OULUN AMMATTIKORKEAKOULU

Kristian Ängeslevä

TEKOÄLY JA SEN INTEGROIMINEN OHJELMISTOKEHITYKSEN ELINKAAREEN NIIN RUOHONJUURITASOLLA KUIN SITÄ YMPÄRÖIVISSÄ TEHTÄVISSÄ

TEKOÄLY JA SEN INTEGROIMINEN OHJELMISTOKEHITYKSEN ELINKAAREEN NIIN RUOHONJUURITASOLLA KUIN SITÄ YMPÄRÖIVISSÄ TEHTÄVISSÄ

Kristian Ängeslevä
Opinnäytetyö
Kevät 2024
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehityksen suuntautumisvaihtoehto

---

Tekijä: Kristian Ängeslevä
Opinnäytetyön nimi: Tekoäly ja sen integroiminen ohjelmistokehityksen elinkaareen niin ruohonjuuri tasolla kuin sitä ympäröivissä tehtävissä
Työn ohjaaja: Raili Simanainen
Työn valmistumislukukausi ja -vuosi: Kevät 2024 Sivumäärä: 27 + 1 liite

---

Tämä opinnäytetyö käsittelee tekoälyn integrointia ohjelmistokehityksen elinkaareen. Tavoitteena oli tutkia, miten tekoäly voi tehostaa ja optimoida ohjelmistokehityksen vaiheita, kuten vaatimusanalyysiä, suunnittelua, koodausta ja ylläpitoa.

Tekoälyllä on potentiaalia automatisoida manuaalisia ja toistuvia tehtäviä, parantaa koodin laatua ja nopeuttaa kehityssyklejä. Se voisi myös optimoida projektinhallintaa, tehostaa resurssien käyttöä ja tunnistaa riskejä niiden realisoitumista.

Sovellusesimerkeistä nähdään, että tekoäly parantaa ohjelmistokehityksen tehokkuutta ja laatua, vähentää virheitä ja nopeuttaa prosessia. Keskeisiä tuloksia olivat tekoälyn kyky tukea vaatimusanalyysia ja suunnittelua, automatisoida koodaus- ja testausvaiheita sekä parantaa projektinhallintaa ja tiimien kommunikaatiota. Tekoälypohjaiset työkalut, kuten älykkäät koodieditorit ja ennakoivat testausjärjestelmät, osoittautuivat hyödyllisiksi.

Johtopäätöksenä todettiin, että tekoälyn integrointi tarjoaa merkittäviä etuja, mutta vaatii huolellista suunnittelua ja mallien päivittämistä käyttäjäpalautteen perusteella. Suosituksena on jatkaa ja laajentaa tekoälyn hyödyntämistä uusien sovellusten ja menetelmien kehittämiseksi.

Asiasanat: Tekoäly, ohjelmistokehitys, optimointi, tuki

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Option of Software Development

Author: Kristian Ängeslevä
Title of thesis: Artificial intelligence and its integration into the software development life cycle both at the grassroots level and in the tasks surrounding it
Supervisor(s): Raili Simanainen
Term and year when the thesis was submitted: Spring 2024
Number of pages: 27 + 1

This thesis examines the integration of artificial intelligence into the software development lifecycle. The aim was to investigate how AI can enhance and optimize various stages of software development, such as requirement analysis, design, coding, testing and maintenance.

AI has the potential to automate manual and repetitive tasks, improve code quality, and accelerate development cycles. It can also optimize project management, enhance resource utilization, and identify risks before they materialize.

Software examples demonstrated that AI improves the efficiency and quality of software development, reduces errors, and speeds up the process. Key findings include AI's ability to support requirements analysis and design, automate coding and testing phases, and improve project management and team communication. AI-based tools, such as intelligent code editors and predictive testing systems, proved to be particularly beneficial.

The conclusion is that AI integration offers significant advantages but requires careful planning and continuous model updates based on user feedback. It is recommended to continue and expand the use of AI to develop new applications and methods.

Keywords: Artificial intelligence, software development, optimization, support

SISÄLLYS

TERMIT JA LYHENTEET

AI (Artificial Intelligence)	Tekoäly eli tietokonejärjestelmien tai ohjelmien kyky suorittaa tehtäviä, jotka vaativat ihmisen älykkyyttä, kuten oppimista ja ongelmanratkaisua
Koneoppiminen	Koneoppiminen on tekoälyn osa-alue, jossa tietokoneet oppivat ja parantavat suorituskykyään kokemuksen perusteella ilman erillistä ohjelmointia.
NLP (Natural Language Network)	Luonnollisen kielenkäsittely, tekoälyn osa-alue, joka keskittyy ihmisten kielen ja tietokoneiden väliseen vuorovaikutukseen.
GAN (Generative Adversarial Network)	Generatiivinen kilpaileva verkko, koneoppimismalli, jossa kaksi hermoverkkoa kilpailee keskenään tuottaakseen realistisia dataesityksiä.
Refaktorointi	Koodin rakenteen parantaminen ilman sen toiminnallisuuden muuttamista, mikä parantaa koodin luettavuutta ja ylläpidettävyyttä.
Automaatio	Automaatio on prosessi, jossa käytetään teknologiaa tehtävien suorittamiseen ilman ihmisen väliintuloa, parantaen tehokkuutta ja vähentäen virheitä.
Projektienhallinta	Projektien suunnittelu, organisointi, valvonta ja hallinta tavoitteiden saavuttamiseksi aikataulussa ja budjetissa.

Testaus	Ohjelmiston toiminnallisuuden ja laadun arviointi, jossa pyritään löytämään virheitä ja varmistamaan, että ohjelmisto toimii odotetusti.
Tietokanta	Järjestelmä, joka tallentaa, hallinnoi ja hakee dataa tehokkaasti, mahdollistaen suuret tietomäärät ja monimutkaiset kyselyt.
Isolation Forest	Eristysmetsät on koneoppimismenetelmä, jota käytetään poikkeamien tunnistamiseen suurten tietomäärien analysoinnissa.

1 JOHDANTO

Tekoäly on muuttanut tapaamme lähestyä teknologiaa ja erityisesti ohjelmistokehityksen alalla sen vaikutus on ollut ja tulee olemaan mullistavaa. Varsinkin tieto- ja viestintätekniikan alalla tekoäly ja se, miten sitä voidaan hyödyntää ohjelmistokehityksessä ja sitä ympäröivissä työelämän tehtävissä parantamaan projektinhallinta ja tiimien välistä kommunikaatiota tulee olemaan avainasemassa seuraavan kymmenen vuoden aikana. Tavoitteena on myös arvioida tekoälyn roolia laadunvarmistuksessa ja virheidenkorjauksessa sekä tutustua käytännön esimerkkeihin onnistuneista tekoälyintegraatioista ohjelmistoprojekteissa.

Ohjelmointi on nopeasti kehittyvä ala, jossa uusien teknologioiden omaksuminen on välttämätöntä. Tekoäly tarjoaa mahdollisuuksia automatisoida monotonisia ja toistuvia tehtäviä, parantaa koodin laatua ja nopeuttaa kehityssyklejä. Tämän lisäksi tekoäly voi auttaa optimoimaan projektinhallintaa tehostaen resurssien käyttöä ja tunnistaen potentiaalisia riskejä ennen niiden realisoitumista.

Ohjelmistokehityksen puolelta tarkastelen tekoälyn soveltamista eri osa-alueilla, kuten vaatimusanalyysissä, suunnittelussa, koodauksessa, testaustauksessa ja ylläpidossa. Käyn läpi, miten tekoäly voi tukea ja parantaa tiimityöskentelyä, varsinkin ketterissä kehitysmenetelmissä, joissa nopeiden syklien ja jatkuvan palautteen anto ja muokkaus on arkipäivää.

Lisäksi esittelen sovellusesimerkkejä, jotka havainnollistavat tekoälyn käyttöä käytännön ohjelmistoprojekteissa ja osoittavat mitä hyötyjä sitä käyttämällä on saavutettu. Nämä esimerkit tarjoavat konkreettisen näkemyksen ja oppimiskokemuksen siitä, miten tekoäly voi vaikuttaa ohjelmistokehitykseen.

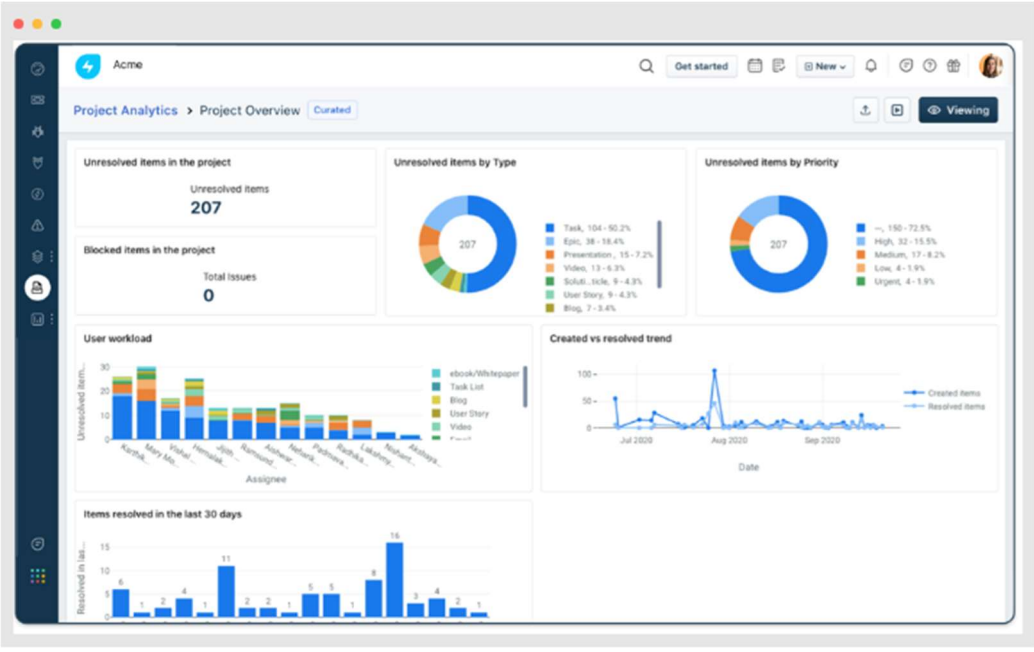
Tämä työ on suunnattu tulevaisuuden ohjelmistokehittäjille ja testaajille, jotka haluavat ymmärtää tekoälyn potentiaalin ja integroida sitä omaan ammatilliseen osaamiseensa. Tavoitteena on tarjota ymmärrystä siitä, miten voimme parhaiten hyödyntää tekoälyä tehokkaamman ja laadukkaamman ohjelmistotuotannon saavuttamiseksi.

2 PROJEKTIHALLINAN JA TIIMIN OPTIMOINTI TEKOÄLYN AVULLA

Tekoälyn integrointi projektinhallintaan on tuonut ennennäkemättömiä mahdollisuuksia parantaa tehokkuutta, päätöksentekoa ja resurssien hallintaa. Tässä luvussa käsitellään, miten tekoäly voi muuttaa ja parantaa projektienhallinnan ja tiimin kommunikoinnin käytäntöjä. Tämä saavutetaan tukemalla niin projektipäälliköitä kuin tiimejäkin vapauttamaan aikaa luovempiin ja strategisimpiin tehtäviin. samalla parantaen projektien tuloksellisuutta inhimillisten virheiden takia. Käytännölliset esimerkit ja teoria tulevat tuomaan näkökulmaa tekoälyn soveltamisesta käytäntöön.

2.1 Aikataulutus ja resurssien hallinta

Tekoälyn hyödyntäminen aikataulutuksessa ja resurssien hallinnassa voi merkittävästi tehostaa projektinhallintaa. Tekoälypohjaiset työkalut kykenevät analysoimaan suuria datamääriä nopeasti niin ohjelmoinnin vaatimuksia kerätessä kuin projektin hallintaan liittyvissä askelissa, mikä tuo tarkempia ennusteita ja suosituksia resurssien allokoinnista ja aikataulutuksesta firman antamilla parametreilla alla olevan kuvan kaltaisesti (kuva 1). Tekoälyn avulla voidaan ennustaa mahdollisia ongelmakohtia projektin eri vaiheissa ja tekoäly voi ehdottaa toimenpiteitä niiden välttämiseksi tai lieventämiseksi (1). Tämä ei ainoastaan nopeuta projektien toimitusaikoja vaan myös vähentää ylityksiä budjetissa ja aikataulussa.



KUVA 1. Freshservice tekoälyä käyttävän projektimanagerointiohjelman näkymä operaatioiden etenemisestä ja tulevaisuuden arvioista (2).

2.2 Riskienhallinta ja ennustaminen

Tekoälyllä suurin potentiaali on riskienhallinnassa. Käyttämällä koneoppimista ja ennustavaa analytiikkaa firman antamilla parametreilla ja resurssitiedoilla voidaan tunnistaa riskejä ja loogisuuksia, jotka saattaisivat jäädä ihmissilmältä huomaamatta. Tämä analyysi perustuu historiallisen datan käsittelyyn ja sen vertaamiseen firman ohjelmistoprojektin valmistumiseen antaen näkemyksen ennusteineen tulevista riskeistä ja niiden realisoitumisista (1). Tämän ihmistyöhön verrattain laajemman lähestymistavan ansiosta voi projektipäällikkö välttää mahdollisia kompastuskiviä ja minimoida riskit projektin epäonnistumiseen.

Siinä missä projektinhallinnassa tekoäly voi tukea projektipäälliköitä isojen liikkeiden ja tietomäärien analysoinnissa voi se myös tukea liikkeitä koko projektin kannalta. Tekoälyä voidaan käyttää parantamaan myös koko tiimin kommunikointia ja yhteistyötä parempien tulosten tuottamiseksi yksittäisen työntekijän tasolla kuin myös keskinäisten toimivien työelimien tasolla (3).

3 OHJELMISTOKEHITYSTÄ YMPÄRÖIVIEN TEHTÄVIEN AUTOMATISOINTI TEKOÄLYN AVULLA

Tekoälyn integroiminen ohjelmistokehityksen eri vaiheisiin on avannut uusia mahdollisuuksia niin automaation, tehokkuuden kuin prosessien optimoinninkin kannalta. Tekoälyn rooli ohjelmistokehityksessä ulottuu kauas perinteisten automaatiotyökalujen rajoista. Se tarjoaa kehittyneitä ratkaisuja, jotka voivat merkittävästi nopeuttaa ja parantaa sekä ohjelmiston suunnittelua että toteutusta. Tässä luvussa tarkastellaan, miten tekoälyä hyödynnetään niin vaatimusanalyysissä, suunnittelussa, koodauksessa, testauksessa kuin ylläpidossakin. Tämä tekoälyn integroiminen ohjelmoijan tuotteen elinkaareen mahdollistaa monimutkaisten ohjelmistojen kehittämisen suuremmalla tehokkuudella ja vähäisemmillä virheillä verrattaessa perinteiseen kehityskaareen.

3.1 Vaatimusanalyysi ja suunnittelu

Vaatimusanalyysi ja suunnitteluvaihe ovat kriittisiä osia ohjelmistokehitysprosessia, ja tekoälyn integrointi näihin vaiheisiin tarjoaa merkittäviä etuja. Tekoäly voi auttaa tunnistamaan, priorisoimaan ja dokumentoimaan vaatimukset tehokkaammin, mikä varmistaa, että lopputuote vastaa tarkasti asiakkaiden ja sidosryhmien odotuksia.

Oli kyseessä sitten asiakkaan tai ostajakunnan vaatimusten kerääminen tai analysointi, tekoälyn käytön kynnys laskee, mitä isompi määrä tietoa halutaan kerätä ja analysoida. Siinä missä pienen firman tai asiakkaan toiveita ja vaatimuksia on helpompi kuunnella kokousten tai kasvokkain käytyjen keskustelujen äärellä, on myös järkevämpää toteuttavan elimen resurssien kannalta, jos vaatimuksia tulee useilta sadoilta eri henkilöiltä tai parhaassa tapauksessa jopa miljoonilta esimerkiksi pelien kehityksen kannalta. Tekoäly voi toteuttaa geneeriset kaavakkeiden täytöt interaktiivisemmalla keskustelubottiperiaatteella, jolle on annettu parametrit firman työkaluista, resursseista ja näihin liittyvistä aikatauluista vaatimuksien määrän kasvaessa.

Siinä missä aktiivinen vuorovaikutus asiakkaan ja toteuttavan elimen välillä on ollut ennen haastavaa resurssien vähyyden ja katkonaisen yhteydenpidon vuoksi, tekoälyn lisääminen tähän tiedon keräämisen hierarkiaan helpottaa mahdollisien suurten tietomäärien käsittelyä koneoppimisen ja luonnollisen kielen käsittelyn (NLP) ansioista. On se sitten tekoälytyökalujen käyttämistä

informaation etsintään keskusteluboteilla, sähköposteilla tai eri kyselyillä niin tekoäly voi yhdistää nämä kaikki tietoväylät yhteen tiiviiseen formaattiin (4), jota ruohonjuuritason työskentelijät voivat sitten käyttää itse tuotteen rakentamiseen. Tässä suuressa tiedon virrassa esiintyvät inhimilliset virheet niin vaatimusten antajien ja prosessoijien välillä vähenee, kun voidaan esittää tarkentavia kysymyksiä tekoälyn puolelta, kun törmätään ristiriitoihin joko aiempien keskustelujen kanssa tai useampien eri ihmisten esittämien toiveiden välillä.

3.2 Dokumentointi, jatkuvuus ja skaalautuvuus

Mitä enemmän asiakkaita tai käyttäjiä tuotteella tulee olemaan, sitä haastavammaksi käsiteltävän tiedon keräys ohjelmistovaatimusten keräyksestä saattaa yritykselle tulla. Dokumentaatio tekoälyn avulla helpottaa tätä tiedon määrän tiivistämistä ja turhan sekä toistuvan tiedot poistamista tarpeen tullen luoden yksityiskohtaisia ja standardin mukaisia vaatimusdokumentteja. Jatkoa ajatellen tekoäly voi myös päivittää muutokset reaaliajassa ja pitää sidosryhmät ajan tasalla näistä muutoksista (4).

Tuotettavan ohjelmiston jatkoa ajatellen tekoälyä voidaan käyttää projektin laajentamiseen, kun vaatimuksista voidaan rakentaa jo kerätyn tiedon päälle lisäominaisuuksia niin kauan kuin tuottavan elimen resurssit ja parametrit ovat tekoälyn tiedossa (4) alla olevan kuvan tapaisesti (kuva 2).

KUVA 2. Jäljitettävyyspuunäkymä Visuresolutions-yhtiön tarjoaman työkalun generoimasta vaikutusanalyysistä, joka näyttää vaatimukset ja tietokannassa jäljitettävät kohteet (5).

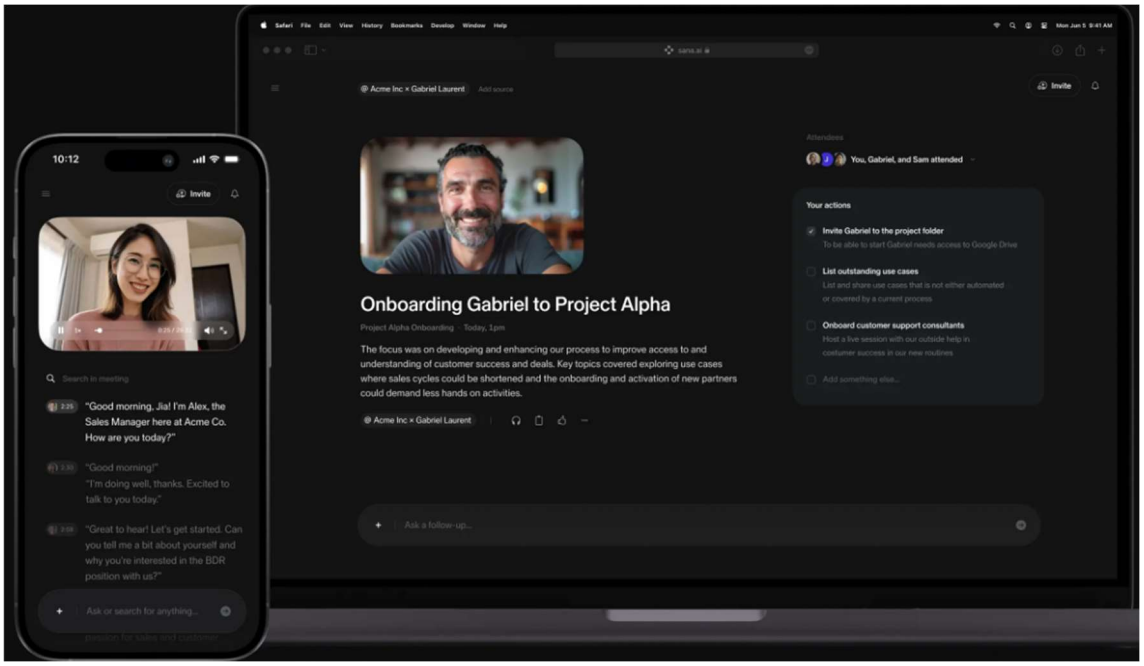
Näiden tekoälyn tarjoamien etujen avulla ohjelmistokehittäjät voivat keskittyä enemmän innovointiin ja luovuuteen, kun taas enemmän aikaa vieviä ja virhealttiita tehtäviä voidaan automatisoida tekoälyn avulla. Mitä isompiin projekteihin mennään, sitä suurempi tekoälyn säästämä aika on, kun poistetaan osaksi ihmisanalyysin tuodut rajoitteet ja riskitekijät.

3.3 Kommunikaation automatisointi ja tehostaminen

Tekoälyn käyttö voi selkeyttää jokaisen tiimin jäsenen roolia ja vastuuta sekä seurata edistymistä, mikä taas auttaa tiimejä pysymään ajan tasalla tehtävien suhteen (6). Mitä alemmas ohjelmointifirman hierarkiassa mennään, sitä enemmän huomataan, kuinka rutiinipohjaisten tehtävien määrä kasvaa esimerkiksi tilannetta päivittävien sähköpostien tai tiimipalaverien muodossa. Tekoäly voi auttaa tässä seuraamalla päivittäistä edistymistä ja luoden tiivistelmää tai raporttia joko työntekijän sanelemana tai tukena kirjoittamisessa alla olevan kuvan kaltaisesti (kuva 3). Tämä kevennys helpottaa ruohonjuuri tason työntekijän keskittymistä itse pääasiaan, mikä tietotekniikan firmoissa on usein se ohjelmointi osio, mikä täten tukee tiimien pysymistä aikataulussa (3).

Mitä tulee kommunikaatioon tiimiläisten välillä yksittäisten tiimiläisten välillä, voi tekoäly auttaa tuomaan esille malleja ja trendejä (3), joita käydään toistuvasti tiimin sisäisesti, on se sitten yksittäisten henkilöiden välillä tai ryhmissä. Näiden toistuvien trendien tai perinteiden tuomassa rutiinissa voi syntyä virheitä, joita ei nähdä, kun siirrytään projektista toiseen ja asiakkaan vaatimukset muuttuvat. Kun tekoälylle annetaan pääsy projektin sisäisiin tavoitteisiin ja parametreihin, voi se ehdottaa tai tuoda esiin epäkohtia, joita se huomaa ja ohjaa täten yksittäisten tai pienten tiimien toimintaa kohti tehokkaampaa työskentelyä.

Yksittäisen työntekijän kautta tekoäly voi auttaa koko tiimiin tehokkuutta ja kehittymistä. Tekoälyä voidaan käyttää esimerkiksi komppaustyökaluna (3) luomaan itsevarmuutta ja vahvempaa pohjaa ideoihin, joita työntekijöillä on, kun ajatusta esimerkiksi tehokkaammasta koodiin liittyvästä muutoksesta ei uskalleta tuoda esiin, jos sitä ei osata esimerkiksi rakentaa muotoon, jossa se uskallettaisiin esitellä firman hierarkiassa korkeammille henkilöille. Ilman tätä tekoälyn komppauksen tuomaa itsevarmuutta ohjelmointi projektin tehokkuus voi jäädä vajaaksi sen täydestä potentiaalista.



KUVA 1. Esimerkki paremmasta kokousalustasta (sana.ai), joka tekee muistiinpanoja samanaikaisesti äänitunnistuksella (7).

3.4 Yhteistyö ja ryhmädynamiikan parantaminen

Tekoälyllä on merkittävä rooli tavoitteiden ja odotettujen tulosten analysoinnissa, tiimin keskittymisessä strategiseen ajatteluun sekä luovassa ongelmanratkaisussa. Tämä muuttaa työdynamiikkaa antaen tiimiläisille aikaa omistautua tyydyttävämpiin ja haastavampiin tehtäviin, jolloin sekä aikaa että henkisiä voimavaroja käytetään tehokkaammin (3). Linkki yhteistyöhön ja itse ryhmän sisäiseen dynamiikkaan tulee ironisesti osittain yhteistyön vähenemisestä, kun tekoälyä käytetään eri ongelmakohtien ratkaisemiseen firman sisäistä tietokantaa käyttäen ja täten säästäen kommunikoinnin tarvetta (3). Kun törmätään ongelmaan, johon ei löydy tekoälypohjaisten työkalujen kautta ratkaisua, voidaan luoda yksityiskohtainen kysely tiimin muille jäsenille. Tällöin saadaan mahdollinen tästä löytyvä ratkaisu tallennettua tulevaisuutta ajatellen, kun tekoälyä käytetään vastaavanlaisissa tilanteissa. Edellisen hypoteettisen tilanteen pohjalta johtohahmot voivat myös laajentaa työntekijöidensä tehokkuutta esimerkiksi antaen heille mahdollisuuksia kasvattaa tietotaitojaan eri alaan liittyvien kurssien muodossa tai lisäten mahdollisia tukevia työkaluja.

Siinä missä tekoälyyn perustuvat työkalut voivat vähentää kommunikoinnin tarvetta, voi se myös helpottaa ja kasvattaa kommunikoinnin määrää ehdottamalla toistuvien ongelmien puimista kokouksissa aiheeseen liittyvien työntekijöiden kanssa, mikä mahdollisesti johtaa innovatiivisiin ratkaisuihin (3).

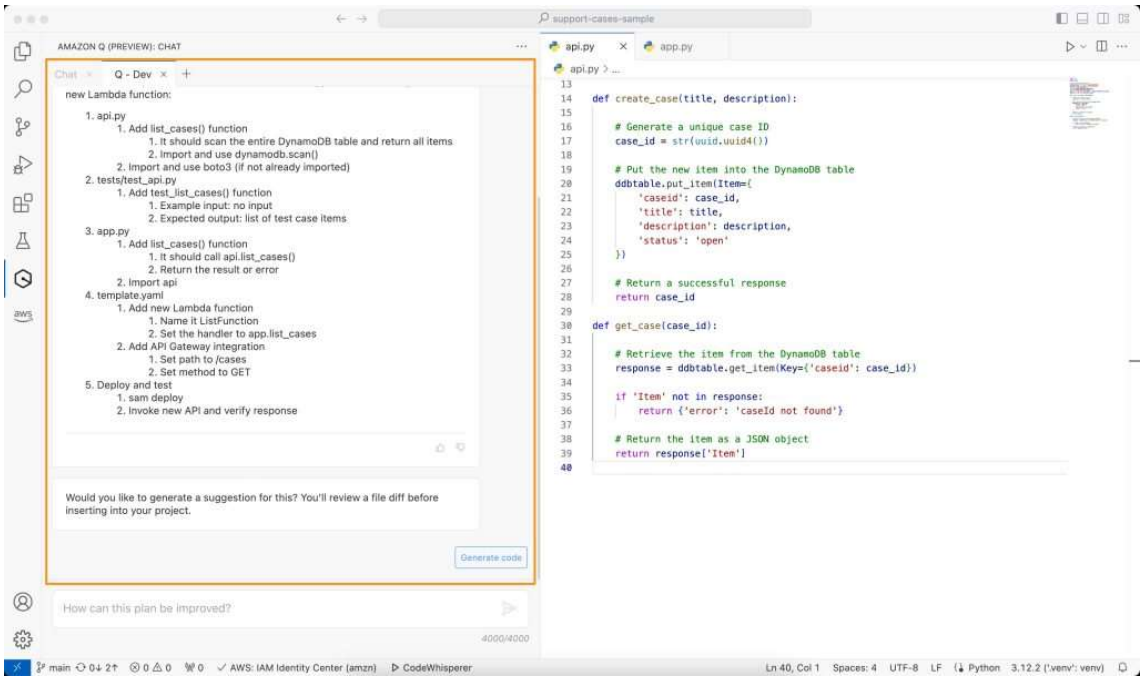
4 KOODAUS, TESTAUS JA YLLÄPITO TEKOÄLYN TUKEMANA

Tekoälyn rooli ohjelmistokehityksen eri vaiheissa on kasvanut valtavasti viimeisen vuosikymmenen aikana, mutta erityisesti sen merkitys on näkynyt koodauksessa, testauksessa, laadunvarmistuksessa ja virheiden korjauksessa. Automatisointi, jota käytetään koodausta ympäröivien tehtävien helpottamisessa, voidaan myös toteuttaa monissa suurta manuaalista työtä tarvitsevissa ohjelmointi prosessien automatisoinneissa parantaen ohjelmistojen laatua nopeuttaen tuotekehityksen syklejä. Seuraavissa luvuissa käsitellään tekoälyn soveltuvuutta näihin eri vaiheisiin.

4.1 Koodaus ja testaus

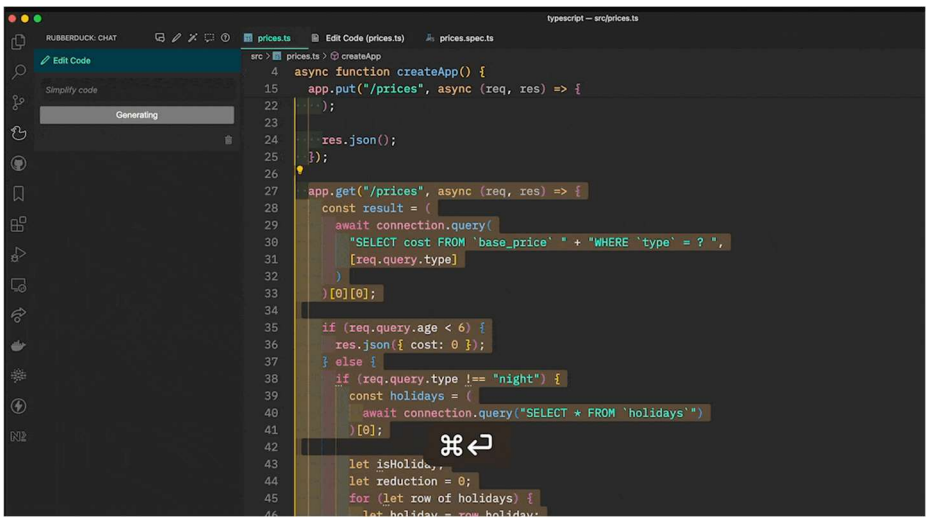
Tekoälyn integrointi ohjelmistokehitykseen on tuonut merkittäviä muutoksia koodausprosessiin viimeisen 10 vuoden aikana. Tekoälyn avulla ohjelmoijat ovat voineet automatisoida monia manuaalisia ja toistuvia tehtäviä, mikä taas on parantanut kehitysprosessien tehokkuutta ja vapauttanut aikaa muihin tärkeisiin ohjelmistokehityksen puoliin kuten luovempiin ongelmanratkaisuihin. Esimerkkinä vaikuttimesta tehokkuuden kasvamiseen voidaan pitää työkaluja, jotka analysoivat olemassa olevia koodipohjia ja ennustavat niiden pohjalta seuraavien koodirivien toiminnallisuutta kyseisen projektin parametreilla. Tämä lisää koodin yhdenmukaisuutta ja helpottaa muiden projektin osallisten koodin ymmärrystä (8).

Tekoälypohjaiset koodausapuvälineet, kuten älykkäät koodieditorit ja koodigeneraattorit (kuva 4), voivat merkittävästi vähentää kehittäjien työtaakkaa. Nämä työkalut käyttävät koneoppimista tunnistaakseen koodipohjista toistuvia rakenteita ja mallinnuksia, mikä mahdollistaa automaattisten koodinkorjausehdotusten tarjoamisen. Tämä vähentää ohjelmoijien tekemiä virheitä ja parantaa yhdenmukaisuutta koodissa.



KUVA 2. Esimerkki Amazon Q developer-ohjelman koodin generoinnista pseudokoodin avulla (9).

Refaktorointi ja koodin yleinen optimointi on yksi monista paikoista, missä tekoälyä voidaan käyttää tehostamaan koodin laatua. Tekoäly voi analysoida valmista koodia ja ehdottaa parannuksia, jotka parantavat koodin luettavuutta joko korjauksien tai kommenttien avulla ja suorituskykyä ilman, että itse toiminnallisuus muuttuu. Tämän koodin analysointi tai vanhan koodin refaktorointi (kuva 5) varsinkin isommissa ohjelmistoprojekteissa helpottuu, koska suurin osa siitä voidaan tehdä nopeammin tekoälyn avulla pohjautuen firman tietokantaan (10) ja sieltä löydettäviin käytänteisiin.



KUVA 3. Esimerkki Rubberduck työkalusta, jota voidaan käyttää refaktoroinnissa (10).

Testauksessa tekoälyn rooli on koodauksen rinnalla mullistava hyppy tehokkuudessa ja nopeudessa. Tekoälypohjaiset testausjärjestelmät voivat simuloida käyttäjäinteraktioita ja erilaisia käyttöolosuhteita, mikä mahdollistaa laajemman ja perusteellisemman testauksen lyhyemmässä ajassa verrattaessa manuaaliseen testaukseen (6). Tekoälyn avulla voidaan myös kehittää ennakoivia testausjärjestelmiä, jotka pystyvät havaitsemaan ja jopa korjaamaan virheitä ennen kuin ne vaikuttavat loppukäyttäjiin.

Automatisoidut testausjärjestelmät käyttävät koneoppimista ja data-analytiikkaa tunnistaakseen ohjelmiston toiminnalliset virheet ja suorituskykyongelmat reaaliajassa. Tämä jatkuva oppiminen ja mukautuminen mahdollistavat sen, että testausjärjestelmät pysyvät relevantteina ja tehokkaina, vaikka itse ohjelmisto kehittyisi ja muuttuisi ajan myötä (6). Näin ohjelmistotiimit voivat keskittyä strategisempiin ja luovempiin tehtäviin, kun tekoäly hoitaa rutiininomaiset ja toistuvat testaukset.

Tekoäly voi myös parantaa testauksen kattavuutta ja syvyyttä. Perinteiset manuaalitestauksen ja automaatiotestauksen menetelmät saattavat jättää huomaamatta tiettyjä virheitä tai suorituskykyongelmia, mutta tekoäly pystyy analysoimaan suuria datamääriä ja testaustuloksia nopeammin tuoden kattavampaa virheseulontaa. Tämä taas parantaa ohjelmiston kokonaislaatua ja luotettavuutta ohjelmiston kestävyyden kannalta. Koodausta seuraavat työkalut tuovat esille myös sillan testauksen puolelle. Kun tekoäly tietää mihin koodin osaan on koskettu, voi se karsia pois ja priorisoida tiettyjä testitapauksia, kun manuaali- tai ihmisajoisissa testeissä käydään läpi kaikki vanhat testit tietämättä syvemmästä ja paremmasta priorisointijärjestyksestä. Tämä priorisointi tuo testaustiimille ja ohjelmoijille nopeampaa palautetta muutoksille herkistä alueista ja nopeuttaa testausprosessia sekä vähentää testauksen tuomia kustannuksia (6).

4.2 Ylläpito ja virheiden korjaus

Ylläpito on viimeinen taival ohjelmistokehityksessä ja tekoälyn hyödyntäminen siinä sekä virheiden korjauksessa on helpottanut ohjelmistokehittäjien takkaa vapauttaen heiltä aikaa muihin projekteihin. Kuten itse koodauksessakin, myös ylläpidossa voidaan käyttää koneoppimismalleja, kuten syväoppimista ja generatiivisia kilpailevia verkkoja (GAN), tunnistamaan ja korjaamaan ohjelmistovirheitä tai trendejä ohjelmankaatumisien ja bugien varalta. Näiden mallien avulla tekoäly pystyy

analysoimaan suuria määriä koodia ja asiakasdataa lennosta, mihin ihmisten resurssit eivät välttämättä riittäisi manuaalipohjaisesti. Tekoäly muuttuu myös paremmaksi ajan mittaan, kun se oppii tunnistamaan virheitä tai trendejä kuten BugLab, joka käyttää juuri GAN-malleja hyväkseen kouluttaen tekoälyä tunnistamaan tietynlaisia bugeja riippuen siitä, kenelle palvelua ja työkaluja tarjotaan (11).

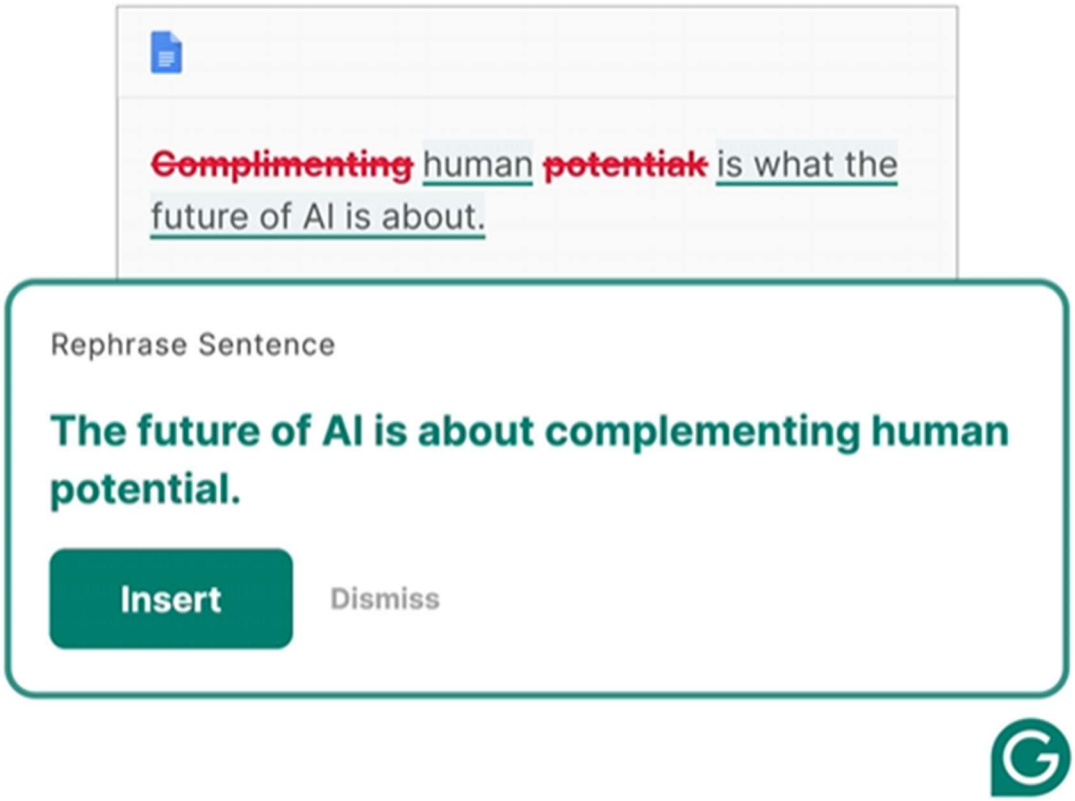
Anomaliatunnistus on yksi kasvavia keskeisiä tekniikoita ohjelmiston ylläpidossa, ja tekoäly on tehnyt isoa edistystä tällä alueella. Koneoppimisen avulla voidaan tunnistaa poikkeamat normaalista toiminnasta, jotka voivat viitata virheisiin tai muihin ongelmiin ohjelmistoissa (12). Tässä tekniikoita, kuten eristysmetsiä (isolation forests), käytetään erottamaan poikkeamat valtavasta määrästä dataa kevyellä muistin tarpeella antaen nopean reaktioajan tunnistamaan ongelmat ennen kuin asiakkaille aiheutuu sen suurempaa häiriötä.

5 SOVELLUSESIMERKIT JA KEHITYSKOHDAT HAASTEINEEN

Tässä luvussa tarkastellaan tekoälyn sovelluksia ohjelmistokehityksessä ja sitä ympäröivissä tehtävissä. Lisäksi esille tuodaan onnistuneita implementointeja sekä niihin liittyviä haasteita ja oppeja.

5.1 Onnistuneet implementoinnit

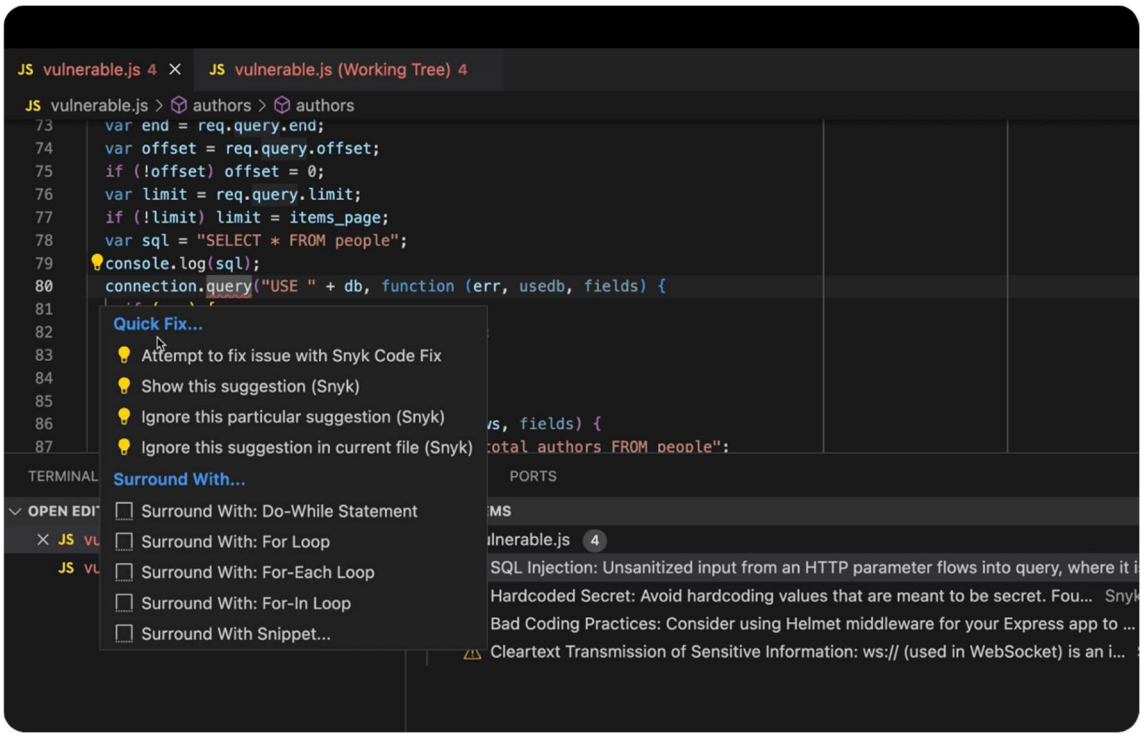
Mitä tulee koodauksen ympäröiviin ja aikaa syöviin tehtäviin, on Grammarly yksi hyvistä tekoälypohjaisista kirjoitusapureista, joka tarjoaa reaaliaikaisia ehdotuksia (kuva 6) ja korjauksia kirjoitetulle tekstille hyödyntäen luonnollisen kielen käsittelyä (NLP) analysoidakseen käyttäjän kirjoittamaa tekstiä. Tämä helpottaa työntekijän taakkaa varsinkin virallisissa kommunikoinneissa asiakkaan kanssa tai yleiseen käyttöön tarkoitettujen dokumenttien kirjoittamisessa, minkä seurauksena käyttäjien kirjoitustarkkuus on parantunut Grammarlyn mukaan jopa 25 % (13).



KUVA 4. Esimerkki Grammarlyn käytöstä (14).

Itse kirjoitusvaiheessa GitHub Copilot on erinomainen tekoälypohjainen koodieditori, joka auttaa ohjelmoijaa kirjoittamaan koodia nopeammin ja vähemmillä virheillä. Työkalu käyttää hyväkseen OpenAI:n GPT-3 mallia ennustamaan ja ehdottamaan ohjelmoijalle seuraavia koodirivejä käyttäjän aloittaessa koodin kirjoittamisen joillain tietyillä avainfunktioilla tai parametreilla. GitHub Copilot-käyttäjien raporttien perusteella tämä on parantanut ohjelmoinnin nopeutta, tuottavuutta ja yhdenmukaisuutta (15).

Koodin tarkistamiseen ja optimointiin on kehitetty DeepCode-työkalu. Tämä koodia analysoiva työkalu analysoi annettua koodia tai ohjelmistoa ja esittää suosituksia koodin parantamiseksi alla olevan kuvan kaltaisesti (kuva 7). Tämä vähentää merkittävästi manuaalista koodin tarkistusta ja parantaa ohjelmistojen laatua ja tehokkuutta. Tämä on vähentänyt asiakaspalautteen perusteella niin virheiden kuin refaktorointien toteuttamiseen käytettyä aikaa jopa 50 % ja parantanut yleisesti ohjelmistojen laatua (16).



KUVA 5. Esimerkki DeepCoden käytöstä analysoinnissa ja koodin korjauksessa (17).

5.2 Haasteet ja oppimiskohdat

Vaikka tekoälyn käyttö ohjelmistokehityksessä tuo merkittäviä etuja, on siinä myös haasteita. Seuraavaksi esitetään esimerkkejä keskeisistä haasteista ja oppimiskohdista tekoälyn implementoinneissa.

Tekoälyn käytössä on tärkeää varmistaa, että käyttäjien tietosuoja säilyy ja että heille kerrotaan avoimesti, miten ja missä heidän tietojaan käytetään. Esimerkiksi Grammarlyn tapauksessa käyttäjäluottamus on säilytetty varmistamalla, että kaikki kirjoitukset analysoidaan turvallisesti ja että käyttäjillä on selkeä käsitys siitä mihin heidän tietojaan käytetään (13).

Resurssit ovat yksi suurista haasteita mitä tulee siihen, kuinka paljon tekoälyä voidaan firmassa käyttää. Tekoälyn käyttö vie resursseja palvelumaksujen tai niitä ylläpitävien serverien kustannusten muodossa, mutta sen ansiosta työntekijöiden ylikuormitus vähenee ja tehokkuus paranee, mikä suurien ja pienempienkin ohjelmistofirmojen olisi hyvä nähdä varsinkin, kun on ennakoitu tekoälyn hyötykäytön tulevan kasvamaan 23–37 % seuraavan 3 vuoden aikana (18).

Jatkuva oppiminen ja mallien parantaminen implementoiduissa tekoälymalleissa on käyttäjäpalautteen sekä käyttödatasta saatujen oivallusten perusteella yksiä suuria haasteita. Tekoälyn käytön kasvu ja sen kehitys tarkoittaa, että ohjelmistofirmojen täytyy olla uudistaa tekniikka melkein vuosittain ainakin ohjelmoinnin saralla, kun ollaan lähestymässä pistettä, missä koko ohjelmointiprojektin toteuttaminen ja nopeus riippuu siitä, kuinka tehokkaita tekoälypohjaisia työkaluja on käytössä.

6 POHDINTA JA JOHTOPÄÄTÖKSET

Tässä osiossa käsittelen tutkimuksen keskeisiä löydöksiä ja niiden merkitystä ohjelmistokehitykselle. Lisäksi tuon esiin suosituksia tulevaisuuden tutkimukseen, jotta tekoälyn potentiaalia voidaan hyödyntää entistä paremmin ja laajemmin ohjelmistokehityksen eri osa-alueilla.

6.1 Tutkimuksen keskeiset löydökset

Tutkimuksesta voidaan nähdä, että tekoälyn integrointi ohjelmistokehitykseen parantaa lopputuotteen laatua sekä tehokkuutta, tehden saman myös sen sisäisille kehitysprosesseille. Tekoäly tukee kehitystä alkaen vaatimusanalyysistä ja suunnittelusta, jatkaen sitä koodaus- ja testivaiheiden automatisointeihin eri osa-alueilla. Tuki jatkuu tiimien välisen kommunikaation helpottamisena eri työkaluilla, antaen samalla projektipäälliköille raportteja, joiden perusteella hän voi tukea tiimiä tilanteeseen sopivilla tavoilla. Tämä kaikki perustuu työkaluihin, kuten älykkäisiin koodieditoreihin ja ennakoiviin testausjärjestelmiin, jotka vähentävät niin virheiden määrää kuin mahdollistavat reaaliaikaista virheiden korjausta tai koodin generointia. Tämä vähentää toistuvan manuaalisen työn määrää, mikä taas vapauttaa kehittäjien aikaa keskittyä luovempiin ongelmanratkaisutehtäviin, joita varten tekoäly ei ole vielä tarpeeksi kehittynyt.

Lisäksi tutkimuksesta voidaan nähdä, että tekoälyn käyttö projektinhallinnassa tehostaa resurssien hallintaa ja aikataulutusta. Toki tämä suurien datamäärien nopeasti läpikäyminen eri tekoälypohjaisilla ennustusalgoritmeilla tuo ylläpitokustannuksia, mutta eri projektin sisäisten ongelma kohtien ja tehokkuutta parantavien toimenpiteiden esiintuominen tuo enemmän kuin tarpeeksi säästöjä ja asiakastyytyväisyyttä, että sijoitus voidaan laskea sen arvoiseksi. Tämä sijoitusarvo voidaan nähdä, kun arviot projektien kestosta ja budjetista ovat luotettavimpia tai ennustettavampia. Tätä tukee taustalla johdettavien tiimien tuloksellisuuden kasvu selkeämpien tavoitteiden ja johdon ansiosta.

6.2 Suositukset tulevaisuuden tutkimukseen

Mahdollista tulevaisuuden tutkimusta ajatellen suosittelisin keskittymään tekoälyn käytön laajentamisesta uusien sovellusten ja menetelmien kehittämiseen. Erityisesti tekoälyn rooli laadunvarmistuksessa ja virheiden korjauksessa on tärkeä kehitys kohde, sillä näiden merkittävyys esimerkiksi ohjelmointia opettelevien opiskelijoiden arjessa on varsin suuri. Laadunvarmistuksessa tekoälyn tuoma suurien käyttöskenaarioiden läpikäyminen voi tuoda esiin aloitteleville kuin kokeneimmillekin ohjelmoijille näkökulmaa siitä, miten omaa koodia ja koodin luomisprosessia kannattaa lähestyä.

Lisäksi olisi tärkeää jatkaa tekoälymallien kehittämistä ja päivittämistä käyttäjäläheisesti, jotta ne pysyisivät tehokkaana ja relevantteina jatkuvasti kehittyvien tarpeiden ja olosuhteiden vuoksi. Koneoppiminen ja jatkuva kehitys myös ihmisten puolelta pitää huolen, että järjestelmät pysyvät ajan tasalla, mutta myös integroituvat paremmin ihmisten arkeen kevyemmän jokapäiväiskäytön ympärillä. Tämä tuo toki mukanaan tietosuoja- ja avoimuuskäytäntöjen kehittymistä ja hyväksynnän etsimistä käyttäjäpuolelta, mutta tätäkin olisi hyvä kehittää suuntaan, missä käyttäjien luottamus saadaan säilytettyä. Tätä sivuten, tulevaisuuden tutkimuksen tuleekin keskittyä myös tekoälyn eettisiin ja yhteiskunnallisiin vaikutuksiin, jotta sen käyttö olisi kestävää, vastuullista ja täten myös viehättävämpää nyky-yhteiskunnan näkökulmasta.

Tekoälyn hyödyntämistä tulee täten jatkaa ja laajentaa innovatiivisten ratkaisujen löytämiseksi, jotta sen toiminallisuudet kaikilla osa-alueilla tukisivat toisiaan helpottaen täten uusien mahdollisten toiminnallisuuksien kehittämistä. Tämä lopputyö toimii ohjenuorana tulevaisuuden ohjelmistokehittäjille, jotka haluavat hyödyntää tekoälyn potentiaalia, mahdollisesti integroiden sitä omaan ohjelmistokehitykseensä luoden täten aina vain parempaa ja parempaa koodia.

LÄHTEET

(1) Tieturi 6.2.2024. *Tekoälyn hyödyntäminen projektinhallinnassa*
Hakupäivä 25.04.2024
https://www.tieturi.fi/blogi/tekoalyn-hyodyntaminen-projektinhallinnassa/

(2) Freshworks 2024. *New-gen project management for modern IT teams*
Hakupäivä 10.05.2024
https://www.freshworks.com/freshservice/lp/project-management/?tactic_id=3389102&utm_source=google-adwords&utm_medium=FS-Search-MidFunnel-InsideEU-North&utm_campaign=FS-Search-MidFunnel-InsideEU-North&utm_term=it%20project%20management%20tools&device=c&matchtype=p&network=g&gclid=Cj0KCQjwjLGyBhCYARIsAPqTz1_ps-J7zCk3m4JwstVdrM1aDj9x5W6595HcwcJ4zsgcdqWCgVmJqz4aAmF7EALw_wcB&audience=kwd-174419423&ad_id=491148936458&gad_source=1

(3) Atef Awad 2024. *Artificial Intelligence (AI) Capabilities Enhance Teamwork*
Hakupäivä 25.04.2024
https://www.linkedin.com/pulse/artificial-intelligence-ai-capabilities-enhance-teamwork-atef-awad-2z2rf/

(4) Visure Solutions, Inc 2024.
Hakupäivä 24.04.2024
https://visuresolutions.com/fi/

(5) Visure Solutions 2024. *Visure Requirements ALM Platform*
Hakupäivä 10.05.2024
https://www.linkedin.com/products/visure-solutions-visure-requirements-alm-platform/?trk=organization_guest_product_card_related-content-card

(6) BrowserStack 2024. *What is AI Testing?*
Hakupäivä 27.04.2024
https://www.browserstack.com/low-code-automation/features/what-is-ai-testing#:~:text=AI%20has%20revolutionized%20this%20process,creative%20aspects%20of%20software%20development

(7) Sana Labs 2024. *Your company's AI assistant*
Hakupäivä 15.05.2024
https://sanalabs.com/assistant?utm_term=artificial+intelligence+document+management&utm_campaign=P+-+Sana+AI+Focus+-+NO+-+BA&utm_source=adwords&utm_medium=ppc&hsa_acc=4334994313&hsa_cam=20326585153&hsa_grp=149992974399&hsa_ad=695701280394&hsa_src=g&hsa_tgt=kwd-369485019111&hsa_kw=artificial+intelligence+document+management&hsa_mt=b&hsa_net=adwords&hsa_ver=3&gad_source=1&gclid=Cj0KCQjwjLGyBhCYARIsAPqTz19zZqvPd4RJ_CE2Krk4R1DDo8wUS9orbPqA6Tho0pDt6DBtD9tB7z8aAr9_EALw_wcB

(8) Cloudester Software LLP 29.02.2024. *AI Integration: Transforming the Future of Software Development*
Hakupäivä 28.04.2024
https://cloudester.com/ai-integration-with-software-development/#:~:text=AI%20integration%20in%20software%20development%20is%20a%20holistic%20shift%2C%20seamlessly,while%20AI%20handles%20routine%20tasks

(9) Amazon Web Services 2024. *Amazon Q Developer, now generally available, includes previews of new capabilities to reimagine developer experience*
Hakupäivä 15.05.2024
https://aws.amazon.com/blogs/aws/amazon-q-developer-now-generally-available-includes-new-capabilities-to-reimagine-developer-experience/

(10) *Nicolas Carlo 2024. Can AI help me refactor legacy code?*
Hakupäivä 27.04.2024
https://understandlegacycode.com/blog/can-ai-refactor-legacy-code/

(11) Miltos Allamanis, Principal Researcher Marc Brockschmidt, Senior Principal Researcher 8.12.2021. *Finding and fixing bugs with deep learning*
Hakupäivä 30.04.2024
https://www.microsoft.com/en-us/research/blog/finding-and-fixing-bugs-with-deep-learning/

(12) Scrums 2024. *Exploring Anomaly Detection for Effective Software Maintenance with AI*
Hakupäivä 01.05.2024
https://www.scrums.com/guides/exploring-anomaly-detection-for-effective-software-maintenance-with-ai

(13) Garima Singh 27.05.2024. *AI App Development Triumphs – A Case Study Showcase*
Hakupäivä 02.05.2024
https://www.appypie.com/blog/case-studies-of-successful-ai-app-development

(14) Grammarly, Inc. 2024. *Responsible AI that ensures your writing and reputation shine*
Hakupäivä 16.05.2024
https://www.grammarly.com/

(15) Visual Studio 2024. *Code completions with GitHub Copilot in VS Code*
Hakupäivä 04.05.2024
https://code.visualstudio.com/docs/copilot/ai-powered-suggestions

(16) *Fredrik Filipsson* 21.02.2024. *AI CODING TOOLS: DEVELOPMENT WITH CUTTING-EDGE TOOLS*
Hakupäivä 05.05.2024
https://redresscompliance.com/ai-coding-tools-development-with-cutting-edge-tools/

(17) Snyk 2024. *Purpose-built AI for secure development*
Hakupäivä 16.05.2024
https://snyk.io/platform/deepcode-ai/

(18) Project Management Institute 2019. *AI Innovators: Cracking the Code on Project Performance*
Hakupäivä 06.05.2024
https://www.pmi.org/learning/thought-leadership/pulse/ai-innovators

(19) Tabnine 2024. *The AI code assistant that you control*
Hakupäivä 24.05.2024
https://www.tabnine.com/

(20) Diffblue 2024. *Autonomous AI unit test suite creation at scale*
Hakupäivä 25.05.2024
https://www.diffblue.com/diffblue-cover/

(21) Devin Schumacher *2025. CodeT5*
Hakupäivä 25.05.2024
https://serp.ai/codet5/

Tekoäly tarjoaa tehokkaita työkaluja ja tekniikoita, jotka voivat parantaa ja tehostaa koodauksen eri vaiheita. Tässä ohjenuoria tekoälyn käyttämiseen eri vaiheissa:

1. Koodin suunnittelu ja ideointi
    - Tekoälypohjaiset ideointityökalut: Tekoäly voi auttaa ideointivaiheessa tarjoamalla ideoita tai esittämällä jatkokysymyksiä tai ongelmakohtia liittyen alkuperäiseen ideaan. GitHub Copilotin tai TabNine (19) kaltaiset ohjelmat voivat antaa esimerkiksi koodiehdotuksia kirjoitetun pseudokoodin perusteella.
    - Tekoäly voi myös auttaa etsimään esimerkki projekteja tai koodinpätkiä, jos sille tarjotaan kysymyksiä tietyntyylisten projektien rakennetarpeista.
2. Koodin kirjoittaminen
    - Tekoälypohjaisilla koodieditoreilla kuten Visual Studio Code:lla ja Github Copilotilla on tarjota automaattista koodin täydennystä tai korjausehdotuksia, jotka aktivoituvat, kun aloittaa kirjoittamaan koodia tai, kun funktio on saatu tehtyä (15).
    - Tekoäly voi myös kääntää valmiin koodin kieleltä toiselle tarpeen mukaan helpottaen eri kielten oppimista ja käyttöä. Tästä hyvänä esimerkkinä Github Copilot, joka voi muuntaa esimerkiksi Python-koodia Ruby-koodiksi (15).
3. Koodin tarkistaminen ja virheiden korjaus
    - Virheiden tunnistukseen ja korjaukseen valmiissa koodissa voi käyttää ohjelmia, kuten DeepCodea, joka analysoi koodia ja tarjoaa sen perusteella parannusehdotuksia (17).
    - DeepCodea voi myös käyttää tietoturva-aukkojen tunnistamiseen ja korjaamiseen (17), mikä on hyödyllistä varsinkin isommissa projekteissa.
4. Testauksen automatisointi
    - Tekoäly voi auttaa luomaan testitapauksia automaattisesti työkaluilla, kuten Diffblue Coverilla, joka generoi testitapauksia pohjautuen koodissa tarvittaviin parametreihin (20).

5. Dokumentointi ja ylläpito
    - Tekoäly voi auttaa luomaan kommentteja ja dokumentaatiota koodille reaaliajassa esimerkiksi CodeT5 nimisen työkalun avulla (21), mikä helpottaa sen ymmärtämistä ja ylläpitoa isommissa ryhmissä tai tulevaisuuden kehitystä ajatellen.
    - Koodin refaktorointi uudempiin ohjelmistokieliversioihin tai yleisesti koodin tehokkuuden kasvattaminen, samat funktiot pitäen, onnistuu tekoälyn avulla. Esimerkiksi Amazon Q developer pystyy tekemään tämän eri Java-versioiden välillä (9).

Yhteenvetona tekoäly hyödyntää koodauksen eri vaiheita tuottavuuden, virheiden vähenemisen ja tehokkaamman oppimisen kannalta. Käyttämällä yllä mainittuja tekoälypohjaisia työkaluja ja tekniikoita voi ohjelmoijat aloittelevasta kokeneempaan tehostaa omaa ohjelmointityötään ja kehittää parempia ohjelmistoratkaisuja.