lukutaitoa sekä eettistä vastuuta keskeisinä osaamisvaatimuksina ohjelmistokehittäjien työssä.

Avainsanat: tekoäly, ohjelmistokehittäjä, AI-lukutaito, generatiivinen tekoäly, ohjelmistokehitys, kielimalli

SISÄLLYS

1 Johdanto

Tekoäly on muodostunut merkittäväksi osaksi ohjelmistokehitystä ja perinteisen ohjelmoinnin rinnalle on noussut tekoälyavusteinen ohjelmointi. Viime vuosina on havaittu huomattavaa kasvua pyrkimyksissä hyödyntää tekoälyä ohjelmistokehityksessä. Erilaisia tekoälypohjaisia työkaluja, kuten GitHub Copilotia ja ChatGPT:tä hyödynnetään yhä laajemmin ja ne ovat juurtuneet osaksi ohjelmistokehittäjien päivittäisiä työprosesseja muun muassa ohjelmoinnin, virheiden korjauksen sekä myös ohjelmistosuunnittelun tukena. (Qiu ym., 2025.) Tässä kandidaatintutkielmassa tarkastellaan, miten tekoälyavusteinen ohjelmointi on muuttanut ohjelmistokehittäjien työkuvaa ja osaamisvaatimuksia sekä millaisissa tehtävissä ja millä tavoin näitä työkaluja hyödynnetään. Tutkielmassa käsitellään myös tekoälyn konkreettisia hyötyjä ja haasteita sekä sen vaikutusta kehittäjän rooliin.

Tekoälyn yleistyminen ohjelmointikehityksessä ei ole sattumaa, vaan sen hyödyntämiseen liittyy merkittäviä taloudellisia sekä käytännöllisiä hyötyjä. Tekoälyavusteisen ohjelmoinnin avulla voidaan parantaa kehitystyön tehokkuutta, vähentää ohjelmistokehitykseen kuluvaa aikaa sekä vähentää ohjelmissa esiintyvien virheiden määrää, mikä voi johtaa huomattaviin kustannussäästöihin. (Qiu ym., 2025). Aihe on tästä syystä erittäin merkityksellinen ohjelmistokehittäjille sekä alalla työskenteleville organisaatioille. Aiempi tutkimus osoittaa, että tekoälyavusteisen ohjelmoinnin on havaittu kasvattavan kehittäjien tehokkuutta etenkin koodin luomisessa, virheiden tunnistamisessa sekä kontekstisidonnaisten ongelmien ratkaisemisessa (Taneski ym., 2025). Lisäksi merkittävät investoinnit kielimallien kehittämiseen viittaavat siihen, että kehittäjien ja tekoälyn yhteistyö tulee tulevaisuudessa syvenemään entisestään, jonka seurauksena voidaan saavuttaa yhä luotettavampia ja turvallisempia ohjelmistoratkaisuja (Qiu ym., 2025).

Suurten kielimallien, kuten ChatGPT:n yleistymisen myötä tutkimuksissa on tarkasteltu laajasti kielimallien tuottaman koodin laatua sekä syötteiden vaikutusta kielimallien antamien vastausten laatuun. Sen sijaan ymmärrys siitä miten ohjelmistokehittäjät käytännössä hyödyntävät tekoälyä osana ohjelmistokehitystä ja millaisissa tehtävissä näitä työkaluja käytetään, on jäänyt vähäisemmälle huomiolle. (Xiao ym., 2024.) Tämä aukko tutkimuksessa korostaa tarvetta tarkastella tekoälyavusteista ohjelmointia kehittäjien näkökulmasta.

Tutkielman tavoitteena on vastata seuraaviin tutkimuskysymyksiin:

(1) Mihin ohjelmistokehityksen tehtäviin tekoälyä hyödynnetään ja miten sitä käytetään?

(2) Millaisia hyötyjä ja haasteita tekoälyavusteinen ohjelmointi tuo ohjelmistokehittäjän työhön?

(3) Miten tekoäly muuttaa ohjelmistokehittäjän roolia ja osaamisvaatimuksia?

Tutkimuskysymyksiin vastataan kirjallisuuskatsauksen avulla, jossa tarkastellaan aiempaa tieteellistä tutkimusta tekoälyavusteisesta ohjelmoinnista. Kirjallisuuskatsauksen tavoitteena on tarkastella olemassa olevaa tutkimustietoa ja analysoida sen avulla tekoälyn vaikutuksia ohjelmistokehittäjän työhön, rooliin sekä osaamisvaatimuksiin muodostaen samalla kokonaiskuvan siitä, miten tekoälyavusteinen ohjelmointi vaikuttaa ohjelmistokehitykseen. Tutkielmassa ensimmäisenä käydään läpi tekoälyn konkreettisia käyttötarkoituksia ohjelmistokehittäjän työssä. Seuraavaksi tarkastellaan sen vaikutusta työn tehokkuuteen, työnkuvaan sekä työn kuormittavuuteen ja riippuvuusrakenteisiin. Tämän jälkeen analysoidaan tekoälyn rajoitteita, luotettavuutta, haittoja sekä mahdollisia vaaroja. Lopuksi käsitellään vielä tekoälyn vaikutuksia ohjelmistokehittäjän rooliin sekä sen vaikutusta kehittäjien osaamisvaatimuksiin. Tämän jälkeen tiivistetään vielä käsitellyt asiat samalla vastaten selkeästi jokaiseen tutkimuskysymykseen.

2 Tekoälyn hyödyntäminen ohjelmistokehityksen työtehtävissä

2.1 Tekoälyn käyttö ohjelmistokehittäjän työssä

Suuret kielimallit (Large Language Model, LLM) ovat tekoälymalleja, jotka kykenevät ymmärtämään ja tuottamaan luonnollista kieltä sekä ohjelmakoodia suurten datamäärien perusteella (Vasiliniuc & Groza, 2023). Näiden kielimallien kuten ChatGPT:n ja GitHub Copilotin kehitys on muuttanut ohjelmistokehitystä perustavanlaatuisesti. Nykyisin näitä kielimalleja hyödynnetään useilla ohjelmistokehityksen osa-alueilla, kuten ohjelmistosuunnittelussa, koodin luomisessa sekä erilaisten ongelmien ratkaisemisessa (Qiu ym., 2025). Kielimallien nopea kehitys on lisännyt kiinnostusta ja tarvetta hyödyntää niitä eri tarkoituksissa, mikä näkyy myös ohjelmistokehityksen parissa. Toisaalta vaikka kiinnostus kielimallien hyödyntämiseen on kasvanut huomattavasti, niiden laaja-alainen ja systemaattinen hyödyntäminen on edelleen suhteellisen vähäistä. Kielimallien integroiminen osaksi ohjelmistokehittäjän työtä ei edellytä pelkästään teknistä ymmärrystä kielimallin toiminnasta, vaan myös syvemmän käsityksen ihmisten ja tekoälyn vuorovaikutuksesta työprosesseissa. (Wang ym., 2026.)

Suuret kielimallit ovat tuoneet ohjelmistokehittäjille uuden tavan tuottaa ja muokata koodia niiden tehokkuuden ja helppokäyttöisyyden ansiosta. Kielimallien avulla koodin generointi onnistuu luonnollisen kielen avulla, joka paikkaa ohjelmistokehittäjän mahdollisia puutteita samalla madaltaen kynnystä toimivan ohjelmistoratkaisun kehittämiselle. Nämä kielimallit on suunniteltu tehokkuuden ja skaalautuvuuden kannalta, jonka vuoksi ne ovat erityisen hyvin soveltuvia pitkienkin ohjelmakoodien luomiseen ja muokkaamiseen. (Qiu ym., 2025.) Kehittäjät hyödyntävätkin kielimalleja työssään ensisijaisesti uuden koodin luomiseen sekä olemassa olevan koodin laadun parantamiseen (Xiao ym., 2024). Tekoälyä hyödynnetään myös koodin dokumentaation luomisessa, jolloin koodin toiminta ja sen tarkoitus saadaan kuvattua selkeästi (Vasiliniuc & Groza, 2023). Tämä vähentää kehittäjien työtaakkaa, koska heidän ei tarvitse erikseen laatia kyseistä dokumentaatiota kehitystyön jälkeen ja varmistaa, että koodin toiminta on helposti ymmärrettävissä myös muille kehittäjille. Tekoälypohjaiset työkalut kykenevät myös auttamaan kehittäjää reaaliaikaisesti koodin luomisessa, antavat siitä palautetta, ehdottavat korjauksia, sekä auttavat havaitsemaan ohjelman heikkouksia (Taneski ym., 2025).

2.2 Tekoäly tiedonhankinnan ja opiskelun tukena

Tekoälyn rooli ohjelmistokehityksessä ei jää pelkästään koodigeneraattorin tasolle, sillä generatiivinen tekoäly on muuttanut ohjelmistokehittäjien työnkuvaa myös tiedonhankinnan ja opiskelun näkökulmasta. (Nguyen ym., 2025). Erityisesti suuret kielimallit ovat yhä merkittävämmässä roolissa kehittäjien oppimisen tukena, ja niiden vaikutusta voidaan tarkastella konstruktiivisen oppimisteorian näkökulmasta, jonka mukaan oppijat rakentavat tietoa kokemuksen, reflektion sekä sosiaalisen vuorovaikutuksen kautta. Kielimallien avulla kehittäjät voivat itsenäisesti etsiä tietoa esittämällä kysymyksiä luonnollisella kielellä, johon nämä kielimallit kykenevät tarjoamaan välittömästi kontekstisidonnaisia ehdotuksia ja vastauksia. (Taneski ym., 2025.) Lisana ja Susanto (2026) korostavat myös tekoälyn tuottaman tiedon laatua ja sen merkitystä tekoälyn kanssa tapahtuvaan oppimiseen. Tiedon laatu koostuu sen tarkkuudesta, ajantasaisuudesta sekä relevanssista, jotka ovat keskeisiä tekijöitä kehittäjien oppimiskokemuksen kannalta (Lisana & Susanto, 2026).

Tekoäly tekee siis itsenäisestä oppimisesta aiempaa vuorovaikutteisempaa sekä käyttäjälähtöisempää. Myös ohjelmistokehittäjien subjektiivinen kokemus tekoälyn kanssa tapahtuvasta vuorovaikutuksesta vaikuttaa oppimiseen. Sujuva käyttökokemus sekä kokemus tekoälyn kyvystä ymmärtää omia ongelmia ja tarpeita voivat lisätä motivaatiota ja sitoutumista tekoälyavusteisiin oppimisprosesseihin (Foroudi ym., 2025). Ohjelmistokehittäjän kokema tyytyväisyys tekoälyyn ja sen tarjoamiin vastauksiin toimii keskeisenä tekijänä sen jatkuvassa käytössä, sillä laadukkaat ja hyödylliset vastaukset lisäävät kehittäjän luottamusta tekoälyyn. Tällä on merkittävä vaikutus tekoälyn hyödyntämiseen myös jatkossa. (Lisana & Susanto, 2026).

Uusien asioiden opiskelun yhteydessä kehittäjä voi kohdata kognitiivista ylikuormitusta, motivaation puutetta, sekä puutteellista yksilöllistä tukea (Nguyen ym., 2025). Ilman tekoälyä kehittäjät ovat usein riippuvaisia kokeneempien kehittäjien tuesta ongelmatilanteissa. Kiireellisessä työympäristössä kokeneempien kehittäjien tuen saaminen ei ole kuitenkaan aina mahdollista, mikä viivyttää ongelman ymmärtämistä, oppimista sekä työn etenemistä. (Qiu ym., 2025.) Kielimallit mahdollistavat vuorovaikutuksen, jonka kautta kehittäjät kykenevät selviytymään osasta näistä ongelmatilanteista ilman ulkopuolista apua, sekä pystyvät syventämään omaa ymmärrystään ja edistämään jatkuvan osaamisen kehittämistä (Taneski ym., 2025). Generatiivinen tekoäly ja erityisesti suuret kielimallit voidaan siis nähdä merkittävänä oppimisen tukena niiden vähentäessä riippuvuutta muihin kehittäjiin ja mahdollistaessa aiempaa itseohjautuvamman osaamisen kehittämisen. Tekoälyn tarjoama mahdollisuus yksilölliseen kehittäjän tarpeisiin mukautuvaan tukeen parantaa sekä oppimiskokemusta, että tukee kehittäjien itsenäistä ongelman ratkaisua (Lisana & Susanto, 2026).

Tekoälyn hyödyntäminen ohjelmistokehityksen työtehtävissä tarjoaa merkittäviä uusia mahdollisuuksia, ja sen vaikutukset näkyvät niin työn tehokkuudessa, työprosesseissa kuin ohjelmistokehittäjien arjessa. Vaikka tekoälyllä on epäilemättä positiivinen vaikutus kehittäjien oppimisprosesseihin, Foroudi ym. (2025) korostavat AI-lukutaidon merkitystä sen tehokkaassa hyödyntämisessä. AI-lukutaidolla tarkoitetaan kykyä ymmärtää tekoälyn toimintaa ja rajoitteita. Hyvä AI-lukutaito mahdollistaa tekoälyn tuottamien tuotosten tarkoituksenmukaisen tulkinnan, minkä ansiosta ohjelmistokehittäjät voivat hyödyntää sitä tehokkaammin oppimisen tukena (Foroudi ym., 2025). Tekoäly toimii siis erinomaisena työkaluna ohjelmistokehittäjän tukena, mutta sen tehokas hyödyntäminen edellyttää kehittäjältä riittävää ymmärrystä sen toiminnasta sekä kykyä soveltaa sitä tarkoituksenmukaisesti.

3 Tekoälyn vaikutukset työn tehokkuuteen ja työnkulkuun

3.1 Tekoäly tehokkuuden tukena

Tekoälyn hyödyntämisellä on laaja-alaiset vaikutukset ohjelmistokehitykseen. Kuten aiemmin todettiin, kehittäjät hyödyntävät tekoälyä erityisesti koodin generoimiseen, virheiden korjaamiseen, ongelmien ratkaisemiseen sekä uusien käsitteiden ja konseptien ymmärtämiseen. Ajansäästö ohjelmistokehittäjän työssä tarkoittaa tehtävien suorittamista tehokkaammin erilaisten työkalujen avulla (Lisana & Susanto, 2026). Tekoälyn avulla ohjelmistokehittäjän työ suoraviivaistuu, sekä ongelmatilanteista eteenpäin pääseminen nopeutuu, minkä seurauksena tekoälyllä on todettu olevan merkittävä vaikutus työn tehokkuuteen, kehitysprosessien nopeuteen, mikä lyhentää tehtävien suorittamiseen kuluvaa aikaa (Taneski ym., 2025). Tuottavuuden ja tehokkuuden kasvun lisäksi tekoälyn on havaittu vähentävän kehittäjien työkuormaa, kun osa rutiininomaisista tehtävistä voidaan automatisoida. Näin tekoälyn vaikutuksen voivat ilmetä sekä tehokkuuden kasvuna, että ohjelmistokehittäjien työkuorman vähenemisenä (Chuang ym., 2025).

Lisäksi virheiden määrän väheneminen, laadukkaampien ratkaisujen tuottaminen, sekä ohjelmistojen kehitysajan nopeutuminen voivat tuoda organisaatiolle kustannussäästöjä ja vahvistaa suoraan sen kilpailukykyä (Qiu ym., 2025). Tämän vuoksi tekoälytyökalujen käytettävyys, luotettavuus ja nopea vasteaika ovat keskeisiä tekijöitä, jotta ohjelmistokehittäjien työ olisi mahdollisimman sujuvaa ja tehokasta (Lisana & Susanto, 2026). Tekoäly voi lisätä ohjelmistokehittäjien tehokkuutta merkittävästi, mutta toisaalta sen käyttö voi johtaa lyhyen aikavälin hyötyjen tavoitteluun pitkän aikavälin osaamisen kustannuksella. Vaikka tekoäly siis kasvattaa kehittäjän tehokkuutta, voi liiallinen tukeutuminen sen tuottamiin ratkaisuihin heikentää pitkän aikavälin tehokkuutta sekä kehittäjän osaamisen kehittymistä. (Nguyen ym., 2026.) Tekoäly ei kuitenkaan ole syrjäyttämässä ohjelmistokehittäjiä vaan toimii enemmän yhteistyökumppanina, jossa kasvanut tehokkuus ja vakaammat ohjelmistoratkaisut perustuvat kehittäjän ja tekoälyn väliseen yhteistyöhön (Qiu ym., 2025).

3.2 Ohjelmistokehittäjän työnkuva muutoksessa

Tekoälyn yleistyminen on muuttanut ja tulee jatkossa muokkaamaan merkittävästi ohjelmistokehittäjien työnkuvaa sekä työskentelytapoja. Tekoäly ei ainoastaan tehosta työnkulkua automatisoimalla rutiininomaisia tehtäviä, kuten dokumentaation tuottamista, vaan se mahdollistaa työn painopisteen

siirtymisen vaativampiin ja luovempiin tehtäviin. Tekoälyn tukemana kehittäjät kykenevät suoriutumaan yhä paremmin tehtävästään, mikä mahdollistaa keskittymisen toimivien ja laadukkaiden ohjelmistoratkaisujen kehittämiseen. Rutiininomaisten ja aikaa vievien ohjelmointitehtävien siirtyessä osittain tekoälyn hoidettavaksi erityisesti kokeneemmille kehittäjille vapautuu aikaa korkeamman tason suunnitteluun sekä liiketoimintakriittisen logiikan toteuttamiseen. Samalla ohjelmistokehittäjän rooli on siirtymässä yhä enemmän tekoälyavusteisesti tuotetun koodin arvioimiseen, muokkaamiseen ja laadunvarmistukseen. (Qiu ym., 2025.)

Tekoälyn yleistyessä korostuu myös vastuu tekoälyavusteisesti tuotetun koodin laadusta. Kehittäjien on havaittu kykenevän tunnistamaan ja korjaamaan virheitä omasta koodistaan tehokkaammin, minkä vuoksi oman työn kriittinen arviointi korostuu ja on keskeinen tekijä laadun varmistamisessa (Bartsch ym., 2026). Samalla ohjelmistokehittäjän työssä korostuu koodivastuu, joka viittaa velvollisuuteen avata ja perustella tuotetut ratkaisut. Tekoälyn kehityksen myötä työnkuva on siirtymässä perinteisestä ohjelmoinnista ja toteuttamisesta tekoälyn ohjaamiseen ja koordinoimiseen (Ergün ym., 2026). Tämä muutos korostaa kehittäjän vastuuta ymmärtää tuotettujen ratkaisujen toiminta sekä arvioida niiden soveltuvuus osaksi tarkasteltavaa kokonaisuutta. Tekoälyn tuottamat ratkaisut voivat vaihdella samallakin syötteellä, jonka korostaa kehittäjän roolia tekoälyn tuotosten arvioinnissa ja vaihtoehtojen vertaamisessa (Ergün ym., 2026).

Tekoälyn tuomaan muutokseen liittyy myös uusia osaamisvaatimuksia. Kielimallien tuottaman lopputuloksen on havaittu riippuvan vahvasti annettujen ohjeiden laadusta, minkä vuoksi kehittäjän tulee osata kuvata ongelma sekä sen konteksti mahdollisimman selkeästi ja tarkasti saadakseen tarkoituksenmukaisia ratkaisuja. Iteratiivinen työskentely tekoälyn kanssa erilaisten syötteiden avulla on muodostumassa keskeiseksi työelämätaidoksi. (Vasiliniuc & Groza, 2023.) Aiemmin ohjelmointi perustui enemmän ihmisten väliseen vuorovaikutukseen, mutta nykyisin vuorovaikutus tapahtuu yhä useammin ihmisen ja tekoälyn välillä, minkä vuoksi kyky kommunikoida tekoälyn kanssa on muodostumassa keskeiseksi ohjelmistokehittäjän työelämätaidoksi (Ergün ym., 2026).

Lisäksi tekoäly on muokannut ohjelmistokehittäjien työskentelytapoja erityisesti ongelmanratkaisun sekä tiedonhaun saralla. Aiemmin kehittäjät etsivät ratkaisuja dokumentaatioista, hakukoneista tai keskustelupalstoilta saadakseen vastauksen ongelmiinsa. Nykyisin ongelmia voidaan ratkaista vuorovaikutteisesti keskustelemalla kielimallin kanssa, mikä mahdollistaa kysymysten esittämisen luonnollisella kielellä sekä välittömien, kontekstisidonnaisten ehdotusten saamisen. Tämä nopeuttaa tiedonhakua ja tukee iteratiivista työskentelyä, joka sujuvoittaa koko kehitysprosessia. (Taneski ym., 2025.) Tekoälyn käyttö perustuu pitkälti iteratiiviseen työskentelyyn, jossa kehittäjä on

jatkuvassa vuorovaikutuksessa kielimallin kanssa ja tarkentaa ohjeitaan, kunnes saavutetaan haluttu lopputulos (Ergün ym., 2026).

3.3 Tekoälyn vaikutus kognitiiviseen kuormitukseen ja riippuvuusrakenteisiin

Tekoälyn vaikutukset eivät rajoitu pelkästään työkuvaan, työn tuottavuuteen tai tehokkuuteen, vaan ne ulottuvat myös ohjelmistokehittäjän kokemaan kognitiiviseen kuormitukseen. Kun tekoäly hoitaa osan rutiininomaisista ja aikaa vievistä tehtävistä, kuten koodin luonnostelun tai virheiden etsimisen samalla tukien tiedonhakua ja ongelmanratkaisua, kehittäjille vapautuu aikaa keskittyä työn kannalta olennaisempiin tehtäviin. Tämä voi vähentää työn kuormittavuutta ja helpottaa päivittäistä työskentelyä. (Chuang ym., 2025.) Samalla tekoälyn käyttöönotto voi lisätä työn henkisiä vaatimuksia. Kehittäjät voivat kokea epävarmuutta tekoälyn kehityksestä ja sen vaikutuksista työn tulevaisuuteen, mikä saattaa lisätä psykologista kuormitusta (Chuang ym., 2025). Tätä tukee myös Ergün ym. (2026), joiden mukaan tekoälyn yleistyminen voi aiheuttaa pelkoa jälkeen jäämisestä sekä huolta automaation vaikutuksista omaan työhön. Samalla korostuu johtamisen merkitys, sillä johtajien tehtävänä on luoda työympäristöön emotionaalisia edellytyksiä, kuten avoimuutta ja luottamusta, jotka tukevat uusien työskentelytapojen omaksumista ja lieventävät muutokseen liittyvää kuormitusta (Ergün ym., 2026).

Kehittäjien kokemukset tekoälyn käytöstä vaikuttavat myös merkittävästi siitä syntyvään kognitiiviseen kuormitukseen. Positiiviset kokemukset helpottavat tekoälyn käyttöön sitoutumista, kun taas negatiiviset kokemukset lisäävät kognitiivista kuormitusta heikentäen tekoälyn hyödyntämisen tehokkuutta (Foroudi ym., 2025). Jatkuvasti muuttuva teknologinen ympäristö edellyttää uudenlaisten työskentelytapojen omaksumista, mikä voi vaikuttaa kehittäjien henkiseen hyvinvointiin ja lisätä uupumisen riskiä (Chuang ym., 2025). Generatiivinen tekoäly (GenAI) tarkoittaa tekoälyä, joka kykenee datan tutkimisen ja järjestelemisen lisäksi tuottamaan uutta sisältöä sen perusteella. Ohjelmistokehittäjän työssä generatiivisella tekoälyllä viitataan usein kielimalleihin, jotka ovat erikoistuneet tekstin ymmärtämiseen ja tuottamiseen. Yang ym. (2026) puhuvat GenAI-johteisesta uupumuksesta ja korostavat erityisesti syötteen muotoilun, vastausten arvioinnin ja epävarmuuden hallinnan lisäävän riskiä mahdolliselle uupumiselle.

Tekoälyn yleistyminen vaikuttaa myös työn merkityksellisyyden ja onnistumisen kokemuksiin tekoälyn viedessä rutiinitehtävät, joista kehittäjät ovat aiemmin voineet saada pieniä onnistumisen tunteita (Ergün ym., 2026). Tekoälyn hyödyntämisen ja ajattelun ulkoistamisen välinen ero on keskeinen ohjelmistokehityksessä. Pahimmassa tapauksessa kehittäjä voi tukeutua liikaa tekoälyyn

myös rutiinitehtävien ulkopuolella, jolloin oma analyysi ja ongelmanratkaisu jää vähemmälle ja kehittäjän kokonaisymmärrys heikkenee (Nguyen ym., 2026). Näin tekoäly voi siis samanaikaisesti keventää työn kuormittavuutta, mutta lisätä työn henkisiä vaatimuksia ja luoda uusia riskitekijöitä.

Ohjelmistokehitystyön rytmi sekä riippuvuusrakenteet ovat myös jatkuvassa muutoksessa tekoälyn vuoksi. Kehitystyö on muuttunut entistä vuorovaikutteisemmaksi, kun kehittäjät tuottavat koodia kielimallien avulla, arvioivat saatuja tuloksia sekä tarkentavat ohjeita tarvittaessa (Qiu ym., 2025). Siinä missä jatkuva ratkaisujen arviointi on keskeinen osa tekoälyavusteista ohjelmointia, kehittäjän kokema luottamus tekoälyyn ja sen tuottamiin ratkaisuihin voi vähentää epävarmuutta ja kasvattaa itsevarmuutta. Tämä puolestaan tukee tekoälyn jatkuvaa käyttöä sekä voi vähentää työstä sekä tekoälyn käytöstä syntyvää kognitiivista kuormitusta. (Lisana & Susanto, 2026.) Tekoälyn käytössä keskeistä on tasapainon löytäminen sen hyödyntämisen ja oman osaamisen kehittämisen välillä. Tekoälyn tarjoama nopea ongelmanratkaisukyky voi ohjata kehittäjiä suosimaan lyhyen aikavälin tehokkuutta, kun monet tehtävät hoituvat nopeasti ja vaivattomasti. Tämä voi kuitenkin tulla oman syvällisemmän ymmärryksen ja oppimisen kustannuksella. (Nguyen ym., 2026.)

Tekoälyn muodostuessa yhä keskeisemmäksi osaksi kehittäjän arkea myös riippuvuus sen saatavuuteen sekä sen tarjoamiin ratkaisuihin kasvaa. Tekoälyn käytölle on ominaista myös ilmiö, jossa käyttäjät hyödyntävät tekoälyä, vaikka tiedostavat siihen liittyvät riskit, mikä vahvistaa riippuvuusrakenteiden muodostumista ohjelmistokehittäjän työssä (Nguyen ym., 2026). Tekoälyn käyttö kehitystyössä edellyttääkin ohjelmistokehittäjältä hyvää ymmärrystä sen mahdollisuuksista, mutta myös sen riskeistä sekä sen käyttöön liittyvistä vaaroista (Cousineau ym., 2025). Tekoälyn vaikutukset ohjelmistokehitykseen eivät rajoitu pelkästään työn tehokkuuden kasvuun tai työskentelytapojen muutokseen, vaan ne ulottuvat kokonaisvaltaisesti kehittäjän työhön vaikuttaen myös päivittäiseen työrytmiin sekä työn kuormittavuuteen.

4 Tekoälyavusteisen ohjelmoinnin haasteet ja riskit

4.1 Tekoälyn rajoitteet ja luotettavuus ohjelmistokehityksessä

Kuten edellisessä luvussa käsiteltiin, tekoäly ja erityisesti kielimallien kehitys on muuttanut kokonaisvaltaisesti ohjelmistokehitystä. Kuitenkaan kielimallien suoriutuminen ei ole aina virheetöntä ja ne voivat ajoittain tuottaa virheellisiä tai jopa mahdottomia ratkaisuja. Lisäksi niiden käyttö saattaa aiheuttaa laatuongelmia ja vaikeuttaa ohjelmakoodin ymmärtämistä, mikä heikentää ylläpidettävyyttä. (Taneski ym., 2025.) Generatiiviseen tekoälyyn liittyy sisäänrakennettuja riskejä, kuten epätarkkuudet, tekoälyn hallusinaatiot, selitettävyysongelmat sekä vinoumat datassa. Erityisesti hallusinaatiot ovat keskeinen ongelma, sillä tekoäly voi ajoittain tuottaa täysin virheellistä sisältöä esittäen sen kuitenkin vakuuttavasti faktatietona. (Lucas ym., 2025.)

Koska kielimallien tuottamien ratkaisujen oikeellisuutta ei voi pitää itsestään selvyytenä, sitä ei voi käyttää ilman jatkuvaa valvontaa. Ongelmaksi tekoälyn hyödyntämisessä voi myös muodostua kehittäjien taipumus hyödyntää itselle käytännöllisimpiä tekoälytyökaluja, jotka eivät aina ole linjassa tekoälyn luotettavuuden kanssa (Cousineau ym., 2025). Tekoälyn luotettavuuden kannalta keskeistä on sen tuottaman tiedon laatu, joka muodostuu tiedon tarkkuudesta, ajankohtaisuudesta ja relevanssista. Puutteet näissä tekijöissä voivat heikentää tekoälyn hyödyllisyyttä ja käytettävyyttä ohjelmistokehityksessä sekä johtaa puutteellisiin tai jopa virheellisiin ratkaisuihin. (Lisana & Susanto, 2026.)

Kehittäjän vastuulle jää siis tuotettujen ratkaisujen tarkastaminen, arviointi ja tarvittaessa niiden korjaaminen. Ohjelmistokehityksessä suuri osa haavoittuvuuksista ja heikkouksista johtuu tahattomista virheistä, jotka olisivat usein vältettävissä huolellisemmalla koodin tarkastelulla ennen käyttöönottoa (Bartsch ym., 2026). Tekoälyavusteinen ohjelmointi nopeuttaa kehitysprosessia, mikä korostaa kriittisen arvioinnin merkitystä. Nopeamman kehityksen myötä virheellinen koodi voi syntyä nopeasti ja jäädä helpommin huomaamatta ilman systemaattista tarkastelua.

Cousineau ym. (2025) korostavat myös tekoälyn yleistymisen kasvattavan tarvetta luotettavien tekoälyjärjestelmien kehittämiselle. Luotettavuudella viitataan järjestelmien kykyyn tuottaa johdonmukaisia, luotettavia ja eettisesti hyväksyttäviä ratkaisuja. Tekoälyn luotettavuuden ja siihen liittyvien riskien analysointi ei kuitenkaan aina ole yksiselitteistä, sillä ne riippuvat tekoälyjärjestelmän monimutkaisuudesta sekä käyttöympäristöstä (Cousineau ym., 2025). Luotettavien ratkaisujen

keskeisyyden vuoksi ohjelmoinnissa myös kehotteiden eli promptien huolellinen muotoilu korostuu, sillä niiden avulla voidaan ohjata kielimallia tuottamaan tarkoituksenmukaisempia ratkaisuja (Taneski ym., 2025).

Tekoälyn käyttöön liittyy myös riski liian suurista odotuksista. Organisaatiot ja kehittäjät saattavat nähdä kielimallit ratkaisuna erilaisiin ongelmiin ja asettaa niille epärealistisia suorituskykyvaatimuksia. Kielimalleilla on kuitenkin selviä rajoitteita, joiden huomioiminen on niiden tehokkaan hyödyntämisen kannalta keskeistä. Edellä kuvattu odotusten ja todellisen suorituskyvyn välinen ristiriita muodostuu usein kielimallien laajamittaisen käyttöönoton esteeksi organisaatioissa. (Wang ym., 2026.) Organisaatioiden tulee myös kyetä lisäämään kehittäjien kokemaa vastuuta tuotetusta koodista ilman, että he kuormittaisivat kehittäjien työtä. Mikäli kehittäjät eivät koe vastuuta tuottamastaan koodista, voi motivaatio kriittiseen analyysiin heiketä, mikä puolestaan lisää virheiden ja laadullisten ongelmien riskiä. (Bartsch ym., 2026.) Tämän lisäksi tekoälyn puutteellinen integrointi osaksi päivittäisiä työprosesseja voi muodostaa merkittävän esteen sen tehokkaalle hyödyntämiselle, sillä se estää tekoälyn vakiintumisen osaksi päivittäistä työskentelyä. Tämä vaatii johdolta aktiivisia toimenpiteitä tämän integraation tukemiseksi. (Ergün ym., 2026.)

4.2 Liiallinen riippuvuus ja osaamisen kehittyminen

Siinä missä tekoälyllä on suuri potentiaali ohjelmistokehityksen tehostamisessa, sen käyttöön liittyy myös riski liiallisen riippuvuuden syntymisestä. Vaikka kielimallit tuottavat lähtökohtaisesti laadukasta ja turvallista koodia, niiden varaan liiallisesti tukeutuminen voi heikentää kehittäjien kykyä tunnistaa virheitä sekä arvioida ohjelmistojen turvallisuutta ja laatua (Qiu ym., 2025). Lisäksi tekoälyavusteisten työkalujen käyttö voi johtaa tilanteisiin, joissa ongelmia ratkaistaan ilman tarvittavaa ymmärrystä niiden vaatimuksista ja toimintalogiikasta (Taneski ym., 2025). Puutteellinen ymmärrys heikentää ongelman määrittelyä ja voi siten vähentää kielimallien tarjoamien ratkaisujen tarkoituksenmukaisuutta, jonka vuoksi kehittäjän rooli laadun varmistamisessa korostuu.

Liiallisen riippuvuuden välttämiseksi tekoälyä tulisi hyödyntää tasapainoisesti perinteisten ongelmanratkaisumenetelmien rinnalla (Taneski ym., 2025). Tekoälyn epävarmuuden ja sen tuomien riskien ymmärtäminen on yhä tärkeämpää ohjelmistokehittäjän työssä. Kehittäjien on myös tärkeä ymmärtää, että tekoälyn käyttöön sisältyy ilmiö, jossa käyttäjät tukeutuvat suuresti tekoälyyn, vaikka tiedostavat sen vaarat ja mahdolliset negatiiviset vaikutukset omalle osaamiselle (Nguyen ym., 2026). Vaarojen tunnistamisen lisäksi luottamus tekoälyjärjestelmiin on kuitenkin keskeinen tekijä

niiden käytössä, sillä se vähentää kehittäjän epäröintiä ja tekee niiden hyödyntämisestä sujuvampaa sekä tehokkaampaa (Lisana & Susanto, 2026). Liiallinen luottamus voi kuitenkin johtaa tilanteeseen, jossa kehittäjä hyväksyy tekoälyn tuottamat ratkaisut ilman riittävää kriittistä arviointia, mikä voi heikentää oman ymmärryksen kehittymistä.

Kuten aiemmassa kappaleessa todettiin, liiallinen riippuvuus voi johtaa puutteelliseen ymmärrykseen, ja siten osaamisen rapautumiseen (Taneski ym., 2025). Kehittäjät saattavat usein vain kopioida ja liittää tekoälyn tuottamia ratkaisuja, jolloin kriittinen arviointi ja oman ymmärryksen luominen jää vähemmälle. Lisäksi tekoälypohjaiset työkalut perustuvat menneeseen dataan, eivätkä ne aina kykene hyödyntämän uusimpia työkaluja ja ratkaisutapoja (Qiu ym., 2025). Tekoäly voi kuitenkin olla hyvänä apuvälineenä etenkin nuoremmille kehittäjille tehtävän selittämisessä ja ongelmanratkaisun tukena. Tärkeää on kuitenkin varmistaa, että kehittäjä todella ymmärtää ratkaisun ja sen taustalla olevan logiikan, sillä riskinä on pelkän vastauksen hyväksyminen ilman kriittistä pohdintaa. (Qiu ym., 2025.) Tekoälyn hyödyntäminen oppimisessa edellyttää ohjelmistokehittäjältä itseohjautuvuutta, reflektointia sekä aktiivista vuorovaikutusta työkalun kanssa (Nguyen ym., 2025). Nämä ovat keskeisiä tekijöitä osaamisen kehittymisessä, niiden puuttuminen voi heikentää oppimista ja johtaa osaamisen heikkenemiseen. Näin ollen tekoälypohjaisia työkaluja ei tulisi tarkastella vain niiden teknisten mahdollisuuksien näkökulmasta, vaan osana monimutkaisia sosioteknisiä järjestelmiä, jotka vaikuttavat kehittäjän oppimiseen, osaamiseen sekä työskentelytapoihin (Wang ym., 2026).

4.3 Tekoälyn psykologiset, eettiset ja organisatoriset vaikutukset

Tekoäly voi vaikuttaa ohjelmistokehittäjien työn tehokkuuteen ja työkuormaan, mutta se voi myös aiheuttaa teknostressiä ja heikentää hyvinvointia, mikä tekee sen vaikutuksista kaksijakoisia. Parhaimmillaan tekoäly parantaa tuottavuutta sekä vähentää työ-perhe-konflikteja, mutta toisaalta kehittäjät voivat kokea pelkoa työpaikan menettämisestä sekä stressiä oman työn tulevaisuudesta tekoälyn nopean kehityksen seurauksena (Chuang ym., 2025). Tekoälyn tuottamat ratkaisut voivat lisätä epävarmuutta työssä. Kehittäjä voi kyseenalaistaa oman syötteen tarkkuutta sekä epäillä kielimallin kykyä ymmärtää annettu tehtävä sen perusteella. Epävarmuus voi vähentää kehittäjien luottamusta tekoälyjärjestelmiin. Tämä taas voi tehdä heistä haluttomampia käyttämään näitä järjestelmiä tulevaisuudessa. (Lisana & Susanto, 2026.)

Vaikka syöte pysyisi samana, kielimalli voi silti tuottaa vaihtelevia vastauksia, joskus ylittäen odotukset ja ajoittain epäonnistuen tehtävässään. Tämä epävarmuus vaikuttaa paitsi psyykkiseen hyvinvointiin, myös motivaatioon työskennellä iteratiivisesti näiden järjestelmien kanssa. Lisäksi epävarmuus voi vaikuttaa kehittäjien sitoutumiseen ja halukkuuteen antaa palautetta. (Yang ym., 2026.) Tekoälyllä on huomattavia positiivisia vaikutuksia, mutta sen kehittyessä ja integroituessa yhä tiiviimmin osaksi kehittäjien työtä, myös sen väärinkäytön mahdollisuudet ja eettiset haasteet lisääntyvät (Lucas ym., 2025).

Tekoälyn tuottama epävarmuus voi heikentää työhyvinvointia, luovuutta sekä kykyä oppia uutta. Vaikka tekoälyllä on lukuisia positiivisia vaikutuksia, nämä psykologiset riskit voivat heikentää sen kokonaisvaltaista hyötyä. Tämän vuoksi organisaatioiden tulisi ottaa käyttöön toimintatapoja, jotka turvaavat kehittäjien hyvinvoinnin tekoälyä hyödyntävässä työympäristössä. (Chuang ym., 2025.) Organisaatioiden tulisi panostaa positiivisten kokemusten luomiseen tekoälyn hyödyntämisessä, sillä ne lisäävät ohjelmistokehittäjien sitoutumista sen käyttöön. Samanaikaisesti organisaatioiden tulisi kyetä vahvistamaan kehittäjien kokemaa vastuuta tuottamastaan koodista. Tähän vastuuseen vaikuttavat muun muassa työn seurannan taso, koodin jäljitettävyys tekijäänsä sekä odotukset työn arvioinnista. (Bartsch ym., 2026.)

Koetun vastuun lisääminen voi kuitenkin kasvattaa kehittäjien kuormitusta. Näin ollen työn laadun varmistaminen saattaa tapahtua työhyvinvoinnin kustannuksella etenkin silloin, koodivastuuta korostetaan ilman riittäviä kannustimia. Samanaikaisesti tekoälyn kanssa tapahtuva vuorovaikutus vaikuttaa merkittävästi kehittäjien tunnekokemuksiin. Positiiviset kokemukset voivat lisätä motivaatiota, kun taas negatiiviset kokemukset voivat heikentää sitoutumista ja lisätä työn kuormittavuutta. (Foroudi ym., 2025.) Tekoälyjärjestelmiä kehittäessä tärkeää on siis ottaa käyttäjän perspektiivi huomioon. Luotettavan tekoälyn kehittäminen ei ole pelkästään yksittäisten kehittäjien vastuulla vaan se edellyttää organisaatiotason ohjausta ja selkeitä toimintamalleja (Cousineau ym., 2025). Tekoälyavusteisten toimintatapojen käyttöönotto edellyttää kokeiluun kannustavaa kulttuuria sekä sallivaa suhtautumista virheisiin. Nämä tekijät tukevat tekoälyn hyödyntämistä ja auttavat kehittäjiä omaksumaan sen käytön ilman liiallista epäonnistumisen pelkoa (Ergün ym., 2026).

Tekoäly tuo ohjelmistokehitykseen merkittäviä hyötyjä, mutta se sisältää myös turvallisuuteen ja eettisyyteen liittyviä riskejä. Tekoälyn tuottama koodi voi olla toimiva ja syntaksisesti oikein, mutta se saattaa sisältää turvallisuusaukkoja ja kehittäjät voivat luottaa liikaa koodin oikeellisuuteen, tekoälyn vakuuttavuuden ja tehokkuuden vuoksi. Tämä lisää tarvetta jatkuvalle valvonnalle, sekä tekoälyn tuottamien ratkaisujen tarkistamiselle. (De ym., 2025.) Nguyen ym. (2026) tukevat tätä

näkemystä tuoden esille käyttäjien haasteet tekoälyn tuoman kätevyyden sekä eettisen vastuun tasapainottamisessa. Tasapainottamista vaikeuttavat myös tekijät, kuten kehittäjän impulsiivisuus, sekä puutteelliset kannustimet vastuullisen toimintatavan noudattamiseen. Lisäksi työelämän aikapaineet sekä tehokkuusvaatimukset voivat saada kehittäjät tukeutumaan liiallisesti tekoälyyn, jossa voi painottua nopeus kriittisen arvioinnin kustannuksella. (Nguyen ym., 2026.) Nämä havainnot korostavat entisestään kehittäjän kykyä arvioida kriittisesti tekoälyn tuottamia ratkaisuja sekä omaa toimintaa sekä suhdetta tekoälyn käyttöön. Yang ym. (2026) tuovat esille miten luotettavan tuloksen saamiseksi jokainen iteraatio vaatii kehittäjältä selvyyden, relevanssin sekä tarkkuuden jatkuvaa arvioimista. Oman syötteen jatkuva kehittäminen ja saatujen tulosten epävarmuus voi kasvattaa henkisen väsymisen riskiä (Yang ym., 2026).

Kätevyyden ja vastuun tasapainottamisen lisäksi tekoälyn käytössä korostuvat myös eettisen kysymykset, kuten vastuunjako tekoälyn tekemistä virheistä, tietosuoja sekä väärinkäytön mahdollisuus (Lucas ym., 2025). Tekoälyjärjestelmien suunnittelu ja käyttö vaatii sekä ihmiskeskeistä että teknistä ajattelua, jotta voidaan tasapainottaa tekoälyn hyödyt turvallisuuden ja eettisyyden kanssa (De ym., 2025). Vaihteleva koodin laatu heikentää ohjelmien toimivuuden varmuutta ja turvallisuutta, mikä muodostaa merkittävän riskin organisaatoille. Cousineau ym. (2025) mukaan tekoälyn eettistä ja luotettavaa käyttöä vaikeuttaa yhtenäisten käytäntöjen puute, mikä luo epävarmuutta siitä, miten tekoälyä tulisi hyödyntää. Tämän vuoksi organisaatioiden rooli selkeiden tekoälyn käyttöön liittyvien ohjeistuksien laatimisessa on keskeinen. Ohjeistukset ehkäisevät osaltaan tekoälyn tuomaa epävarmuutta, yhdenmukaistavat esimiesten ja kehittäjien odotuksia sekä vähentävät riskiä liiallisesta tukeutumisesta tekoälyyn (Nguyen ym., 2026). Tämän lisäksi on tärkeää, että kehittäjillä on yhteinen tekoälyn kyvykkyyksistä ja rajoista sekä siitä, milloin kontrolli tulee siirtää takaisin ihmiselle (Ergün ym., 2026). Kehittäjien motivointi ja vastuun kasvattaminen koodin tarkastamisessa ovat keskeisiä toimenpiteitä koodin turvallisuuden ja toimivuuden varmistamiseksi (Bartsch ym., 2026).

5 Tekoälyn vaikutukset ohjelmistokehittäjien rooliin ja osaamisvaatimuksiin

5.1 Ohjelmistokehittäjien rooli muutoksessa

Ohjelmistokehittäjien rooli on muuttumassa tekoälyn kehityksen myötä. Erityisesti suuret kielimallit ovat olleet suuressa roolissa ohjelmistokehittäjien roolin muutoksessa. Kehittäjiltä vaaditaan yhä enemmän kykyä siirtyä pelkästä koodin luomisesta laajempaan rooliin, joka sisältää päätöksentekoa, eettistä valvontaa sekä yhteistyötä tekoälyjärjestelmien kanssa. Työn painopiste on siirtymässä koodin kirjoittamisesta kohti tekoälyn tuottaman sisällön ohjaamista, arvioimista sekä laadunvarmistusta. Tämä muutos korostaa ihmisen ja tekoälyn toisiaan täydentäviä vahvuuksia, kuten ihmisten kontekstuaalista ymmärrystä sekä tekoälyn kykyä käsitellä suuria datamääriä ja tunnistaa rakenteita. (De ym., 2025.)

Ohjelmistokehittäjän rooli on siirtymässä yhä enemmän laadunvarmistamiseen, mikä korostaa vastuuta tuotetun koodin laadusta (Bartsch ym., 2026). Samalla tekoälyn vakiintuessa keskeiseksi työvälineeksi kasvaa riski siihen liiallisesta tukeutumisesta, mikä voi heikentää kehittäjän itsenäistä ongelmanratkaisukykyä (Nguyen ym., 2026). Tämän myötä ohjelmistokehittäjän roolissa korostuvat pitkän aikavälin ajattelu, vastuullisuus sekä kehittäjän kyky säädellä omaa toimintaansa. Kehittäjän on kyettävä välttämään lyhyen aikavälin tehokkuuden ja helppouden priorisointia tilanteissa, joissa se tapahtuu pitkän aikavälin osaamisen kustannuksella (Nguyen ym., 2026). Näin ollen ohjelmistokehittäjän roolissa keskeiseksi nousee kyky arvioida kriittisesti omaa toimintaa sekä toimintatapoja.

Kehittäjät osallistuvat yhä enemmän kriittiseen arviointiin ja strategiseen päätöksentekoon tekoälytyökalujen integroituessa jokapäiväisiin ohjelmoinnin työtehtäviin (De ym., 2025). Qiu ym. (2025) mukaan tekoäly integroituu entistä tiiviimmin kehitysympäristöihin vuoteen 2030 mennessä, mikä muuttaa kehittäjien päivittäisiä työtehtäviä edelleen. Samalla korostuvat kehittäjien vastuut järjestelmäarkkitehtuurin suunnitellussa, tietoturvassa ja suorituskyvyn optimoinnissa. Kehittäjän rooli on siirtymässä siis yhä enemmän tekoälyn ohjaajaksi ja valvojaksi, mikä muuttaa ohjelmistokehittäjien roolia ja identiteettiä (Qiu ym., 2025). Kehittäjät, jotka identifioivat itsensä ongelmanratkaisijoiksi, näkevät tekoälyn todennäköisemmin hyödyllisenä työkaluna, kun taas ne, jotka arvostavat syvällistä teknistä osaamista, saattavat suhtautua kriittisemmin tekoälytyökalujen hyödyntämiseen (De ym., 2025). Roolin muutos herättää siis kehittäjissä erilaisia reaktioita riippuen heidän ammatillisesta identiteetistään.

5.2 Uudet osaamisvaatimukset tekoälyavusteisessa kehityksessä

Luonnollisesti tämä nopea roolin muutos tuo mukanaan uusia osaamisvaatimuksia. Ohjelmistokehittäjien työssä ei enää pärjää pelkällä ohjelmointistaidolla, sillä muuttuva työnkuva edellyttää yhä monipuolisempia valmiuksia. Yksi keskeisimmistä uusista osaamisalueista on kyky ohjata kielimalleja tehokkaasti. Kehittäjän tulee osata muotoilla tehtävät ja niiden konteksti riittävän tarkasti, sillä annettujen kehotteiden laatu vaikuttaa merkittävästi tekoälyn tuottamien ratkaisujen tarkoituksenmukaisuuteen. Kyky laadukkaiden kehotteiden luomiseen on muodostumassa yhä tärkeämmäksi osaksi ohjelmistokehittäjän ammattitaitoa. (Vasiliniuc & Groza, 2023.)

Samalla kehittäjien on kyettävä arvioimaan kielimallien tuottamien ratkaisujen toimivuutta, turvallisuutta sekä eettisyyttä annetussa kontekstissa. Tekoälyn tuottamien ohjelmien mahdolliset haavoittuvuudet edellyttävät kehittäjiltä kykyä sekä tekniseen osaamiseen, kriittiseen ajatteluun sekä tehokkaaseen yhteistyöhön ja viestintään (De ym., 2025). Niin sanottu AI-lukutaito, eli ymmärrys tekoälyn tarjoamista mahdollisuuksista ja vahvuuksista, mutta toisaalta myös ymmärrys sen rajoitteista sekä heikkouksista on siis muodostumassa yhä keskeisemmäksi työelämätaidoksi ohjelmistokehittäjän roolissa (Foroudi ym., 2025). Kyseinen taito on välttämätön tekoälyn tehokkaan ja turvallisen hyödyntämisen kannalta.

Nykypäivän nopeasti muuttuva teknologinen ympäristö edellyttää ohjelmistokehittäjiltä jatkuvaa oppimista ja uusien taitojen omaksumista, jotta he voivat menestyä tekoälyavusteisessa kehitystyössä. Nämä uudet taidot eivät ole pelkästään teknisiä vaan, osa niistä liittyy kehittäjän muuttuvaan rooliin strategisessa ja vastuullisessa ajattelussa. Ohjelmistokehittäjän roolin henkinen kuormitus voi myös kasvaa tekoälyavusteisten toimintatapojen yleistyessä. (Lisana & Susanto, 2026.) Tekoälyn käyttöön liittyvä epävarmuus sekä oman työn tulevaisuuden, että tekoälyn tuottamien ratkaisujen luotettavuuden osalta yhdistettynä työn aikapaineisiin korostaa ohjelmistokehittäjän roolissa henkisen sietokyvyn merkitystä (Chuang ym., 2025; Lisana & Susanto, 2026). Epävarmuuden hallinta ja paineensietokyky nousevat yhä keskeisimmiksi taidoiksi, jotta kehittäjä kykenee ylläpitämään tehokkuuden sekä tekemään harkittuja päätöksiä. Ohjelmistokehittäjän roolin haasteet ovat siirtymässä teknisestä osaamisesta yhä enemmän henkisen jaksamisen, eettisten kysymysten sekä tekoälyn tasapainoisen hyödyntämisen suuntaan. Kehittäjän tulee samanaikaisesti osata käyttää tekoälyä tehokkaasti ja turvallisesti sekä ylläpitää ja kehittää omaa osaamistaan.

6 Yhteenveto ja johtopäätökset

Tutkielman tarkoituksena oli tarkastella tekoälyä ohjelmistokehittäjien näkökulmasta. Tekoäly on kehittynyt viime vuosina nopeasti ja etenkin generatiivinen tekoäly sekä kielimallit ovat saavuttaneet laajaa suosiota. Samanaikaisesti kehittäjien rooli on ollut murrosvaiheessa tekoälyn kyetessä yhä paremmin kontekstisidonnaiseen ongelmanratkaisuun. Tekoäly toimii yhtäältä kehittäjien työkaluna, mutta toisaalta muokkaa heidän työnkuvaansa, osaamistarpeita sekä roolia. Tekoälyn kehitykselle ei ole nähtävissä selkeää päätepistettä ja sen vaikutus ohjelmistokehittäjien työhön tulee todennäköisesti kasvamaan tulevaisuudessa, jonka vuoksi aiheen käsittely on tärkeää ja ajankohtaista. Aiempi tutkimus osoittaa, että tekoälyllä on laaja-alaisia vaikutuksia ohjelmistokehittäjien päivittäiseen työskentelyyn. Sen on todettu vaikuttavan muun muassa työn tehokkuuteen, kuormitukseen sekä osaamisen kehittämiseen.

Vaikutukset eivät kuitenkaan ole yksiselitteisesti myönteisiä, sillä tekoäly voi joissain tapauksissa heikentää kehittäjien osaamista työhyvinvointia ja jaksamista. Tämän tutkielman tavoitteena oli tarkastella tekoälyn vaikutuksia ohjelmistokehittäjän työhön monipuolisesti ja tasa-painoisesti tuoden esiin sekä hyödyt, että haitat. Tutkielmassa pyrittiin välttämään yksipuolista tekoälyn korostamista ja tarkastelemaan ilmiötä mahdollisimman realistisesti. Tekoälyn vaikutuksia analysoitiin kolmen tutkimuskysymyksen avulla. Näiden kysymysten pohjalta tutkielma jäsenneltiin neljään käsittelylukuun, joiden tavoitteena oli vastata esitettyihin tutkimuskysymyksiin. Seuraavaksi tarkastellaan kutakin tutkimuskysymystä erikseen ja niihin muodostettuja vastauksia.

Tutkielman ensimmäinen tutkimuskysymys oli ”Mihin ohjelmistokehityksen tehtäviin tekoälyä hyödynnetään ja miten sitä käytetään?”. Kysymyksen tavoitteena oli hahmotella tekoälyn konkreettisia käyttökohteita sekä sen tuottamaa hyötyä ohjelmistokehittäjien työssä. Erityisesti suuret kielimallit, kuten ChatGPT ovat saavuttaneet laajaa suosiota kehittäjien keskuudessa. Ne kykenevät käsittelemään luonnollista kieltä sekä ymmärtämään kehittäjien ongelmia kontekstisidonnaisesti, mikä mahdollistaa tarkoituksenmukaisten ratkaisujen tuottamisen. Tekoälyä hyödynnetään erityisesti koodin kirjoittamisessa ja muokkaamisessa, dokumentaation tuottamisessa sekä erilaisten ongelmien ratkaisemisessa. Kielimallien keskeisenä vahvuutena voidaan pitää vuorovaikutteisuutta sekä iteratiivista toimintamallia, jossa kehittäjä voi tarkentaa syötettään ja esittää jatkokysymyksiä tavoitellun lopputuloksen saavuttamiseksi. Tekoälyn käyttö ei kuitenkaan poista kehittäjän vastuuta, sillä sen tuottamat ratkaisut eivät ole aina luotettavia. Tämän vuoksi tekoälyn hyödyntäminen edellyttää kriittistä arviointia eikä oman ajattelun ulkoistaminen ole perusteltua. Tekoälyä hyödynnetään myös oppimisen ja tiedonhaun tukena. Sen kyky tarjota kontekstisidonnaista ja tarkasti kohdennettua

tietoa helpottaa erityisesti sellaisten ongelmien ratkaisemista, joiden selvittäminen perinteisin menetelmin voisi olla aikaa vievää. Lisäksi tekoälyn nopeus ja vuorovaikutteisuus tukevat itseohjautuvaa oppimista sekä vähentää riippuvuutta kokeneemmista kehittäjistä. Tekoälyn hyödyntäminen sekä oppimisessa että työskentelyssä edellyttää kuitenkin kehittäjiltä riittävää AI-lukutaitoa. Tämä tarkoittaa kykyä ymmärtää tekoälyn toimintaperiaatteita ja rajoitteita sekä hyödyntää sitä tarkoituksenmukaisesti eri tilanteissa.

Tutkimuksen toinen tutkimuskysymys oli ”Millaisia hyötyjä ja haasteita tekoälyavusteinen ohjelmointi tuo ohjelmistokehittäjän työhön?” Tavoitteena oli tarkastella, miten tekoälyn käyttökohteet todellisuudessa parantavat kehittäjän työtä sekä toisaalta millaisia haasteita sen käyttöön liittyy. Tutkielmassa hyödyt ja haasteet käsiteltiin erillisinä kokonaisuuksina ja sama jäsentely säilytetään myös yhteenvedossa. Tekoälyn keskeisenä hyötynä voidaan pitää ohjelmistokehityksen tehokkuuden kasvua. Tekoäly tukee koodin tuottamista, virheiden tunnistamista sekä ongelmanratkaisua. Vaikka tekoäly ei korvaa kehittäjien syvällistä teknistä osaamista, se mahdollistaa rutiininomaisten tehtävien osittaisen ulkoistamisen, mikä vapauttaa aikaa vaativampiin tehtäviin ja organisaatioiden näkökulmasta kasvanut tehokkuus voi tuottaa myös kustannussäästöjä. Lisäksi nopea palautteen saaminen ja iteratiivinen työskentelytapa voivat parantaa tuotetun koodin laatua ja vähentää siinä olevien virheiden määrää. Tekoäly lisää työn vuorovaikutteisuutta ja tehostaa tiedonhakua sekä toimii opiskelun tukena. Kehittäjät ovat vähemmän riippuvaisia toisistaan, mikä voi nopeuttaa työskentelyä ja vähentää odotusaikoja. Osaamisen kehittäminen muuttuu yhä itseohjautuvammaksi, kun tekoäly tarjoaa tuke myös sellaisissa tilanteissa, joissa kokeneempien kehittäjien apua ei ole välittömästi saatavilla.

Vaikka tekoäly voi hoitaa rutiinitehtäviä ja toimia kehittäjän tukena, se ei ole virheetön. Erityisesti kielimallit voivat tuottaa harhaanjohtavia tai virheellisiä ratkaisuja, ja ne saattavat ajoittain myös hallusinoida eli tuottaa sisältö, joka ei perustu paikkansapitävään tietoon. Tekoälyn tuottama koodi voi vaikuttaa toimivalta, mutta silti sisältää heikkouksia tai poiketa ohjelmistokehityksen vakiintuneista standardeista. Tämän vuoksi vastuu tuotosten arvioinnista ja laadun varmistamisesta säilyy kehittäjällä. Tekoälyn hyödyntäminen voi lisätä työn kognitiivista kuormitusta, sillä kehittäjien tulee sekä muotoilla syötteensä huolellisesti että arvioida jatkuvasti vastausten luotettavuutta. Liiallinen tukeutuminen tekoälyyn voi heikentää kriittistä ajattelua sekä omaa osaamista, mikä puolestaan vaikeuttaa tekoälyn tuottamien ratkaisujen arviointia. Ratkaisujen hyödyntäminen ilman riittävää ymmärrystä heikentää myös koodin selitettävyyttä ja ylläpidettävyyttä. Vaikka tekoäly voi tukea oppimista, siihen liittyy riski oman ajattelun ulkoistamisesta ja osaamisen kehittämisen laiminlyömisestä. Lisäksi organisaatioiden odotuksen tekoälyn tuottavuus- ja laatuvaikutuksista voivat olla

epärealistisia. Tämän vuoksi organisaatioiden rooli korostuu selkeiden ohjeistuksien ja vastuukäytäntöjen luomisessa, jotta tekoälyn käyttö on hallittua ja sen mahdollisuudet ja rajoitukset ymmärretään yhtenäisesti. Tekoäly voi samanaikaisesti parantaa tuottavuutta ja lisätä työn kuormittavuutta sekä epävarmuutta. Kehittäjät toimivat jatkuvassa vuorovaikutuksessa tekoälyn kanssa arvioiden sen tuottamia ratkaisuja, mikä voi lisätä työn henkistä rasitusta. Kasvanut kuormitus ja epävarmuus voivat heikentää kehittäjän tehokkuutta, mikä voi heikentää tekoälystä saatavaa hyötyä. Siinä missä riippuvuus muihin kehittäjiin on laskussa, on riippuvuus tekoälystä ja sen tuottamasta tuesta kasvamassa, mikä muuttaa työn riippuvuusrakenteita. Keskeistä tekoälyn hyödyntämisessä on tasapainon löytäminen tehokkuuden sekä hyvinvoinnin välillä. Tekoälyä tulisi hyödyntää sen vahvuuksia korostaen, mutta samalla sen tuottamiin ratkaisuihin tulee suhtautua kriittisesti.

Tutkimuksen kolmas tutkimuskysymys oli ”Miten tekoäly muuttaa ohjelmistokehittäjän roolia ja osaamisvaatimuksia?”. Aiemmassa tarkastelussa käsiteltiin jo tekoälyn vaikutuksia työnkuvaa ja käytännön työtehtäviin, mutta tämän kysymyksen tavoitteena oli hahmottaa laajemmin kehittäjän roolia tekoälyn kehittyessä. Keskeinen kysymys on, missä määrin tekoäly korvaa ohjelmistokehittäjiä ja millainen merkitys kehittäjillä säilyy tulevaisuudessa. Tekoäly ei nykytilanteessa korvaa ohjelmistokehittäjiä, vaan toimii enemmänkin heidän työparinaan. Se voi hoitaa etenkin rutiininomaisempia ja yksikertaisempia tehtäviä, jolloin kehittäjälle jää enemmän aikaa tekoälyn ohjaamiseen, tuotosten arviointiin ja laadunvarmistamiseen.

Kehittäjän roolissa korostuvatkin yhä enemmän päätöksenteko, eettinen vastuu sekä kyky toimia yhteistyössä tekoälyn kanssa. Kehittäjän rooli on siirtymässä hiljalleen perinteisestä teknisestä toteuttajasta, tekoälyn koordinoijaksi. Samalla vastuu tietoturvasta, laadusta sekä ohjelmistoarkkitehtuurista korostuu. Tekoälyn nopea kehitys edellyttää jatkuvaa oppimista, ja roolin liittyvät haasteet kytkeytyvät yhä enemmän osaamisen ylläpitämiseen, henkiseen kuormitukseen ja epävarmuuden sietämiseen. AI-lukutaito on muodostumassa keskeiseksi osaamisvaatimukseksi. Vaikka tekoäly ei korvaa etenkään kokeneiden kehittäjien asiantuntemusta, se muuttaa merkittävästi heidän rooliansa ja työnkuvaa. Roolissa vaaditaan yhä enemmän teknisen toteutuksen ulkopuolista osaamista, kuten osallistumista päätöksentekoon ja kokonaisuuksien hallintaan.

Tämän tutkielman tavoitteena oli muodostaa kokonaisvaltainen ymmärrys tekoälyn merkityksestä ohjelmistokehittäjän rooliin tarkastelemalla sen käyttötarkoituksia sekä siihen liittyviä hyötyjä ja haittoja. Tarkoituksena oli jäsentää tekoälyn ja ohjelmistokehittäjien välistä nykyistä suhdetta mahdollisimman kattavasti, mikä vuoksi ilmiötä tarkasteltiin laaja-alaisesti eri näkökulmista.

Jatkossa tekoälyn kehittyessä ja integroituessa yhä syvällisemmin ohjelmistokehittäjien työhön avautuu tarve entistä syvemmälle tutkimukselle. Erityisesti kiinnostava jatkotutkimuksen kohde on tekoälyn vaikutus kehittäjien osaamiseen pitkällä aikavälillä. Tässä tutkielmassa aihetta sivuttiin aiemman tieteellisen kirjallisuuden pohjalta, mutta empiirisempi ja syvällisempi tarkastelu voisi tuottaa kriittistä lisätietoa siitä, miten tekoälyn lisääntyvä käyttö muokkaa kehittäjien osaamista ja asiantuntijuutta käytännössä.

Lähteet

Bartsch, S. C., Schmidt, J.-H., Adam, M., & Benlian, A. (2026). Increasing developers' code accountability perceptions in open source software development. International Journal of Information Management, 86, 102974. https://doi.org/10.1016/j.ijinfomgt.2025.102974

Chuang, Y.-T., Chiang, H.-L., & Lin, A.-P. (2025). Insights from the Job Demands–Resources Model: AI's dual impact on employees' work and life well-being. International Journal of Information Management, 83, 102887. https://doi.org/10.1016/j.ijinfomgt.2025.102887

Cousineau, C., Dara, R., & Chowdhury, A. (2025). Trustworthy AI: AI developers' lens to implementation challenges and opportunities. Data and Information Management, 9(2), 100082. https://doi.org/10.1016/j.dim.2024.100082

De, E., Le, H., Meduri, K., Nadella, G., & Gonaygunta, H. (2025). Redefining the Programmer: Human-AI Collaboration, LLMs, and Security in Modern Software Engineering. Computers, Materials & Continua, 85(2), 3569–3582. https://doi.org/10.32604/cmc.2025.068137

Ergün, A., Baer, M., & Plattfaut, R. (2026). Leading GenAI-Equipped teams: Recalibrating leadership in the age of generative artificial intelligence. Journal of Information Technology, 02683962261428154. https://doi.org/10.1177/02683962261428154

Foroudi, P., Marvi, R., & Zha, D. (2025). AI sensation and engagement: Unpacking the sensory experience in human-AI interaction. International Journal of Information Management, 84, 102918. https://doi.org/10.1016/j.ijinfomgt.2025.102918

Lisana, L., & Susanto, A. S. (2026). Exploring Students' Intentions to Reuse Chat AI LLMs in Learning: An S-O-R Framework Approach in Indonesia. Journal of Information Technology Education: Research, 25, 04.

Lucas, S., Heinitz, R. M., Becker, S. J., & Charton, J. E. (2025). Developing a framework for addressing ethi-cal challenges in generative AI. Journal of Information Technology Case and Application Research, 0(0), 1–15. https://doi.org/10.1080/15228053.2025.2558443

Nguyen, M., Agarwal, R., Malik, A., & Pandey, R. (2025). Generative AI: Elevating knowledge acquisition and retention and recall through autonomy, interactivity and engagement. Journal of Enterprise Information Management, 38(6), 1692–1719. https://doi.org/10.1108/JEIM-08-2024-0433

Nguyen, M., Zhang, Y., Bu, Y., & Belk, R. (2026). Generative AI in academic research activities: The hidden side of self-detrimental consumption. International Journal of Information Management, 87, 103024. https://doi.org/10.1016/j.ijinfomgt.2025.103024

Qiu, K., Puccinelli, N., Ciniselli, M., & Di Grazia, L. (2025). From Today's Code to Tomorrow's Symphony: The AI Transformation of Developer's Routine by 2030. ACM Trans. Softw. Eng. Methodol., 34(5), 121:1-121:17. https://doi.org/10.1145/3709353

Taneski, V., Karakatič, S., Rek, P., & Jošt, G. (2025). Impact of Developer Queries on the Effectiveness of Conversational Large Language Models in Programming. Applied Sciences, 15(12). https://doi.org/10.3390/app15126836

Vasiliniuc, M.-S., & Groza, A. (2023). Case study: Using AI-assisted code generation in mobile teams. 2023 IEEE 19th International Conference on Intelligent Computer Communication and Processing (ICCP), 339–346. https://doi.org/10.1109/ICCP60212.2023.10398656

Wang, X., Zhong, W., Huang, K., & Liang, B. (2026). High interest but low adoption: Navigating organizations' journey towards generative artificial intelligence implementation. International Journal of Information Management, 87, 103009. https://doi.org/10.1016/j.ijinfomgt.2025.103009

Xiao, T., Treude, C., Hata, H., & Matsumoto, K. (2024). DevGPT: Studying Developer-ChatGPT Conversations. Proceedings of the 21st International Conference on Mining Software Repositories, MSR '24, 227–230. https://doi.org/10.1145/3643991.3648400

Yang, H., Zeng, Y., Xing, H., & Hu, P. (2026). Fatigued by uncertainties: Exploring the cognitive and emotional costs of generative AI usage. International Journal of Information Management, 87, 103010. https://doi.org/10.1016/j.ijinfomgt.2025.103010

Liitteet

Selvitys tekoälyn käytöstä

Tutkielmassa hyödynnettiin OpenAI:n ChatGPT:tä kielenhuollon tukena. Tekoälyä käytettiin tekstin sujuvuuden, selkeyden ja akateemisen ilmaisun parantamiseen sekä kirjoitusvirheiden tunnistamiseen. Kaikki tekoälyn tuottamat ehdotukset tarkastettiin, ja varmistettiin, että tekstin muokkaamien ei muuttanut sen merkitystä.

Käytettyjä kehotteita olivat esimerkiksi:

- ”Voitko antaa ehdotuksia, miten parantaisin tämän lauseen sujuvuutta?”
- ”Voitko tarkistaa tämän kappaleen ja kertoa, jos siinä on kirjoitusvirheitä tai epäselviä ilmaisuja?”
- ”Miten tämän lauseen voisi muotoilla akateemisemmin?”