Ihmisen ja tekoälyn yhteistyö ohjelmistokehityksessä

Susanna Tuomi

Haaga-Helia ammattikorkeakoulu

Haaga-Helian amk-tutkinnot

Amk-opinnäytetyö

2024

Tiivistelmä

Tekijä(t)
Susanna Tuomi

Tutkinto
Tradenomi

Raportin/Opinnäytetyön nimi
Ihmisen ja tekoälyn yhteistyö ohjelmistokehityksessä

Sivu- ja liitesivumäärä
46

Tutkimuksessa tarkasteltiin ihmisen ja tekoälyn yhteistyötä ohjelmistokehityksessä. Kuinka tekoälytyökaluja hyödynnetään ohjelmistokehityksessä ja mitä hyötyä niiden käytöstä voi olla? Missä määrin tekoälytyökalujen käyttö voi tehostaa työntekoa tai tuottavuutta, sekä mitkä ovat ohjelmistokehittäjien asenteet tekoälytyökaluja kohtaan? Liittyykö tekoälynkäyttöön uhkia/eettisiä haasteita, onko vaikutuksia työelämään?

Työn teoriaosuudessa paneuduttiin generatiiviseen tekoälyyn ja siihen liittyviin teknologioihin sekä valittiin tarkasteltaviksi työkaluiksi ChatGPT, GitHub Copilot ja Tabnine. Tutkimusosiossa tutkittiin kuvailevana kirjallisuuskatsauksena kuutta artikkelia vuosilta 2023–2024, joissa analysoitiin työkalujen tehokkuutta, kehittäjien asenteita, vaikutusta työllisyyteen sekä eettisiä haasteita.

ChatGPT ja GitHub Copilot osoittautuivat suosituimmiksi työkaluiksi. Käyttäjät hyödynsivät niitä muun muassa koodin tuottamisessa, virheiden korjaamisessa ja uusien ratkaisujen löytämisessä. ChatGPT suoriutui erityisen hyvin koodiongelmien ratkaisemisessa, mutta sen tehokkuus riippui käyttäjän tarkasta ohjeistuksesta ja arvioinnista. Copilotin käyttö tehosti ohjelmointia, mutta sen tuottamat ratkaisut vaativat usein muokkausta. Yksinkertaisten ongelmien ratkaisemisessa tekoäly oli tehokas, mutta monimutkaisempien haasteiden kohdalla sen suoriutuminen oli heikompaa.

Tekoälytyökalujen käyttötapoja kaksi: kiihdytystilassa ohjelmoija hyödyntää tekoälyä nopeiden ja yksinkertaisten koodirivien tuottamiseen, tutkimustilassa etsitään ratkaisuja tuntemattomiin ongelmiin tai syntakseihin, jolloin ohjelmoijat olivat valmiimpia käymään läpi useita vaihtoehtoja ja muokkaamaan niitä tarpeen mukaan. Molemmissa tiloissa ihmisen rooli oli keskeinen, sillä tekoälyn tuottamaa koodia ei voinut automaattisesti hyväksyä.

Ihmisen ja tekoälyn yhteistyön ja vuorovaikutuksen valossa huomattiin, että vaikka tekoäly pystyy tuottamaan laadukasta koodia, ohjelmoijan vastuulla on varmistaa sen toimivuus ja soveltuvuus. Kokeneet ohjelmoijat arvostivat tekoälyn tuomia etuja, mutta vähemmän kokeneet käyttäjät suhtautuivat työkaluihin kriittisemmin, kokien niiden käytön välillä sekavana.

Vaikka tekoälytyökalut voivat parantaa ohjelmoijien tuottavuutta ja tehostaa työskentelyä, tekoäly tarvitsee yhä tarkkaa ohjeistusta ja ihmisen arviointia, eikä sen tuottamaa koodia voida pitää virheettömänä. Ihmisen rooli kriittisenä tarkastajana säilyy keskeisenä, sekä erityisesti luovuutta ja toimialatuntemusta vaativissa työtehtävissä. Vaikka tekoälyllä voidaan joitakin tehtäviä automatisoida, se ei toistaiseksi vielä korvaa ihmistä täysin, ja seurauksena voi kehittyä uusia ammatteja ja tehtäviä. Eettisesti tärkeimpiä haasteita ovat tietosuoja, syrjinnän ja vinoumien ennaltaehkäisy sekä tekoälyn tuottaman virheellisen tiedon tunnistaminen. Lisäksi tekoälyn vaikutusta ihmisen päätöksentekoon on arvioitava tarkasti, jotta varmistetaan, että tekoäly toimii toivotulla, eettisesti hyväksyttävällä, tavalla.

Asiasanat
ohjelmistokehitys, tekoälytyökalut, chatGPT, GitHub Copilot, Tabnine

Sisällys

1 Johdanto

Suurin osa suuntautumisopinnoistani on ohjelmistokehityksessä, joten halusin ehdottomasti valita aiheen, joka käsittelisi tätä. Lisäksi halusin tutkia ja tarkastella ajankohtaisen ja alati eri aloilla yleistyvän työkalun käyttöä eli tekoälyä. Näin syntyi aihe tälle työlle, eli käytännön katsaus ihmisen ja tekoälyn yhteistyöhön ohjelmistokehityksessä.

1.1 Aiheen ajankohtaisuus ja perustelut aiheen valinnalle

Aihe on tärkeä, koska yleisellä tasolla tekoälyn osallisuutta ja roolia työelämässä ei voi enää tässä kohtaa sivuuttaa. On ilmeistä, että se on tullut jäädäkseen ja tulee ottamaan osaa yhä enenevissä määrin esimerkiksi rutiininomaisiin ja toistuviin työtehtäviin alasta riippumatta. Haluan itse tutkia tekoälyn käyttöä ohjelmistokehityksessä, koska toivon löytäväni näkökulmia tekoälyn käyttämiseen työkaluna käytännön ohjelmointityössä, jota voisin myöhemmin itsekin hyödyntää alalle pyrkivänä. Kiinnostavan aiheesta tekee sen, että on mielenkiintoista selvittää esimerkiksi, miten paljon tekoälyllä voidaan korvata tällä hetkellä ihmisen suorittamaa koodaustyötä. Tulen työssäni tarkastelemaan, miten tekoäly tukee ohjelmistokehittäjää työssään nyt ja lähitulevaisuudessa, sekä yleisesti ottaen asenteita tekoälytyökalujen käytön suhteen. Tekoälyn käytön helppoutta ja nopeutta lähes ylistetään useissa eri lähteissä, joten eihän se voi kuin olla pelkästään hyväksi ihmiselle. Aikaa jää niin paljon muuhun, kun tekoälyllä voi hoitaa ison osan työtehtävistä, vai voisiko sen käyttöön liittyä myös riskejä, joihin huomio ei kiinnity yhtä vahvasti kuin hyötyihin? Pyrin työssäni tutkimaan asiaa objektiivisesti ja tarkastelen myös mahdollisia eettisiä kysymyksiä, joita aiheesta syntyy, mm. kuinka tekoälyn yleistyvä käyttö ohjelmoinnissa voi vaikuttaa työtehtäviin ja työllisyyteen sekä tekoälyn käyttämään dataan.

Aion tutkia tätä aihetta, koska vaikken itse ole vielä alalla, olen itse käyttänyt tekoälyä (esimerkiksi chatGPT) opintojenaikaisissa projekteissa sekä yksin että osana ryhmää. Myös opettajien keskuudessa tähän käytäntöön on kannustettu ja uskon siis, että tekoäly on myös työelämässä enenevissä määrin aktiivisessa käytössä eri ohjelmistokehitystyön vaiheissa. Haluan tuoda esille eri näkökulmia siitä, mitkä ovat siis alalla vallitsevat käytännöt tekoälyn käytön suhteen. Paneudun ohjelmistokehitystekoälytyökalujen hyötyihin mm. tehokkuuden lisäämisessä ja ongelmien ratkaisemisen työkaluna yksilötasolla. Tarkastelussa on myös tekoälyn vaikutukset työelämään ja mahdolliset haasteet ihmisen ja tekoälyn yhteistyössä sekä eettiset kysymykset, joita nousee muun muassa tekoälyn käyttämästä datasta. Tutkimuksen tarkoituksena on siis antaa lukijalle katsaus tekoälyn

käytön käytännöistä ohjelmistoalalla ja sen tarjoamista hyödyistä mutta toisaalta myös saada mahdollisimman kokonaisvaltainen kuva siitä, mihin ja miten tekoälyn käyttö tulee vaikuttamaan lähitulevaisuudessa.

1.2 Tutkimuksen luonne, tausta ja aihekokonaisuus

Käsiteltävä aihe liittyy siis olennaisesti tekoälyyn ja trendeihin sen käytössä nyt ja tulevaisuudessa ohjelmistokehityksen työkaluna.

Aion tutkimuksessani saada kattavan käsityksen tekoälyn tuomista mahdollisuuksista työkaluna yleisellä tasolla, mutta pääpaino on ohjelmistokehityksessä käytössä olevissa käytännöissä.

Tutkimuksen taustana on mielenkiintoni ohjelmistokehitystä kohtaan, alalle pyrkivänä, sekä kiinnostus tekoälyn mahdollisuuksista käytännön tehtävissä.

Laadin raportin vetoketjuperiaatteen rakenteella.

Tutkimusmenetelmänä on perehtyä tekoälyn käyttöä tutkivaan kirjallisuuteen ja tarkastella käyttöä nimenomaan ohjelmistokehitystyössä. Tutkimusosuudessa analysoidaan valittua tutkimusaineistoa (artikkeleita/tutkimuksia), joiden pohjalta pyritään antamaan vastaukset asetettuihin tutkimuskysymyksiin.

Opinnäytetyössä on hyödynnetty ChatGPT-, DeepL- tekoälysovelluksia ideoiden kehittämisessä ja jäsentämisessä. Tekoälyn tuottamia tekstejä on opinnäytetyössä jatkotyöstetty, jotta ne olisivat virheettömiä, relevantteja, selkeitä ja ymmärrettäviä. Kaikkia raportissa mainittuja lähteitä on käytetty oikeaoppisesti, eivätkä ne ole tekoälyn luomia.

1.3 Keskeiset käsitteet

Taulukko 1. Keskeiset käsitteet

Ohjelmistokehitys	Ohjelmistojen koodin osia tai kokonaisuuksia luova, testaava ja ylläpitävä prosessi.
Tekoäly	Tietojenkäsittelytiede, jolla pyrittiin jo 1950-luvulta lähtien luomaan ihmisälykkyyttä simuloivia tai sitä päihittäviä älykkäitä koneita.
Generatiivinen tekoäly	Tekoälyä, joka tuottaa jotakin toimintansa tuloksena, esim. tekstiä, koodia, kuvia, ääntä.

Koneoppiminen	1990-luvun lopussa tullut tekoälyn alakäsite. Koneelle annetaan dataa, jolla se pystyy oppimaan ja sen pohjalta tekemään ennustuksia, päätelmiä ja päätöksiä. Kolme alaluokkaa: 1) Ohjattu oppiminen, 2) Ohjaamaton oppiminen ja 3) Vahvistusoppiminen.
Algoritmit	Algoritmit ohjeistavat, kuinka toimia jonkin ongelman ratkaisemiseksi, yleensä kyseessä on matemaattinen kaava tai menetelmä.
Data	Tekoäly tarvitsee toimiakseen suuren määrän dataa, jolla se ensin koulutetaan ja jota se vastaavasti käyttää myöhemmin päätöksenteossaan
Syväoppiminen	Syväoppimisessa käytetään ohjattua oppimista, käytössä on suuri määrä dataa ja algoritmi toimii neuroverkkoja hyödyntäen. Neuroverkoissa pystytään määrittelemään painokertoimilla asioiden väliset yhteydet tärkeysjärjestykseen.
Itseorganisoituvat kartat (SOM)	Teuvo Kohosen vuonna 1981 kehittelemä neuroverkkojen alaryhmä (Self-organized Map), jossa oppiminen on ohjaamatonta ja kilpailullista: naapurisolut kilpailevat keskenään aktiivisuudesta ja kehittyvät signaalimallien tunnistajiksi ilman valvontaa, itseorganisoituen.
Natural Language Processing (NLP)	Luonnollisen kielen käsittely tarkoittaa koneellisesti tehtävää tekstin luokittelua, luontia tai keskustelua. NLP:ssä käytetään sekä ohjattua oppimista luonnollisen kielen opettamiseen että syväoppimisen menetelmiä.
Large Language Models (LLM)	NLP, jossa mallien kouluttamiseen on annettu suurempi määrä dataa.
Ohjelmistokehityksen tekoälytyökalut	Työkalut, joita ohjelmoija voi käyttää mm. koodin tuottamiseen ja ongelmien ratkaisuun (esim. automaattinen koodin täydennys, vianetsintä) tuottavuuden ja tehokkuuden lisäämiseksi työssään.

OpenAI ja ChatGPT	Vuonna 2022 lanseerattu keskustelunomaisesti, ihmisen kanssa vuorovaikutuksessa, toimiva NLP. ChatGPT tarvitsee toimiakseen ihmiseltä syötteen. ChatGPT voi koodin kirjoittamisen lisäksi mm. tuottaa monenlaisia eri tekstejä halutun tyylin mukaisesti.
GitHub Copilot	Vuonna 2022 lanseerattu OpenAI:n Codex-malliin pohjautuva ohjelma. Codex on GPT-3:n versio, joka on hienosäädetty ja koulutettu koodinkirjoittamistarkoitukseen. Toimintoina mm. koodin ehdotukset reaaliaikaisesti kirjoittaessa sekä koodin automaattinen täyttö.
Tabnine	Vuonna 2018 julkaistu, ja täten toiminut pioneerina tekoälytyökaluna ohjelmistokehityksessä. Toimintoina koodin kirjoittaminen, testaaminen ja ylläpito. Tabninella on generoitu arviolta yli 1 % koko maailman koodeista.
Eettiset haasteet	Tekoälytyökalujen käyttöön liittyvät eettiset huolenaiheet mm. liittyen käytettyyn dataan sekä vaikutuksina työllisyyteen.

2 Tekoälyllä tehokkuutta

Tekoäly ei ole uusi nykyajan villitys vaan se on ollut jo kauan olemassa ja sitä on kehitelty vuosikymmenien ajan. Tekoäly eli englanniksi *Artificial intelligence* (AI) on jo 1950-luvulta lähtöisin oleva, joka pitää sisällään tietojenkäsittelytieteen, jolla pyrittiin luomaan ihmisälykkyyttä simuloivia tai sitä päihittäviä älykkäitä koneita. Vuonna 1950 Alan Turing tutki pystyisikö rakentamaan sellaista tekoälyä, joka pystyisi kommunikoimaan niin hyvin, että kokeen tekijä ei pystyisi erottamaan onko vuorovaikutuksessa ihmisen vai koneen kanssa. Tämän lisäksi 1950- luvulla kehitettiin myös tekoälyohjelma testaamaan matemaattisia teorioita, ja tekoäly kykeni todistamaan 73 % niistä. (Kananen, Puolitaival 2019: 229–230).

Machine Learning eli koneoppiminen on tekoälyn alakäsite: 1990-luvun lopussa koneelle oli mahdollista antaa dataa, jolla se pystyi oppimaan ja sen pohjalta tekemään ennustuksia, päätelmiä ja päätöksiä. Tekoälyn koneoppimisen kehityksen vuoksi esimerkiksi 1997 IBM:n kehittämä Deep Blue tekoäly onnistui päihittämään shakin maailmanmestarin Kasparovin. (Kananen, Puolitaival 2019: 229–230, Dean 12.3.2024).

Tämän jälkeen oli neuroverkkojen kehitysaikaa: *deep learning* (suomeksi syväoppiminen tai neuroverkot) on koneoppimisen tekniikka, jossa käytetään neuroverkkoja datan prosessointiin ja päätöksenteossa (Dean 12.3.2024). Jo vuonna 1981 Teuvo Kohonen esitteli neuroverkkomallin itseorganisoituvat kartat (Self-Organizing Maps, SOM) (Kohonen 1990: 1464). Itseorganisoituvat kartat olivat tehokas työkalu erityisesti, kun tarvittiin paikallista ja rakenteellista järjestelyä tiedon käsittelyssä. Se onnistui järjestäytymään itsenäisesti ja luomaan karttoja, jotka visualisoivat monimutkaisia tietorakenteita. Soveltui erinomaisesti esimerkiksi hahmontunnistuksessa ja robotiikassa monimutkaisten syötteiden, kuten kuvien ja ääniaineiston, analysointiin. (Kohonen 1990: 1476–1477).

On siis totta, että tekoäly tuntuu olevan nyt yksi puhutuimmista aiheista työpaikkojen kahvipöydissä alasta riippumatta, ja tähän on yksinkertaisena syynä se, että se on yleistynyt ja kehittynyt huimaa vauhtia 2010-luvulta lähtien. Huimaa kehitystä on edesauttanut kolme seikkaa: 1) tietokoneiden laskentateho on kasvanut ja muistia on saatavilla hyvin edullisesti 2) tekoälyn käyttöön on valtava määrä dataa saatavilla ja 3) matemaattisten ideoiden toteutus ja jakaminen on tehokasta (Kananen, Puolitaival 2019: 35–36, 127). Vuonna 2011 IBM:n kehittämä tekoäly Watson päihitti kaksi suurmestaria Jeopardy-kisassa, jossa koneen oli osattava tunnistaa myös sanaleikkejä, jotka tekoälylle lähtökohtaisesti ovat erittäin vaikeita opettaa (Kananen, Puolitaival 2019: 230).

Viimeisimpänä kehitysaskeleena on 2020-luvulla alkunsa saanut generatiivinen tekoäly (*generative AI*): tekoälylle annetaan *prompti* eli ohjeistus tekstimuodossa ja sen lisäksi käytössä olevan datan perusteella se onnistuu luomaan kirjoitettua, visuaalista tai auditiivista sisältöä. (Dean 12.3.2024).

Vuonna 2018 kaksi eri tekoälyä päihitti ensimmäistä kertaa ihmisen luetun ymmärtämisen testissä, jonka oli laatinut Stanfordin yliopisto. Voitaisiin siis sanoa, että tekoälystä on nyt kehittynyt sen verran ”älykäs” (itseasiassa tekoälyllä ei ole tekemistä ’älyn’ kanssa, vaan toimintaa ohjaa algoritmiin perustuvien todennäköisyyksien laskeminen (Kananen, Puolitaival 2019: 27)), että siitä on tullut suosittu apuri ja työkalu kovinkin eri aloille ohjelmoinnista, johtotehtäviin ja jopa luoviin aloihin.

2.1 Generatiivinen tekoäly

Tekoälyn tarkentavalla nimityksellä *generatiivinen tekoäly* siis tarkoitetaan tekoälyä, joka tuottaa jotakin toimintansa tuloksena, esim. tekstiä, koodia, kuvia, ääntä. Tällä hetkellä eletään tekoälyn aikakautta, sillä tilastot kertovat, että 50 % organisaatioista ovat ottaneet tekoälyn käyttöön ainakin yhdessä yrityksen osa-alueista ja peräti 70 % työntekijöistä suhtautuu myönteisesti tekoälyn hyödyntämiseen työssään (Dean 12.3.2024). Tulen keskittymään työssäni generatiivisen tekoälyn käyttöön ohjelmointitehtävissä. Seuraavaksi hieman taustatietoa koneoppimisen algoritmeista.

2.1.1 Koneoppiminen, algoritmit ja data

Tässä kohtaa pieni katsaus koneoppimisen algoritmeihin. Algoritmit ohjeistavat, kuinka toimia jonkin ongelman ratkaisemiseksi, yleensä kyseessä on matemaattinen kaava tai menetelmä. Algoritmi valitaan ratkaistavan ongelman perusteella. Algoritmeja on olemassa useita erilaisia, ja usein voidaankin päätyä käyttämään useampaa. (Kananen, Puolitaival 2019: 112). Koneoppiminen voidaan jakaa kolmeen eri alaluokkaan: 1) Ohjattu oppiminen, 2) Ohjaamaton oppiminen ja 3) Vahvistusoppiminen. Keskeisenä asiana on datan kerääminen, sekä sen valmistelu ja käyttö mallin treenauksessa. Kunkin alaluokan algoritmit luokitellaankin sen mukaan, kuinka data annetaan tietokoneen käyttöön. Tekoälyn teknisen koulutuksen aikana syntyy lopulta malli, jota voidaan käyttää täysin uuteen dataan. (Kananen, Puolitaival 2019: 44–47).

Oheisessa taulukossa (Kuva 1) on esitettynä näiden kolmen eri koneoppimisen tyyppien erot.

	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Description	The model learns by mapping labeled inputs to known outputs.	The model is trained without labels and without a specific reward.	The model learns from its environment based on rewards and penalties.
Data	Labeled data	Unlabeled data	No static dataset
Objective	To predict the output of unseen inputs	To discover underlying patterns in the data, such as clusters	To determine the optimal strategy via trial and error

Kuva 1. Koneoppimisen tyypit. Dhamani, Engler 2024.

Ohjatussa oppimisessa data tulee merkitä valmiilla merkinnöillä (engl. *labeled data*), ja tämän jälkeen algoritmilla luodaan malli uuden datan käsittelyyn. Lisäksi on hyvä tiedostaa, että koulutuksessa käytetyn datan laadulla on suuri merkitys mallin toimivuuteen ja tarkkuuteen. (Kananen, Puolitaival 2019: 49-50, Dhamani, Engler 2024).

Ohjaamattomassa oppimisessa dataa ei merkitä kuten ohjatussa oppimisessa, sen sijaan algoritmia pyydetään itsenäisesti etsimään annetusta datasta säännönmukaisuuksia, poikkeamia tai ryhmittymiä (Kananen, Puolitaival 2019: 51-52, Dhamani, Engler 2024). Myös mm. itseorganisoituvissa kartoissa (SOM) oppiminen on ohjaamatonta ja kilpailullista: naapurisolut kilpailevat keskenään aktiivisuudesta ja kehittyvät signaalimallien tunnistajiksi ilman valvontaa, itseorganisoituen (Kohonen 1990: 1464).

Vahvistusoppimisessa dataa ei anneta ennalta lainkaan, vaan algoritmi oppii “lennosta” ja jatkuvasti toimintaympäristössään annettujen palkintojen tai rangaistusten perusteella. Vahvistusoppimisen tuloksena voidaan löytää optimaalinen tapa suorittaa jokin tehtävä. (Kananen, Puolitaival 2019: 158-163, Dhamani, Engler 2024).

Generatiivinen tekoäly lukeutuu syväoppimisen (engl. *deep learning*) kategoriaan. Syväoppimisessa käytetään ohjattua oppimista, käytössä on suuri määrä dataa ja algoritmi toimii

neuroverkkoja hyödyntäen. Neuroverkoissa pystytään määrittelemään painokertoimilla asioiden väliset yhteydet tärkeysjärjestykseen. Englannin termi ‘deep’ viittaakin käytössä olevien neuroverkkojen syvyyteen, eli eri tasot inputin ja outputin välillä. (Kananen, Puolitaival 2019: 44, 127-135, Dhamani, Engler 2024).

2.1.2 NLP – Natural Language Processing

NLP eli suomeksi luonnollisen kielen käsittely tarkoittaa ”koneellisesti tehtävää tekstin luokittelua, luontia eli generointia tai keskustelua” (Kananen, Puolitaival 2019: 141). Käytännössä NLP mm. muuttaa äänen/puheen tekstiksi tai tekstin ääneksi/puheeksi, tiivistää tekstisisältöä tai etsii siitä tiettyjä asioita, sekä mahdollistaa keskustelun käyttäjän kanssa (chatbotit). NLP:ssä on käytetty sekä ohjattua oppimista luonnollisen kielen opettamiseen että syväoppimisen menetelmillä laskettu sanojen etäisyyksiä toisiinsa ja onnistuttu näin löytämään mm. synonyymejä. (Kananen, Puolitaival 2019: 141–149, Dhamani, Engler 2024).

Kun koitti aika, jolloin mallien kouluttamiseen oli mahdollista antaa suurempi määrä dataa, syntyivät Large Language Models (LLM). LLM:t ovat hyvin monipuolisia ja yleiskäyttöisiä, mikä on mahdollistanut niiden käytön monenlaisissa luonnollisen kielen tehtävissä: mm. chat-keskustelut ja tekstin luokittelu tai tiivistäminen. (Dhamani, Engler 2024). Vuonna 2018 OpenAI julkaisi ensimmäisen GPT:n (*Generative Pretraining*), joka oli koulutettu massiivisella datamäärällä erityyppisten aineistojen käsittelyyn. Tätä ennen useimmat NLP-mallit pystyivät toimimaan vain tietyissä ennalta määritellyissä tehtävissä. *Fine-tuning*-periaatteella viitataan malliin, joka on treenattu suurella datamäärällä (LLM), ja voi täten hyödyntää jo oppimaansa räätälöiden sitä erilaisiin tehtäviin ja tarkoituksiin. Vuonna 2022 julkaistu ChatGPT merkitsee viimeisintä läpimurtoa NLP:n historiassa ja se saavuttikin lyhyessä ajassa ennennäkemättömän suuren suosion. (Dhamani, Engler 2024).

2.2 Tekoälytyökalut ohjelmistokehityksessä

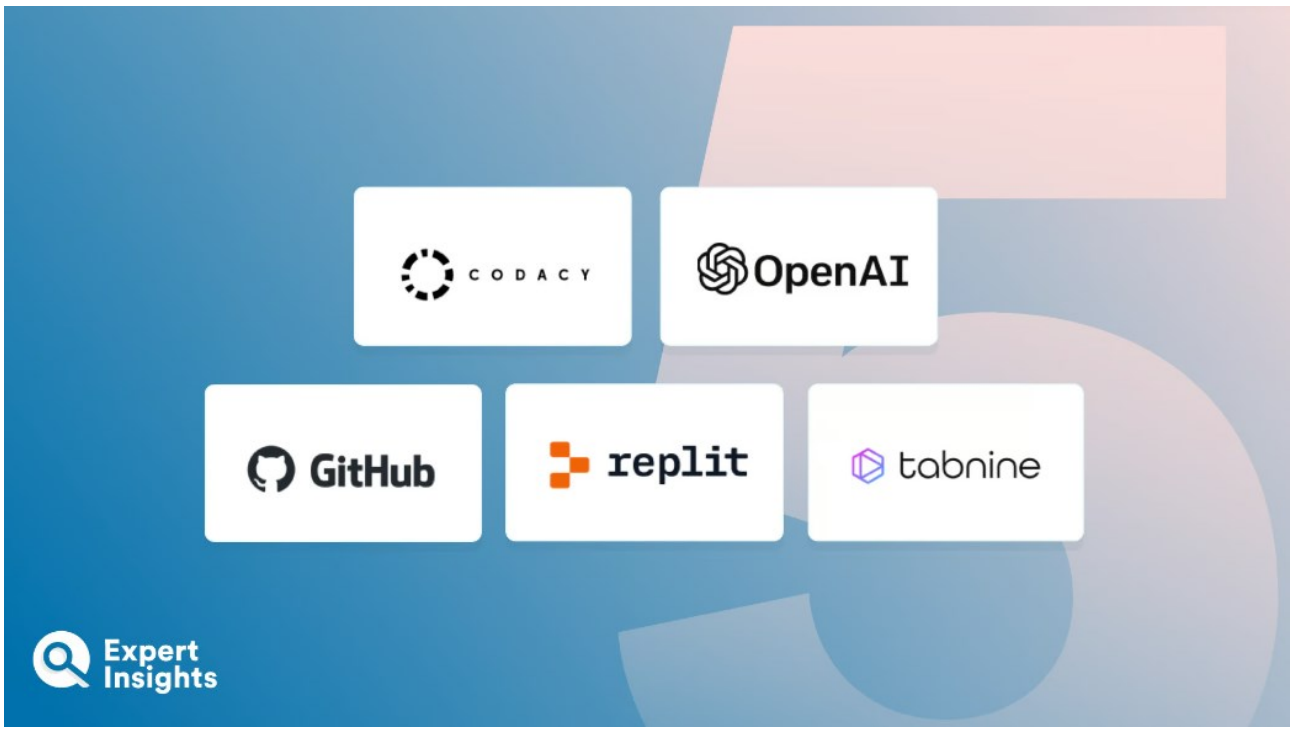
Yksi suosituimmista LLM:n käytöistä keskittyy juuri koodin tuottamiseen. Huolimatta näiden työkalujen käyttöön liittyvistä esiin nousseista haasteista mm. turvallisuuden, läpinäkyvyyden ja lisenssien suhteen, yhä useampi ohjelmistokehittäjä käyttää päivittäin työnsä tehostamiseen tekoälyn työkalua. Koodin generoinnin työkalut alkoivat yleistyä markkinoille tulleen GitHub Copilotin myötä vuonna 2022. Koodin tuottamisen lisäksi näitä työkaluja on käytetty myös muihin tarkoituksiin kuten englantia vieraana kielenä puhuvien avustaminen, koodaushaastatteluihin valmistautumiseen sekä koodin testaamiseen.

On syytä huomata, että näiden ohjelmointityökalujen tarkoituksena (ja näin niitä myös markkinoidaan) on täydentää ihmistä sen sijaan, että ne korvaisivat hänet. Esimerkiksi vaikka Copilot tekee yleensä hyviä ehdotuksia, se ei hahmota ohjelmia samalla tavalla kuin ihminen, jolloin seurauksena voi olla tyhmiltä vaikuttavia virheitä. Vuonna 2023 julkaistu GPT-4 taas suoriutui suhteellisen hyvin LeetCoden (verkkoalusta tietorakenteiden ja algoritmien koodaushaasteiden ratkaisemiseen) helpoista ongelmista mutta sillä oli vaikeuksia vaikeampien ongelmien ratkaisemisessa. Toisin sanoen ihmisen panos koodaustehtävien ongelmanratkaisussa on edelleen keskeinen. (Dhamani, Engler 2024).

Ohjelmistokehityksessä käytettäviä työkaluja on nykyään lukuisia erilaisia. Koodarien keskuudessa on ollut 15 vuoden ajan olemassa kansainvälinen Stack Overflow -yhteisö (Stack Overflow), jossa on vaihdeltu ajatuksia, monesti keskusteltu esiin nousseista ongelmista ja etsitty vertaistukea ja apua tilanteisiin, joissa oma tietämys ei ole riittänyt tai muuten on jonkin työvaiheen tai ongelman kanssa jumissa. Tässä yhteisössä on tehty vuonna 2023 kysely tekoälytyökalujen käytöstä ohjelmistokehittäjän työssä sekä asenteista niiden käytöstä nyt ja tulevaisuudessa. Kyselyyn vastasi yli 90 000 ohjelmistokehittäjää (Stack Overflow kysely). Kyselyssä selvisi, että tekoälytyökaluja käyttävät jo työssään 44 % ja lähiaikoina aikoo 26 % ottaa niitä käyttöön. Eniten on tekoälytyökalut käytössä ohjelmointia opiskelevilla 55 %, ja todennäköistä onkin, että he myötävaikuttaisivat vanhempien pidempään alalla olevien tekoälytyökalujen käyttöönottoon lähivuosina. Toisaalta vaikka tekoälytyökalujen suosio on kasvussa, pitkään alalla ohjelmoineet saattavat vielä suhtautua niihin varauksella. Kysely nimittäin osoittaa, että vain alle 3 % tekoälytyökaluja jo käyttävät tai lähiaikoina käyttävät luottavat niihin täysin, ja toisaalta vajaa 6 % ei luota niihin lainkaan. Vaikka suurin osa (39 %) jokseenkin luottaakin työkaluihin, selvää on, että tekoälyn täytyy vielä todistaa hyötynsä, sillä täyteen luottamukseen on vielä matkaa. Mitä tulee asenteisiin tekoälyn käyttöön myönteisesti suhtautuvat silti jopa 77 %, erityisesti mm. data scientist ja front-end sekä fullstack developer -työtehtävissä. Enemmistö (83 %) käyttää tekoälytyökaluja koodin kirjoittamiseen tai koodin virheenkorjaukseen (*debug*). Tuottavuuden lisääntyminen koetaan suurimmaksi hyödyksi (33 %), toisena oppimisen nopeuttaminen (25 %) ja kolmantena tehokkuuden lisääminen (24 %). (Yepis 2023).

Kolme yleisintä käytössä olevaa työkalua kyselyyn vastanneiden kesken olivat 1) ylivoimaisesti suosituimpana GitHub Copilot 55 % 2) toisena Tabnine 13 % 3) ja kolmantena AWS CodeWhisperer reilulla 5 % kannatuksella. ChatGPT mainitaan kyselyssä AI hakutyökalujen osiossa, jossa se jyrää 83 %:n kannatuksella. (Stack Overflow kysely). Vaikka chatGPT:tä ei tässä tutkimuksessa mainita AI ohjelmistokehityksen työkalujen osiossa, olen sen tässä tutkimuksessani sisällyttänyt yhtenä työkaluna, koska se on yksi tunnetuimmista ja itselläkin käytössä oleva työkalu, varmasti siitäkin syystä, että ilmainen. Työkalujen suosiota olen tutkinut myös käytännön selvitystyöllä teh-

den useita sekä Google- että Bing-hakuja etsien tämänhetkisiä suosituimpia tekoälytyökaluja ohjelmoinnissa. Tämänkin pohjalta olen lopulta päätynyt valitsemaan työkalut, joita haluan analysoida tarkemmin myöhemmissä osioissa. Rajaan näin ollen tutkimuksesta pois AWS CodeWhispererin ja käyn tarkemmin läpi seuraavia työkaluja: 1) OpenAI/chatGPT, 2) GitHub Copilot, ja 3) Tabnine.



Kuva 2. Parhaimmiksi ja käytetyimmiksi todettuja koodaajien työkaluja vuonna 2024 (Witts)

Ohjelmistokehityksen tekoälytyökaluja käytetään käytännössä seuraavanlaisiin tehtäviin:

- Tuottavuuden lisääminen: Kehittäjät käyttävät tekoälytyökaluja nopeuttaakseen koodausta automaattisen koodin täydentämisen avulla. Tämä vähentää koodiin meneviä virheitä.
- Virheiden havaitseminen: Nämä työkalut voivat havaita virheet itsenäisesti ja tarjota ehdotuksia niiden korjaamiseksi, tämä taas hyödyttää myös testaamisvaiheessa.
- Koodin optimointi: Tekoälytyökalut tarjoavat tietoa sovelluksen mahdollisista suorituskykyongelmista ja voivat tehdä ehdotuksia koodin yleisen toiminnallisuuden parantamiseksi.
- Monimutkaisten järjestelmien yksinkertaistaminen: NLP:tä hyödyntäen tekoälytyökalut kykenevät auttamaan kehittäjiä ymmärtämään monimutkaista koodia ja dokumentaatiota.
- Ennakoiva ylläpito: Kehittäjät käyttävät tekoälytyökaluja varmistaakseen järjestelmän vakauden ja luotettavuuden. Tekoälyä hyödynnetään järjestelmän reaaliaikaisessa seurannassa ja analysoinnissa, jolloin se voi havaita mahdolliset ongelmat varhaisessa vaiheessa.

(York 2024).

Hyvä tekoälytyökalu ohjelmistokehitykseen tukee kehittäjää useilla eri osa-alueilla. Ensinnäkin se tarjoaa koodiehdotuksia ja automaattista täydennystä, mikä nopeuttaa kehitysprosessia ja vähentää virheiden määrää merkittävästi.

Lisäksi tällainen työkalu tarkastelee jatkuvasti koodia ja analysoi sen laatua, tyyliä sekä mahdollisia ongelmakohtia. Työkalun tulee myös täyttää tietosuoja- ja tietoturvavaatimukset varmistaakseen, että sekä kehittäjien tiedot että itse koodi pysyvät suojattuina.

Työkalu osaa myös ehdottaa parannuksia, jotka voivat liittyä esimerkiksi tehokkuuden parantamiseen. Se tukee lisäksi automaattista koodin testausta ja yksikkötestausta, jotka ovat olennaisia ohjelmistojen laadunvarmistuksessa ja virheiden löytämisessä mahdollisimman varhaisessa vaiheessa.

Hyvin suunniteltu tekoälytyökalu voidaan integroida saumattomasti kehittäjän päivittäin käyttämiin kolmannen osapuolen sovelluksiin ja ohjelmiin. Lopuksi, työkalun tulee tukea useita eri ohjelmointikieliä ja ymmärtää luonnollisia kieliä, mikä helpottaa vuorovaikutusta kehittäjän kanssa ja tehostaa työskentelyä entisestään. (York 2024).

Seuraavaksi tarkastelen kutakin kolmea valitsemaani työkalua ja pyrin nostamaan esille niin hyödyt kuin puutteet kunkin osalta.

2.2.1 OpenAI / ChatGPT

Kun syksyllä 2022 ChatGPT julkaistiin, se lanseerattiin kyseisillä saatesanoilla (vapaasti suomennettu englannista):

> ”Olemme kouluttaneet mallin nimeltä ChatGPT, joka toimii keskustelunomaisesti. Dialoginen muoto mahdollistaa sen, että ChatGPT voi vastata jatkokysymyksiin, myöntää virheensä, kyseenalaistaa virheelliset lähtökohdat ja hylätä sopimattomat pyynnöt.”
>
> (OpenAI Blog)

Kouluttamiseen on käytetty vahvistusoppimisen menetelmää ihmiseltä saadun palautteen perusteella (*Reinforcement Learning from Human Feedback*). Kouluttautuminen tapahtui Azuren tekoälyn supertietokoneinfrastruktuurissa valtavalla määrällä ihmisen kirjoittamaa dataa (kirjat, artikkelit, asiakirjat) eri aiheista, tyyleistä ja genreistä. (OpenAI Blog, Salo 2023: 43–45).

ChatGPT on siis suunniteltu toimimaan vuorovaikutuksessa ihmisen kanssa, koko sen toiminta perustuu tähän. ChatGPT tarvitsee toimiakseen ihmiseltä syötteen ja toimii keskustelumuodossa.

ChatGPT voi koodin kirjoittamisen lisäksi tuottaa monenlaisia eri tekstejä ja halutun tyylin mukaisesti. Työssäni kuitenkin tarkastelen ChatGPT:tä vain siltä osin kuin ohjelmistokehitystyössä sen käyttöön tukeudutaan. ChatGPT:ltä voidaan pyytää koodin tarkistusta tai vian selvittämistä, antamalla sille koodia, jolloin vastauksena saadaan ehdotuksia mahdollisista keinoista, joilla korjata vika. Vaihtoehtoisesti chatGPT:tä voidaan pyytää generoimaan koodia toivottuun tarkoitukseen, esimerkiksi koodata jokin sovelluksen osa tietyllä ohjelmointikielellä. Osoitteessa chat.openai.com toimivasta ohjelmistosta on saatavilla kaksi versiota:

1) ilmainen versio ChatGPT 3.5., joka toimii niin selaimessa kuin iOS ja Android järjestelmissä
2) 20 dollarin kuukausihintaan GPT-4 versio, jossa ilmaista versiota laajemmat ja monipuolisemmat mahdollisuudet kuten keskustelu myös kuvien ja äänen avulla, sekä kuvien generointia ja jopa uusia välineitä videoiden sisällön tiivistämiseen (Vision). (OpenAI ChatGPT, Dean 2024)

Kummassakin versiossa chatGPT:n käytöstä tekee mukavaa se, että se muistaa käydyt keskustelut. Voidaan pyytää tarkennusta jo aiemmin kysyttyyn tai uutta näkökulmaa aikaisemmin keskustelussa esitettyyn asiaan, mikä toisin sanoen luo vaikutelman aidosta keskustelutilanteesta (Salo 2023: 43–45).

ChatGPT toimii virtuaalisena ohjelmointikumppanina, joka voi:

1) tehdä koodiehdotuksia (koodin kirjoitusta, vian korjausta yms.) kuvatun ongelman perusteella
2) luoda automaattisesti dokumentaation ja kommentteja koodin rakenteen ja toiminnallisuuden perusteella
3) tarjota henkilökohtaisia koulutustilaisuuksia simuloimalla reaalimaailman koodausskenaarioita ja välittömän palautteen antamista
4) avustaa koodin tarkistuksessa tuoden esiin ongelmia ja ehdottaen ratkaisuja jopa koko järjestelmän tason optimointiin
5) tuottaa automaattisia päivityksiä ja pystyy jopa ymmärtämään vaatimuksia jäsentämällä luonnollisen kielen syötteitä ja tuottamaan alustavaa koodia tai järjestelmäsuunnitelmia.

(Rahmaniar 2023: 4–5).

ChatGPT pystyy suoriutumaan monenlaisista käytännön ohjelmointitehtävistä tehokkaasti ja joustavasti. Se osaa esimerkiksi luoda funktioita halutuilla toiminnallisuuksilla, samalla myös selittäen ja kommentoiden tuottamaansa koodia. Lisäksi se kykenee rakentamaan tietokantaschemoja annettujen taulukoiden perusteella. Taulukon sisältöä voidaan myös muokata keskustelun edetessä.

ChatGPT voi myös antaa ehdotuksia koodin optimointiin, esimerkiksi auttamalla vähentämään suoritusajan kestoa. Lopuksi, se osaa auttaa virhetekstien tulkinnassa ja ohjata koodin virheiden korjauksessa. (Alto 2023).

Merkittävin hyöty, jonka käyttäjä, tässä tapauksessa ohjelmoija, voi saada, on koodin tuottamisen nopeutuminen. ChatGPT tekee koodiehdotuksia ja jopa kirjoittaa suurempia koodinpätkiä, tämän seurauksena voi ohjelmoijalla kulua vähemmän aikaa koodaamiseen ja myös virheiden määrä koodissa pienentyä. Ohjelmoijan on kuitenkin aina syytä tarkistaa, että tuotettu koodi vastaa laadultaan alan standardeja. (Rahmaniar 2023: 2–5).

On syytä myös tiedostaa, että vaikka kyseessä on ihmiskeskustelua vahvasti simuloiva työkalu, se ei toimi täysin moitteetta ja onkin syytä suhtautua siihen joissain määrin myös varauksellisesti. ChatGPT saattaa joskus tuottaa uskottavan kuuloisia, mutta virheellisiä tai järjettömiä vastauksia, joka johtuu osittain myös siitä, ettei se ”tiedä” asioita samalla tavalla kuin ihminen vaan ennustaa sanojen järjestystä pohjautuen mallin parametreihin. Syötteen sanamuodolla on myös väliä, eli jos saman asian kysyy hieman eri sanamuodolla saattaa saada myös hieman erilaisen vastauksen. (Salo 2023: 43–45, OpenAI Blog). ChatGPT ei myöskään aina ymmärrä kontekstia, erityisesti monitiedosto-ohjelmistoprojekteissa, tällöin tarvitaan ihmisen valvontaa ja mahdollisia korjaustoimenpiteitä (Rahmaniar 2023: 2).

Mainittakoon myös, että ohjelmoijalta vaaditaan jonkin verran ymmärrystä ja osaamista ohjelmointikielistä ja tekniikoista, jotta chatGPT:tä voi hyödyntää parhaalla mahdollisella tavalla. Laskennallisella ajattelulla on huomattu olevan vaikutusta ChatGPT:n optimaaliseen käyttöön (Jeuring, Roel & Hieke 2023: 9).

2.2.2 GitHub Copilot

GitHub Copilot on vuonna 2022 lanseerattu OpenAI:n Codex-malliin pohjautuva ohjelma. Codex on GPT-3:n versio, joka on hienosäädetty ja koulutettu koodinkirjoittamistarkoitukseen yli tusinalla eri ohjelmointikielellä. GitHub Copilot ehdottaa koodia reaaliaikaisesti kirjoittaessa, täyttää automaattisesti toistuvaa koodia, näyttää vaihtoehtoisia ehdotuksia sekä muuntaa kommentit koodiksi. (Dhamani, Engler 2024).

Toisin kuin chatGPT GitHub Copilotista ei ole saatavilla täysin ilmaista versiota. Omaan käyttöön jäsenyyden saa 10 dollarin kuukausimaksulla. Yrityksille on lisäksi olemassa omat hintavammat paketit. GitHub Copilot ilmoittaa sivuillaan yli 50 000:n yrityksen tällä hetkellä käyttävän Copilotia ja

viittaavat osaltaan myös aiemmin mainitsemaani Stack Overflow:n kyselyyn, jossa 55 % vastaajista suosi Copilotin käyttöä. (Github Copilot 2024, Stack Overflow kysely). Tässä kohtaa ei voi olla mainitsematta GitHubia, eli yksi käytetyimpiä, ellei käytetyin versionhallintatyökalu ohjelmistokehityksessä. Koska ohjelmistoprojekteissa on usein monia koodaajia samanaikaisesti, versionhallinnan avulla pidetään yksinkertaisesti huoli siitä, että jokaisen koodaajan tekemät muutokset rekisteröidään erikseen, jolloin pystytään tarvittaessa palaamaan aiempaan versioon virheen sattuessa. Versionhallinnalla pystytään ylipäänsä seuraamaan projektissa tehtyjä muutoksia, parannuksia ja korjauksia. Copilot on siis yksi työkalu monien muiden joukossa, joita GitHub käyttäjilleen tarjoaa. Väitän, että myös työkalun markkinoinnilla ja näkyvyydellä ohjelmoijien keskuudessa on ollut suuri vaikutus työkalun suosioon.

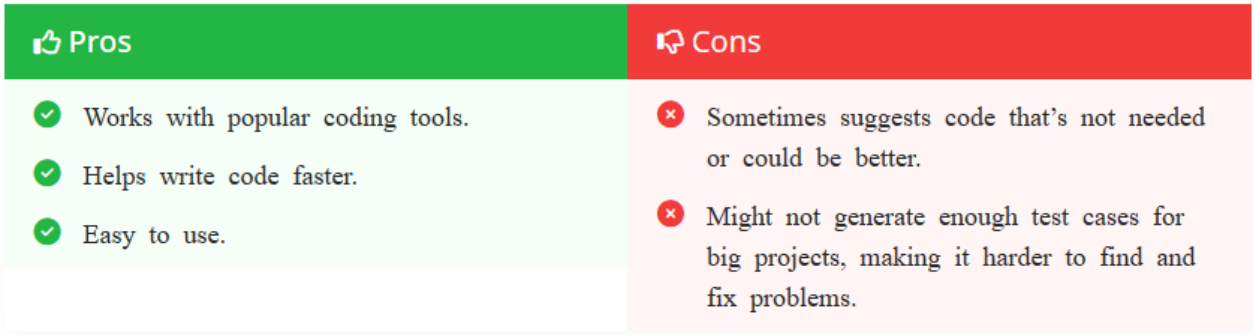
GitHub Copilot integroituu seuraaviin ohjelmointiympäristöihin kuten Visual Studio Codeen, Visual Studioon, JetBrains IDE:iin ja Neovimiin. Chat-toiminto on tällä hetkellä käytettävissä vain Visual Studio Codessa, JetBrainsissa ja Visual Studiossa. GitHub Copilot perustuu GitHubin, OpenAI:n ja Microsoftin kehittämiin generatiivisiin tekoälymalleihin. Se on koulutettu luonnollisen kielen tekstillä ja lähdekoodilla, joka on peräisin julkisesti saatavilla olevista lähteistä, mukaan lukien GitHubin julkiset repositoriot, eli GitHub käyttäjien projektien koodiaineisto. GitHub Copilot on koulutettu kaikilla kielillä, mutta kunkin kielen osalta ehdotusten laatu riippuu kyseisen kielen koulutusdatan määrästä ja laadusta: esimerkiksi JavaScriptillä löytyy paljon dataa repositorioista mutta ”harvinaisemmilla” kielillä se voi tuottaa vähemmän luotettavia ehdotuksia.

GitHub Copilotin toimintaperiaatteita:

1) *Koodiehdotuksen luomiseksi* Copilot-laajennus tutkii koodia editorissa keskittyen kursoria edeltäviin ja sen jälkeisiin riveihin, mutta myös muihin editorissa avoinna oleviin tiedostoihin ja repositorien URL-osoitteisiin tai tiedostopolkuihin asiaankuuluvan asiayhteyden tunnistamiseksi.
2) *Koodieditorissa käytävää keskustelua* varten Copilot-laajennus luo kontekstisidonnaisen kehotuksen yhdistämällä kehotuksen ja muita konteksteja, kuten aktiivisessa asiakirjassa avoinna olevan kooditiedoston, koodivalinnat ja muita yleisiä tietoja kuten ohjelmistokehykset, ohjelmointikielet ja dependenssit.
3) Jotta voidaan luoda *ehdotus GitHub.com-sivuston chatissa*, esimerkiksi antaa vastaus chatissä kysyttyyn kysymykseen, Copilot luo kontekstisidonnaisen kehotuksen yhdistämällä kehotuksen ja muita konteksteja, kuten aiemmat kehotukset, GitHub.com-sivuston avoimet sivut sekä koodipohjasta tai Bing-hakupalvelusta haettu konteksti.

4) Kaikissa kolmessa em. tapauksessa tiedot lähetetään Copilotin malliin, joka tekee *todennäköisyyspohjaisen* määrityksen siitä, mitä todennäköisesti tulee seuraavaksi, ja luo ehdotuksia.

(GitHub Copilot 2024).



Kuva 3. Pros & Cons GitHub Copilot. Gombrich

Kuvan 3 avulla voimme tiivistää, että hyvinä asioina koetaan sen yhteensopivuus monen alan työkalujen kanssa, se nopeuttaa koodin kirjoittamista sekä sen helppokäyttöisyys. Haastavaa Copilotin käytöstä taas tekee se, että joskus sen ehdottama koodi ei ole tarpeellista tai se ei ole tarpeeksi laadukasta. Lisäksi on havaittu, että suuremmissa projekteissa se ei välttämättä generoi tarpeeksi testitapauksia, jotta pystyttäisiin varmasti havaitsemaan ja korjaamaan kaikki mahdolliset viat.

Copilotin käytön puolesta puhuu myös tiettävästi ensimmäinen tieteellisesti toteutettu tutkimus ammattilaisohjelmoijien tekoälytyökalujen käytön tuottavuuden mittaamiseen, joka toteutettiin keväällä 2023. Tutkimuksessa testiryhmää, joista n. puolet käytti ohjelmoinnissaan GitHub Copilotia ja puolet taas ei, pyydettiin suorittamaan ohjelmointitehtävä. Tutkimuksen tuloksena saatiin, että Copilotia tehtävässä käyttäneet ohjelmoijat suoriutuivat annetusta testistä jopa 55,8 % nopeammin. Yleisesti ottaen huomattiin myös että Copilotin käytöstä hyötyivät erityisesti ne, joilla vähemmän työvuosia takana. (Peng, Kalliamvakou, Cihon & Demirer 2023: 5–7).

2.2.3 Tabnine

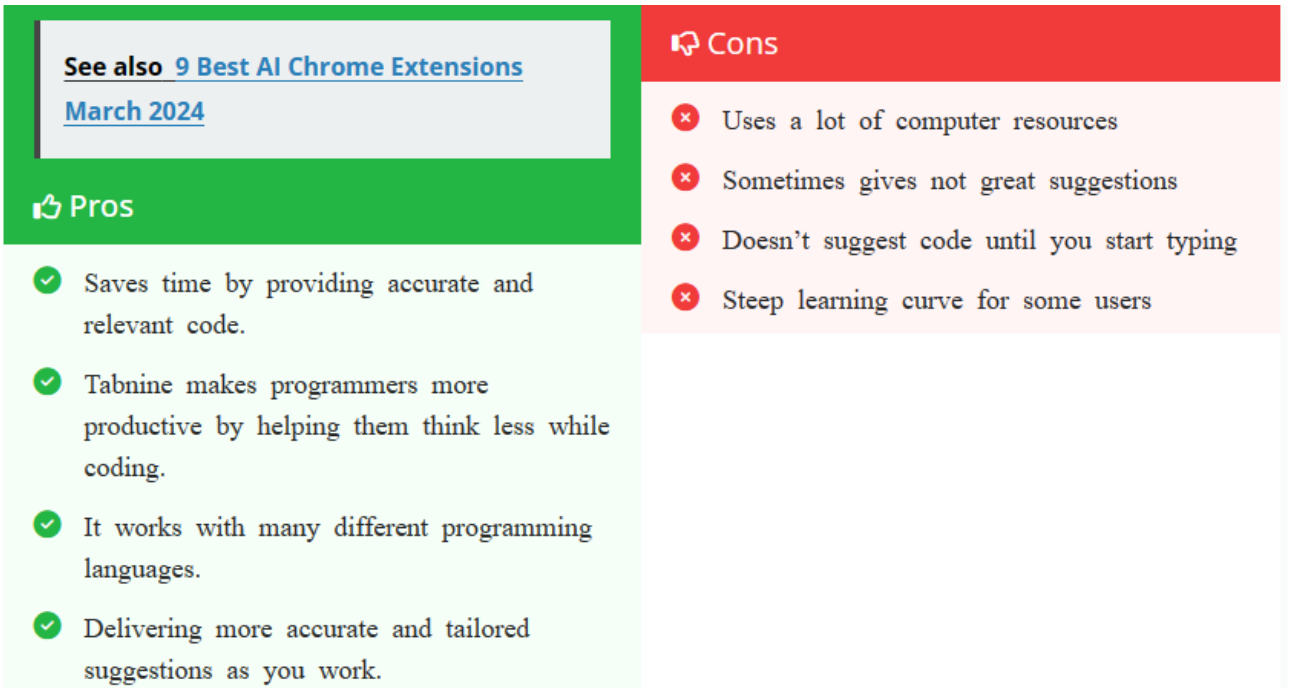
Tabnine markkinoidaan AI koodausassistenttina, joka auttaa kirjoittamaan, testaamaan ja ylläpitämään koodia nopeammin, osana tiimiä tai yksin työskentelevänä kehittäjänä, ja sen käyttö onnistuu monissa kehitysympäristöissä. Tabnine julkaistiin ensimmäisen kerran 2018 ja se onkin toiminut pioneerina tekoälytyökaluna ohjelmistokehityksessä. Lukuina Tabninea käyttää kuukausitasolla yli miljoona koodaajaa, se automatisoi 30–50 % koodin kirjoittamisesta ja onkin generoinut arviolta yli 1 % koko maailman koodeista. (Tabnine 2023).

Myös Tabnine, kuten muut ohjelmistokehityksen tekoälytyökalut, on koulutettu suurella koodimäärällä ja ennustaa, mitä todennäköisesti kirjoitetaan seuraavaksi, ja antaa ehdotuksia. Suurten kielimallien (LLM) nopea kehitys mahdollisti harppauksen eteenpäin tekoälytyökaluille. Koneoppimisalgoritmien avulla Tabnine kykenee ymmärtämään koodauskieltä, havaitsemaan virheitä, ehdottamaan parannuksia ja kirjoittamaan itse koodinosia. Tabnine voi toisin sanoen suorittaa monia tehtäviä, jotka perinteisesti ovat olleet ihmisten suorittamia. (Tabnine Code Assistant Guide 2023).

Tabninea voi hyödyntää seuraavissa tehtävissä:

1) *Koodin täydentäminen*, mikä vähentää koodin kirjoittamiseen ja virheenkorjaukseen kuluvaa aikaa. Tekoäly voi ennustaa kokonaisia koodilohkoja muuttujien ja funktioiden nimien täydentämisen lisäksi. Tabnine analysoi reaaliaikaisesti kontekstin, ymmärtää mihin ollaan pyrkimässä ja ehdottaa näiden pohjalta koodia, mitä todennäköisimmin oltaisiin kirjoittamassa. Lisäksi voidaan automatisoida useita tehtäviä esim. tietokantakyselyjen ja yksikkötestien kirjoittamista.
2) *Virheiden tunnistaminen ja korjaus*, mikä parantaa koodin laatua. Tabnine analysoi koodia sitä kirjoittaessa ja havaitsee virheet saman tien, oli sitten kyseessä puuttuva puolipiste, väärin nimetty muuttuja tai logiikkavirhe, ja ehdottaa ratkaisun.
3) *Koodin tarkastelu ja tutkiminen*. Tabninen kanssa voi keskustella ja pyytää siltä kantaa koodin laadun arvioimisessa toivottujen ominaisuuksien valossa. Tämä keskustelumahdollisuus tehostaa koodin tarkistusprosesseja, auttaa virheenkorjauksessa ja tarjoaa kehittäjille helpon tavan ymmärtää ja tarkastella suuriakin määriä koodeja.
4) Nuorempien (Junior) *kehittäjien oppimisen nopeuttaminen* ja tuki. Koodin generointi ja tiedustelut luonnollisen kielen ’promptien’ avulla. Tämä ominaisuus ei ainoastaan nopeuta koodausprosessia, vaan se myös tukee niitä ohjelmoijia, jotka eivät ole täysin perehtyneet tiettyyn kieleen tai kirjastoihin.
5) *Yhteistyön ja parhaiden käytäntöjen parantaminen* sekä tiimien koodauskäytäntöjen yhdenmukaistaminen. Tabnine ehdottaa tapoja, joilla koodista voidaan tehdä tehokkaampaa, puhtaampaa ja helpommin ylläpidettävää.

(Tabnine Code Assistant Guide 2023).



Kuva 4. Pros & Cons Tabnine. Gombrich

Kuvan 4 avulla voidaan tiivistää Tabninen hyödyt sekä toisaalta myös asiat, joissa olisi vielä parannettavaa. Tabninen koetaan lisäävän tehokkuutta ja säästävän aikaa, ja sen tarjoamiin koodiehdotuksiin ollaan yleisesti ottaen tyytyväisiä. Tabnine tukee monia ohjelmointikieliä, ja se osaa ehdottaa yhä tarkempia, kontekstiin sopivia, räätälöityjä ehdotuksia. Parannusta toivottaisiin sen resurssien käyttöön, sillä nykyisellään käyttää niitä paljon. Toisinaan sen ehdotukset eivät vastaa odotuksia, eikä se tee ehdotuksia kuin vasta sen jälkeen kun koodia alkaa kirjoittaa. Lisäksi sen käyttöönotto voidaan kokea hankalaksi.

3 Tekoälytyökalujen vaikutukset työelämään ja eettiset haasteet

3.1 Tulevaisuuden näkymät tekoälypohjaisten työkalujen käytössä

Yleisesti ottaen, kuten jo aiemmissa luvuissa todettiin, tekoälyn käyttö on levinnyt monilla aloilla ja jo juurtunut moniin tehtäviin yhteiskunnassamme. On otettava huomioon, että tekoälyn hyödyntäminen eri liike-elämän osa-alueilla tulee vain lisääntymään ja monipuolistumaan. Sitran vuoden 2023 megatrendeissä todetaan seuraavan teollisen vallankumouksen olevan datan ja tekoälyn vallankumous, joka ihannetapauksessa tehokkuuden lisäämisellä vapauttaisi ihmiset tekemän luovempia asioita. Julkaisussa painotetaan kuitenkin myös sääntelyiden välttämättömyyttä yhteisten pelisääntöjen sopimiseksi, ja Euroopassa pyritään edistämään yksilön oikeuksia. Tekoälysovellukset tulevat yleistymään ja ulottumaan laajemmalle koko yhteiskunnan osa-alueille: räätälöidyistä suosituksista itseohjautuviin autoihin. Dataa tullaan myös keräämään entistä enemmän ja käyttämään tekoälyn algoritmeissa, jolloin on ensisijaisen tärkeää olla kriittistä ajattelua siitä, kuinka ja mistä data kerätään, ja minkälaista se on, sekä ymmärrystä tekoälyn käyttämän tiedon vinoumista. (Megatrendit Sitra 2023).

Haas tuo esiin blogissaan esimerkkinä pienet muutaman hengen ohjelmistoalan start up yritykset, joilla on nyt tekoälytyökalujen avulla samat, ellei paremmat mahdollisuudet tulla markkinoille suurempiin sadan työntekijän teknologiajätteihin verrattuna (Haas 2023). Yrityksen perustamisen kustannuksia saadaan pienennettyä ja samalla tuote saadaan lanseerattua markkinoille nopeammin. Lisäksi avautuu myös uusia mahdollisuuksia, kun tekoälytyökalun käyttö yhdessä *no-code*:n (ei koodaamista vaativa työskentelytapa) kanssa esim. käyttöliittymä, joka pohjautuu generatiivisen tekoälyn luomaan teknologiaan. Tekoälytyökalun kanssa keskustelemalla voisi siis rakentaa täysin toimivan sovelluksen ilman koodaamista – näin paljon tulevaisuuden näkymät ovat muutamassa vuodessa muuttuneet. (Haas 2023). Tekoälyä hyödyntävät low- ja no-code alustat (joissa käyttäjän ei tarvitse juurikaan kirjoittaa koodia) kuten Retool, Bubble ja OpenAI Codex mahdollistavat sovellusten rakentamisen nopeammin ja ilman suurta ohjelmoinnin tuntemusta.

On ennustettu, että vuonna 2024 kaikesta ohjelmistokehityksestä ilman koodia tuotetun osuus olisi jopa 65 %, sekä ohjelmistokehitystiimien tuottavuus paranisi 10-kertaisesti. (Nanda 2023). Myös Forbesin katsauksessa ennustettiin vuonna 2022, että ohjelmistojen kehittäminen, ohjelmistojen laadun tarkistaminen ja ylipäänsä datan käsittely tulevat seuraavien 4–10 vuoden aikana olemaan yhä enenevissä määrin tekoälyn suorittamaa automatisoitua työtä, koska elämme vahvasti digitalisoituvassa ja nopeatempoisessa maailmassa, jossa tarve kasvaa uusille tehokkaille ja laadukkaille ohjelmistoille (Forbes Technology Council 2022).

Aaltonen (2019: 25–27) uskoo tekoälyn muuttavan työmarkkinoita radikaalistikin mutta ei sinänsä automaattisesti vähentävän työllisyyttä. Työtehtävät tulevat hänen mielestään väistämättä muuttumaan, osa työtehtävistä poistuu ja puolestaan taas syntyy uusia. Tarvitaan ihmisiä tekoälyjärjestelmien suunnitteluun ja rakentamiseen samalla kuitenkin huolehtien siitä, että ehkäistään ihmisten polarisoituminen ja eriarvoistuminen työelämässä. Tuloerojen ja varallisuuden voimakas uudelleenjako sekä työelämästä syrjäytyminen ovat mahdollisia tekoälyn vaikutuksia tulevaisuuden yhteiskuntaan ja näihin tulee suhtautua vakavasti (Taulli 2023). Salo (2023: 164–165) linjaa, että tekoäly voi automatisoida monia työtehtäviä, esimerkiksi koodaajien toistuvat ja rutiininomaiset työtehtävät se pystyy monilta osin hoitamaan. Tästä voi seurata työttömyyden kasvua ja siten myös sosiaalista eriarvoistumista. Toisaalta Salo (2023: 164–165) uskoo, että tekoälyn ja sen kehityksen ympärille voidaan luoda uusia työpaikkoja ja toimialoja kuten tekoälyasiantuntijat, datatieteilijät ja koneoppimisen kehittäjät, joita jo tänä päivänä tarvitaan enenevissä määrin.

Stack Overflow kyselyyn (Yepis 2023) vastanneilta kysyttiin myös, kuinka he uskovat tekoälytyökalujen muuttavan työskentelyään tulevan vuoden aikana. Sekä koodin kirjoittamisen että koodin virheenkorjaamisessa n. 75 % katsoi, että työnkuva muuttuisi jonkin verran tai paljon. Myös laajemmalla ohjelmistoprojektityöskentelyn tasolla koettiin, että tekoälytyökalut muuttaisivat jonkin verran tai paljon (n. 65 %) työtehtäviä/prosesseja mm. seuraavilla osa-alueilla: projektin suunnittelu, commitointi eli koodimuutosten tallentamista versionhallintajärjestelmään ja koodin tarkistus, käyttöönotto ja monitorointi sekä tiimissä työskentely.

Erityisesti yritysten tarpeisiin esimerkiksi GitHub Copilot Business ja Enterprise versioissa tarjotaan yksilötason Copilotiin palveluiden lisäksi organisaatiolisenssien sekä käytäntöjen hallintaa sekä IP-korvauksia. GitHub Copilot Business versiossa Copilot toimii koodausympäristössä eli IDE:ssä (*Integrated Development Environment*) ja CLI:ssä (*Command Line Interface*), ja vuoden 2024 alussa myös mobiilina. GitHub Copilot Enterprise sisältää edellä mainittujen lisäksi räätälöintimahdollisuuden organisaatioille sekä Copilotin integroituna GitHub.comiin chat-käyttöliittymänä, jonka avulla kehittäjät voivat keskustella koodistaan. GitHub Copilot Enterprise voi lisäksi luetteloida organisaation koodipohjan, jotta asiakkaan tietämys ymmärretään syvällisemmin räätälöityjä ehdotuksia varten. (GitHub GitHub Copilot 2024).

Gupta (2023) suhtautuu ohjelmistokehityksen tekoälytyökalujen käyttöön optimisesti. Hän ei näe, että ne tulisivat korvaamaan ohjelmoijat, vaan sen sijaan ohjelmoijien tulisi tutkia sitä, kuinka tekoälyä hyödyntämällä saadaan tuottavuutta parannettua. Gupta nostaa lisäksi esiin kuusi perustelua sille, miksi tekoälytyökalut kuten GPT-4 eivät voi korvata ihmistä ohjelmistokehitystyössä. Tekoälytyökaluilla ei ole tarpeeksi alla mainittuja ominaisuuksia, joita taas ohjelmoijilla ja kehittäjillä on:

1) alan käytäntöjen tuntemus,

2) asiantuntemusta monimutkaisten/loogisten ongelmien tunnistamiseen ja ratkaisemiseen,
3) kykyä ymmärtää laajempi konteksti, projektin vaatimukset ja käyttäjien odotukset, jolloin tehdään tietoon perustuvia päätöksiä koodia kirjoittaessa,
4) tietoa ja kokemusta, jota tarvitaan sovellusten ylläpitoon ja pitkäaikaiseen tukeen,
5) kykyä tehdä yhteistyötä, ideointia ja ongelmanratkaisua tiiminä osallistuen keskusteluihin, jakaen näkemyksiä sekä tarjoten luovia ratkaisuja,
6) luovuutta ja innovatiivisuutta, jota tarvitaan uraauurtaviin ratkaisuihin tai täysin uusien algoritmien tai lähestymistapojen kehittämiseen.

Todettakoon lopuksi, että ymmärtääksemme tekoälyn vaikutukset lähitulevaisuuden työelämään on siihen suhtauduttava vastuullisesti (Taulli 2023).

3.2 Eettiset haasteet tekoälyn ja ihmisen yhteistyössä

Tekoälyn eettisyyteen liittyviä huolia ovat mm. turvallisuusuhat ja väärinkäyttö. Kaikki eivät näistä syistä ole vain innoissaan tekoälyn ja sen ”älykkyyden” tai jopa taidon suhteen. Kyseessä on kuitenkin suhteellisen uusi ja iso askel ihmiselle (jo tehdyn ”digiloikan” jatkeeksi), että välillä tekoälyn toimintaa ja sen tekemiä johtopäätöksiä (algoritmin tuottamaa vastausta) on ihmisen vaikea ymmärtää. Kun tekoälyn käyttämä algoritmi on hiottu todella tarkaksi monine parametreineen ja datan määrä on suuri, tekoäly pystyy tosiaankin työstämään kaikkea annettua dataa mallikkaasti, ihmistä tehokkaammin ja laskemaan todennäköisyyksiä ja päätelmiä, joihin ihminen, ei ainakaan yhtä nopeasti, pystyisi. Tämän seurauksena syntyy tilanne, jossa ihminen ei ihan pysty ymmärtämään miten kyseiseen tulokseen on päästy. Tällaisessa tapauksessa tekoälyn käytöstä ei voi olla ihmiselle hyötyä. Jotta ihminen voi hyödyntää tekoälyn tuottamaa sisältöä, on ihmisen kuitenkin oltava tietoinen siitä mitä tekoälyn on pyydetty tekevän. Ihmisen on siis valvottava konetta, vaikka monet rutiininomaiset tehtävät kone pystyykin hoitamaan mallikkaasti ja tarkasti ilman ihmisen aktiivista tarkastusta. (Kananen, Puolitaival 2019: 202, 210–211, 222).

Tekoälyn kuuluisi olla vastuullista. On yhdistettävä teknologista ymmärrystä, oikeudellista näkemystä sekä eettistä pohdintaa, jotta voimme hyödyntää tekoälyn tuomia mahdollisuuksia samalla varmistaen sen vastuullisen ja eettisen käytön (Salo 2023: 151–152). Yksityisyydensuoja ja datanhallinta nousevat esiin huolenaiheina, sillä tekoäly tarvitsee toimiakseen hyvin valtavan määrän dataa. Tarvitaan siis uusia toimintamalleja, jotta varmistutaan siitä, että nämä toteutuvat myös tekoälyn käytössä. (Salo 2023: 152–153). Muita esiin nousevia haasteita tekoälyn käytössä ovat tasaarvon, monimuotoisuuden ja syrjimättömyyden toteutuminen. Tekoälyn tulee olla kaikkien saatavilla, ja palvella koko yhteiskuntaa. Algoritmeja tulisi myös valvoa, jotta käytettävään dataan ei päädy ennakkoluuloja lisäävää aineistoa, jotka voisivat johtaa syrjintään. (Salo 2023: 153–155).

Salo (2023: 155) huomauttaa, että tekoälyä voi ja pitääkin käyttää työkaluna tasa-arvon, monimuotoisuuden ja syrjimättömyyden edistämiseen niin yksilön tasolla kuin koko yhteiskunnan tasolla.

Ohjelmistokehitystekoälytyökalun käytön kuten ChatGPT:n uhkiin voivat lukeutua tietoturva sekä eettiset kysymykset. Esimerkiksi liiallinen riippuvuus tai kriittisen tarkastelun puute voi johtaa ihmisen tahattomaan valvontaan. Lisäksi, mikäli tekoälyjärjestelmiä ei valvota, voisivat ne pahimmassa tapauksessa vahvistaa jo olemassa olevia ennakkoluuloja, eikä sen sijaan olla avoimia ja oikeudenmukaisia kuten niiden pitäisi. Mitä tekoälyn generoimaan koodiin tulee, tulevaisuudessa olisi syytä sisällyttää esim. chatGPT:n uusiin versioihin myös eettisiä ohjeistuksia, joilla taataan yksityisyyden suoja, tietosuoja sekä oikeudenmukaisuus koodissa. (Rahmaniar 2023: 2, 5).

Euroopan Unionin laatiman julkaisun perusteella vastuullisen tekoälyn käytössä tulee ottaa laajemmin huomioon perusoikeudet kuten yksityisyys, tietosuoja, syrjimättömyys ja oikeussuoja. Jotta tekoälyn käyttöön liittyviä eettisiä kysymyksiä voidaan valvoa tehokkaammin ja varmistaa yleinen vastuullisuus, on nykyisten valvontaelinten asiantuntemusta vahvistettava. Myös tekoälyn teknologioihin erikoistuneet organisaatiot voivat lisätä vastuullisuutta tekoälyjärjestelmien käyttöön. On esimerkiksi varmistettava, että tekoälyyn perustuvat päätökset eivät ole syrjiviä. Automatisoidussa tekoälyyn perustuvassa päätöksenteossa tulisi huomioida palvelun läpinäkyvyys, eli selventää miten tekoälyä käytetään päätöksenteossa sekä soveltaa tietosuojasääntöjä. (European Union Agency for Fundamental Rights 2021: 5, 8–12).

On myös tapauksia, joissa tekoälyn käyttö on eettisistä syistä kielletty. Esimerkiksi Italian tietosuojaviranomaiset olivat huolissaan OpenAI mallin yksityisyyden suojasta sekä oliko se GDPR:n (*General Data Protection Regulation*) eli yleisen tietosuoja-asetuksen mukainen. Huolia nousi mm. henkilötietojen keräämisestä, säilytyksestä ja käyttämisestä tekoälyn kouluttamiseen sekä alaikäisten palvelun käytöstä. Näiden esiin nousseiden huolien seurauksena chatGPT:n käyttö kiellettiin Italiassa keväällä 2023. Muita maita, joissa chatGPT:n käyttö on kielletty ovat mm. Kiina, Iran, Pohjois-Korea ja Venäjä. (McCallum 2023).

Taulli (2023) nostaa esiin Joey Pitrikin listaamia tekoälyn mahdollisia uhkia yhteiskunnalle. Ensinnäkin kansallinen turvallisuus ja demokratia voivat olla vaarassa, kun poliittisia syväväärennöksiä (deepfake) ja muuta generatiivista tekoälyä käytetään väärän tiedon levittämiseen ja turvallisuuden horjuttamiseen.

Toiseksi, identiteettivarkaudet ja petokset ovat uusi uhka, sillä generatiivinen tekoäly mahdollistaa syväväärennösten käytön tilien haltuunotossa ja henkilöllisyyksien väärinkäytössä.

Kolmanneksi, yksityisyys on vaakalaudalla, kun syväväärennöksissä käytetään ihmisten kuvia ja videoita ilman heidän suostumustaan.

Muut Taullin (2023) mainitsemat uhat kohdistuvat kyberturvallisuuteen, esimerkiksi tietojen kalastausviestien lisääntyminen sekä haittaohjelmien koodaaminen. Se, onko tekoälyn haitallinen käyttö mahdollista estää, jää nähtäväksi. Toki yksi mahdollisuus on säädellä sen käyttöä ja käytänteitä, joita jo aiemmin käsiteltiin EU:n laatiman julkaisun avulla. Ongelma säädösten laatimisessa on työn hitaus, kun vastaavasti taas teknologian kehitys on nopeaa (Taulli 2023).

4 Tutkimus

Tietoperustaan nojautuen pyrin työn empiirisessä osassa saamaan vastauksia tutkimuskysymyksiin, jotka ovat siis:

- miten tekoälytyökaluja käytetään ohjelmistokehitystyössä, sekä mitä hyötyjä niiden käytöstä on?
- miten tekoälytyökalujen käyttö vaikuttaa työtehtäviin? kuinka ne voivat tehostaa työntekoa?
- mitä eettisiä haasteita tai riskejä tekoälyn käyttöön liittyy? voiko tekoäly korvata koodaajan?

4.1 Kuvaileva kirjallisuuskatsaus

Tutkimusmetodina on kuvaileva kirjallisuuskatsaus. Kirjallisuuskatsauksen tutkimuskohteena on tutkijoiden alkuperäistutkimukset. Tutkimusmetodin avulla voidaan tunnistaa, arvioida, tulkita ja yhdistää olemassa olevaa tietoa. Tavoitteena on täten tiivistää alkuperäistutkimusten olemassa oleva tieto ja tehdä johtopäätökset asetettuihin tutkimuskysymyksiin. Kuvaileva (*narratiivinen*) kirjallisuuskatsauksen avulla voidaan jäsentää ja järjestää jo olemassa olevaa tietoa johdonmukaisesti ja pystytään saamaan kattava kuva aiheesta. Kuvailevan kirjallisuuskatsauksen tavoitteena on aiheen ymmärtäminen, sen esiin tuominen argumentoiden ja hyvin jäsennellysti. Aiheen ymmärtäminen voi tarkoittaa sen tunnistamista, vahvistamista tai kyseenalaistamista, ja lisäksi voidaan tuoda esille aiempien tutkimusten nostamia kysymyksiä. Kuvaileva kirjallisuuskatsaus antaa lukijalle mahdollisuuden syventää tietoaan vähitellen tutkimusta lukiessa ja eri aihealueisiin tutustumalla. Kuvailevassa kirjallisuuskatsauksessa tutkimusaineiston valinta on vapaampaa. Tätä ominaisuutta on myös kritisoitu subjektiivisuudestaan, mutta toisaalta sen vahvuutena pidetään tutkijan mahdollisuutta keskittyä juuri hänen tutkimuksensa tärkeisiin kysymyksiin ja aineistolähtöisesti syventyä niihin. Tutkija pystyy sitten analysoimansa aineiston pohjalta tekemään havaintoja ja oivalluksia. Kuvailevissa kirjallisuuskatsauksissa voidaan vielä tehdä jako *kartoittavan katsauksen* ja *scoping-katsauksen* välille. Kartoittavan katsauksen tavoitteena on mm. muodostaa kokonaiskäsitys tutkimuksissa olevista käsitteistä ja aiheen teorioista. Scoping-katsauksessa pyritään saamaan kuva aiheen laajuudesta tarkastelemalla tutkimusten määrää, laatua ja luonnetta. (Vilkka 2023).

4.2 Menetelmän käyttö ja tutkimuksen aineisto

Tutkimuksessani en pyri luomaan yleiskatsausta aiheen laajuudesta tutkimalla kaikkia mahdollisia tutkimuksia, joita aiheesta on olemassa. Sen sijaan pyrin ymmärtämään tutkimuksissa olevaa tietoa ja sitä yhdistelemällä ja jäsentämällä antamaan vastauksia asettamiini tutkimuskysymyksiin. Tutkimusaineiston valinnassa korostui tutkimusten näkökulman ja asetelmien sopivuus minun tutkimukseni tueksi. Tutkimusaineistoksi valikoitui kuusi vertaisarvioitua artikkelia. Alla dokumentoituna tutkimusaineiston hakuprosessi ja valinta.

Haulla Haaga-Helian Finnasta: ”generative AI software development” vertaisarvioituja artikkeleita, joissa esiintyy koko hakuteksti, löytyi 8914 kappaletta. 2023 eteenpäin julkaistuja artikkeleita 3817 kappaletta. Kun lisää hakutekstiin ”chatGPT” tuloksia tulee 845 kappaletta. Koska tulokset eivät ole tarpeeksi täsmällisiä vaan artikkelit käsittelevät hyvin laajasti ja eri näkökulmista chatGPT:n käyttöä eri aloilla, hakutekstiä oli muutettava hieman: ”generative AI programming chatGPT”. Tällä kertaa tuloksia tuli kohtuullisempi määrä: 416 kappaletta, joista löytyi kiinnostava chatGPT:n tehokkuutta tutkiva artikkeli ”*Effectiveness of ChatGPT in Coding: A Comparative Analysis of Popular Large Language Models*”. Samaa hakutekstiä hyödyntäen etsin seuraavaksi ”generative AI programming copilot”, ja tuloksia sain vain 50 kappaletta. Mutta koska mikään näistä ei vaikuttanut otsikkotasolla tutkivan nimenomaan GitHub Copilotia (vaan useimmat käsitteli chatGPT:n käyttöä) päätin vielä muokata hakutekstiä: ”AI coding assistant copilot”, jolloin tuloksia tuli vain 32 kappaletta, joista ensimmäinen artikkeli tuntui sopivan tutkimusnäkökulmaani ”*Grounded Copilot: How Programmers Interact with Code-Generating Models*”. Lisäksi löytyi myös kolmas artikkeli, joka yleisellä tasolla tutkii ohjelmistokehityksessä käytettävien tekoälytyökalujen oikeellisuutta, tehokkuutta ja ylläpidettävyyttä verrattuna ihmiseen ”*Program Code Generation with Generative AIs*”.

Hakutekstillä ”AI coding assistant tabnine” tuli kaikista vähiten tuloksia, vain 4 kappaletta. Päätin hakea seuraavaksi pelkällä ”tabnine” hakutekstilä, jolloin tuloksia tuli 9 kappaletta. Yksi tulos tuli toistamiseen ja päätin valita kyseisen artikkelin ”*Evaluating the Usability and Functionality of Intelligent Source Code Completion Assistants: A Comprehensive Review*”, jossa arvioidaan älykkäiden lähdekoodin täydennysavustajien käytettävyyttä ja toimivuutta, neljänneksi tutkittavaksi.

Loput artikkelit tulisi valita aiheista, jotka tutkivat tekoälytyökalujen käyttöä tulevaisuudessa, kuinka ne vaikuttavat ohjelmistokehittäjien työnkuvaan sekä mitä eettisiä haasteita tekoälytyökalujen käytöllä voi olla nyt ja tästä eteenpäin. Hakutekstillä ”AI programmin future” tuli aivan liikaa tuloksia, jopa 10171 kappaletta, joten oli selvää, että hakua tuli tarkentaa. Hakutekstillä ”AI programming software developer future” tuli vähemmän, vaikka silti monia tuloksia, 1409 kappaletta. Vaikka artikkelissa ”*“Will I be replaced?” Assessing ChatGPT's effect on software development and programmer perceptions of AI tools*” näkökulma on chatGPT:ssä, uskon minkä tahansa tekoälytyökalun vaikuttavan ohjelmistokehittäjän työnkuvaan melko samankaltaisesti, joten päätin valita tämän viidenneksi artikkeliksi.

Viimeisen artikkelin tulisi siis tarkastella tekoälytyökaluihin liittyviä haasteita ja eettisiä ongelmia. Hakutekstillä ”AI programming challenges ethics” ei tullut kovin relevantteja tuloksia (1322 kappaletta), joten hakutekstiä tuli jälleen optimoida. Hakutekstillä ”AI coding assistant software development challenges” tuloksia tuli 570 kappaletta, eikä löytynyt sopivasta näkökulmasta aihetta tutkivaa artikkelia. Lopulta viimeinen, eli kuudes artikkeli ”*Ethical Challenges in the Development of Virtual*

Assistants Powered by Large Language Models” löytyi hakutekstillä ”AI ethical challenges software development”.

Ohessa vielä valittu tutkimusaineisto:

1) Coello, C. E. A., Alimam, M. N., & Kouatly, R. (2024). Effectiveness of ChatGPT in Coding: A Comparative Analysis of Popular Large Language Models. Digital, 4(1), 114-125. Luettavissa: https://doi.org/10.3390/digital4010005
2) Barke, S., James, M. B., & Polikarpova, N. (2023). Grounded Copilot: How Programmers Interact with Code-Generating Models. *Proceedings of ACM on programming languages, 7*(OOPSLA1), 85-111. Luettavissa: https://doi.org/10.1145/3586030
3) Idrisov, B., & Schlippe, T. (2024). Program Code Generation with Generative Ais. *Algorithms, 17*(2), 62. https://doi.org/10.3390/a17020062
4) Hliš, T., Četina, L., Beranič, T., & Pavlič, L. (2023). Evaluating the Usability and Functionality of Intelligent Source Code Completion Assistants: A Comprehensive Review. *Applied sciences, 13*(24), 13061. Luettavissa: https://doi.org/10.3390/app132413061
5) Kuhail, M. A., Mathew, S. S., Khalil, A., Berengueres, J., & Shah, S. J. H. (2024). “Will I be replaced?” Assessing ChatGPT's effect on software development and programmer perceptions of AI tools. *Science of computer programming, 235*, . Luettavissa: https://doi.org/10.1016/j.scico.2024.103111
6) Piñeiro-Martín, A., García-Mateo, C., Docío-Fernández, L. & López-Pérez, M. d. C. (2023). Ethical Challenges in the Development of Virtual Assistants Powered by Large Language Models. Electronics (Basel), 12(14), 3170. Luettavissa: https://doi.org/10.3390/electronics12143170.

4.3 Tulokset

Tässä osiossa tarkastelen löydöksiä aihealueittain aloittaen esiin nousseista havainnoista ohjelmoinnissa käytettyjen tekoälytyökalujen käytön suhteen sekä löydöksistä liittyen tekoälytyökalujen tehokkuuteen ja tuotetun koodin laatuun. Tämän jälkeen tarkastelussa tekoälytyökalujen vaikutukset ohjelmoinnin työprosesseihin sekä tulevaisuuden työnäkymiin. Lopuksi katsaus tekoälyyn liittyviin eettisiin haasteisiin niin työelämässä kuin yhteiskunnassa.

4.3.1 Havaintoja tekoälytyökalujen käytöstä ja tehokkuudesta

Coello, Alimam & Kouatly:n (2024) tutkimuksessa tutkittiin viiden LLM:n (Bard, Bing, Claude LLMs, ChatGPT-3.5 ja ChatGPT-4) toimintaa ja tehokkuutta 460:n keskenään erilaisten ohjelmisto-ongelmien ratkaisussa Python-kielellä. Ongelmat olivat luonteeltaan ja vaikeustasoltaan sellaisia, jotka lähtötason ohjelmoija osaisi ratkaista.

Tutkimuksessa haluttiin tutkia selainpohjaisten versioiden toimivuutta ja luotettavuutta, koska niitä pääsääntöisesti ohjelmistokehittäjät käyttävät helpon ja ilmaisen käytettävyyden vuoksi. LLM:lle annettiin prompti, jonka perusteella se generoi vastauksensa ja ratkaisunsa ongelmaan. LLM:lle annettiin palautetta, jonka perusteella se pystyi korjaamaan ratkaisuehdotustaan tarpeeksi, jotta se läpäisisi ihmisen ohjaamat testit. LLM:t saivat kuitenkin ehdottaa enintään 10 kertaa ratkaisua, jonka jälkeen testi merkittiin epäonnistuneeksi. Vain läpäistyistä testeistä sai pisteen. Parhaiten tutkimuksessa suoriutuivat ChatGPT-4 (1072 pistettä eli 87,51 % oli oikeita ratkaisuja) sekä ChatGPT-3.5 (1019 pistettä eli 83,18 %) (Coello, Alimam & Kouatly, 119–120).

Kun tutkimuksessa verrattiin koodin laatua, parhaimmat pisteet sai Bing, sillä LOC *(lines of code* eli koodirivien määrä) oli pienin, pärjäten siis niukasti paremmin kuin GPT-4 (Coello, Alimam, Kouatly, 120–121). Tutkimuksessa paneuduttiin myös LLM:n kykyyn reagoida saamaansa palautteeseen ihmiseltä. Palautteella pyrittiin selittämään LLM:lle tehdyt virheet ja kuinka ne voisi korjata. GPT-4 ei ollut läpäissyt 16 ongelmaa, ja saadun palautteen avulla se onnistui läpäisemään testit peräti 14 osalta. Voidaan siis todeta, että chatGPT onnistui annetun palautteen perusteella optimoimaan ja korjaamaan koodia riittävästi, jotta lopulta onnistui läpäisemään testin virheittä. On siis vahvaa näyttöä siitä, että chatGPT kykenee kehittämään vastaustaan yhä uudelleen, jotta päästään halutuun lopputulemaan. Toisaalta on todettava myös, että chatGPT:n kuten muiden LLM:ien käyttö on vahvasti sidottuna ihmisen panokseen. LLM saattoi tutkimuksessa antaa samaan ongelmaan kaksi eri vaihtoehtoa, mikäli promptin sanamuotoa muutti hieman. On ihmisen vastuulla ja velvollisuutena tietää, kuinka muotoilla promptit niin, että LLM osaa ehdottaa ko. ongelmaan oikeaa ratkaisua, sekä lopuksi myös varmistettava LLM:n tuottaman koodin toimivuus sekä soveltuvuus annetussa kontekstissa. Näihin toimenpiteisiin on ohjelmoijan varattava työpäivästään aikaa ja resursseja, vaikka muutoin säästäisi aikaa koodin tuottamisessa. (Coello, Alimam & Kouatly, 122–123). Koodin tuottamisessa ChatGPT-4 ylsi tutkimuksessa kokonaisuudessaan 86,23 % onnistumiseen. Muut (poisluettuna chatGPT) suoriutuivat heikommin ja vastauksissa oli mm. enemmän virheitä tai ne eivät kyenneet korjaamaan koodia lainkaan tai tarpeeksi tehokkaasti annetun palautteen perusteella. Tämä osoittaa sen, että LLM:n generoima koodi vaatii ihmisen tarkistuksen ennen sen otta-

mista käyttöön. Kuten aiemmin jo mainittiin, kyseessä on työkalu, joka tarvitsee toimiakseen kuitenkin ohjelmoijalta ohjausta promptien muodossa sekä tarkistusta, eikä voida täten lainkaan väittää, että se voisi korvata ihmisen työpanoksen. (Coello, Alimam & Kouatly, 123–124).

Idrisov & Schlippe:n (2024) tutkimuksessa tutkitaan seitsemän tekoälytyökalun käyttöä ohjelmistokehityksessä. Työssäni tarkastelen pääasiassa tuloksia chatGPT:n ja Copilotin osalta. Tutkimuksessa tekoälytyökalujen tuli ratkaista kuusi algoritmeihin liittyvää ohjelmointiongelmaa: 2 helppoa, 2 keskivaikeaa ja 2 haastavaa. Ongelma tuli ratkaista kolmella kielellä: Java, Python ja C++. Tekoälyn vastausten arvioinnissa mitattiin tehokkuutta sekä ylläpidettävyyttä ja näitä verrattiin ihmisohjelmoijan luomiin ratkaisuihin, jotka löytyivät LeetCode yhteisöstä. (Idrisov, Schlippe, 5–6) Tutkimuksen tavoitteena oli myös luoda kokonaiskuva tekoälyn tuottaman ratkaisun laadusta. Tämän saavuttamiseksi mitattiin mm. koodin pituutta (LOC) ja kompleksisuutta (*cyclomatic complexity*, lineaarisesti riippumattomien polkujen määrä ohjelmakoodissa), sekä kuinka paljon algoritmi tarvitsi aikaa ja tilaa, suoritusaikaa (millisekunneissa) ja muistinkäyttöä (MB), sekä koodin ylläpidettävyyttä. Lisäksi tutkimuksessa tutkittiin promptien vaikutusta tekoälyn tarjoamiin ratkaisuihin. Prompteja iteroitiin, eli muokattiin useita kertoja tekoälyltä saadun koodin perusteella, jotta päästäisiin toivottuun lopputulokseen. Prompti annettiin muodossa ”Ratkaise seuraava ongelma <ohjelmointikielen> koodilla. <määrittelyt>”, yhdessä alkuperäisen tehtävän kuvauksen, esimerkkien, tarkennusten ja rajoitusten kanssa. Jos alkuperäinen prompti/ohje ei tuottanut oikeaa tai suoritettavaa koodia, hiottiin sitä iteratiivisesti poistamalla mm. esimerkkejä. (Idrisov & Schlippe, 7–8).

Tutkimuksessa analysoitiin ratkaisuja, jotka onnistuivat ratkaisemaan ohjelmointitehtävät oikein. Voitiin todeta, että parhaiten suoriutui Copilot 9 kokonaispistemäärällä ja onnistui ratkaisemaan täten 50 % tehtävistä. ChatGPT on jaetulla toisella sijalla ratkaisten vain 22 % ongelmista. Mielenkiintoista on huomata, että ChatGPT onnistui kuitenkin ratkaisemaan yhden haastavan ongelman Pythonilla, Copilot taas ei ratkaissut haastavia tehtäviä lainkaan mutta sen sijaan keskivaikean tason ongelmista se ratkaisi kaksi Javalla ja kaksi C++:lla. Ihmisohjelmoija onnistui ratkaisemaan kaikki tehtävät. (Idrisov & Schlippe, 10).

Kun näitä oikein ratkaistuja koodeja analysoitiin edellä mainittujen mittareiden valossa, pystyttiin toteamaan, että kahdeksassa tapauksessa tekoäly tuotti vähemmän koodirivejä kuin ihminen (31 %), mutta 13 tapauksessa (50 %) ihminen taas suoriutui paremmin. GH Copilot (sekä BingAI) suoriutui parhaiten tuottamalla kolme ohjelmakoodia vähemmällä koodirivimäärällä kuin ihminen. (Idrisov, Schlippe, 10). ChatGPT ratkaisi haastavan tason ongelman vain 10 koodirivillä, mikä on 60 % vähemmän kuin ihmisellä. Tekoälyn ratkaisujen kompleksisuus oli 27 % tapauksista vähäisem-

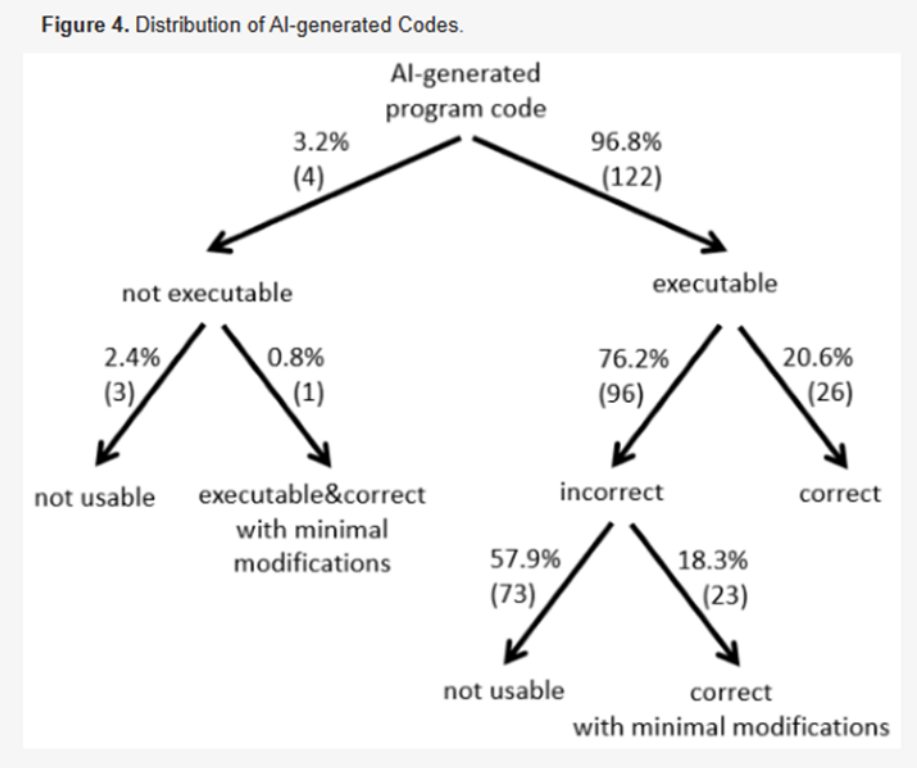
pää kuin ohjelmoijalla, eli koodin laatu oli parempaa. GH Copilot suoriutui paremmin kuin muut tekoälytyökalut, tuottaen kolme koodia, joissa oli vähemmän kompleksisuutta kuin ihmisellä. ChatGPT:n generoimassa koodissa Pythonilla oli jopa 50 % vähemmän kompleksisuutta. (Idrisov & Schlippe, 11).

Useimmissa tapauksissa (50 %) tekoälyn koodi ei pärjännyt ihmisen koodille aikaa arvioidessa. Tilan käytön suhteen tekoälytyökalut toimivat paremmin ja Copilot tuotti kolme koodia paremmalla tilalla kuin ihminen. (Idrisov, Schlippe, 12). Tietyissä tapauksissa (23 %) tekoäly tuotti koodia, joka suoritettiin nopeammin kuin ihmisen kirjoittama. Suurimmassa osassa tapauksista (65 %) tekoäly jäi kuitenkin jälkeen suorituskyvyssä. (Idrisov & Schlippe, 13).

Viidessä tapauksessa (19 %) generatiivinen tekoäly pystyi ratkaisemaan ohjelmointiongelman koodilla, joka suoritettiin vähemmällä muistinkäytöllä kuin ihmisen mutta 42 % kerroista tulos oli päinvastainen, eli ihmisohjelmoijan koodi käytti vähemmän muistia. Kolmessatoista tapauksessa (50 %) generatiivinen tekoäly pystyi tuottamaan koodia, jolla oli korkeampi ylläpidettävyysindeksi kuin ihmisellä, mikä vastaavasti tarkoittaa sitä, että puolet kerroista ihmisen tuottama koodi oli parempaa. (Idrisov & Schlippe, 13–14).

Tutkimuksessa tutkittiin lopuksi myös toimimattomien ratkaisujen joukosta niitä, jotka olisi mahdollista saada toimimaan pienellä korjauksella. Tutkimuksessa analysoitiin koodin korjaukseen kuluvaa aikaa (TTC *time to correct*). Analyysissä päästään lopputulokseen, että aikaa voidaan säästää 8,85 prosentista peräti 71,31 prosenttiin, jos tekoälyn generoima ohjelmakoodi korjataan eikä kirjoiteta alusta asti. (Idrisov & Schlippe, 14–15).

Lopuksi vielä tiivistettynä puukaavion muodossa tekoälytyökalujen suorituskyky tutkimuksessa (Kuva 5).



Kuva 5. Tekoälytyökalujen suorituskyky (Idrisov & Schlippe, 16)

Kuvan 5 perusteella 96,8 % tekoälyntuottamista ratkaisuista oli suoritettavia, ja näistä 20,6 % tuotti oikean vastauksen mutta 76,2 % ei toiminut. 18,3 % näistä toimimattomista koodeista pystyttiin pienillä korjauksilla saamaan toimimaan oikein. Voidaan siis todeta, että generatiiviset tekoälytyökalut suoriutuvat vaihtelevasti ohjelmointikielestä ja koodausongelmasta riippuen. Tutkimus todistaa myös sen, ettemme voi vielä täysin luottaa siihen, että tekoäly tuottaisi aina oikeaa, tehokasta ja ylläpidettävää ohjelmakoodia. Tekoäly voi kuitenkin nopeuttaa ohjelmointia, koska vaikka ohjelmakoodi olisi toimimatonta, se voidaan saada korjattua pienellä vaivalla ja lyhyessä ajassa. (Idrisov & Schlippe, 16–17).

Hliš, Četina, Beranič, & Pavlič:n (2023) tutkimus toteutettiin kolmivaiheisena, ensin tutkijat tekivät systemaattisen kirjallisuuskatsauksen[1] selvittääkseen saatavilla olevat älykkäät avustajat lähdekoodin täydennyksessä, niiden tärkeimmät toiminnot, sekä miten nämä toiminnot poikkesivat

[1] Tutkimuskysymykseen vastaamiseksi pyritään löytämään kattavalla hakuprosessilla tarkoituksenmukaisimmat ja laadukkaimmat alkuperäistutkimukset. Ennalta määritellyillä ja täsmällisillä tutkimuskysymyksillä sekä järjestelmällisillä menettelytavoilla haetaan puolueettomia, yleistettäviä tuloksia. (Vilkka, H. 2023. Kirjallisuuskatsaus metodina, opinnäytetyön osana ja tekstilajina. Helsinki: Art House. luku 1.2.3)

perinteisistä avustajista. Tämän jälkeen ohjelmoijia pyydettiin vastaamaan kyselyyn, jolla pyrittiin selvittämään suhtautumista älytyökaluihin ja niiden tarpeellisuutta. Kyselystä saatua dataa käytettiin kolmannessa vaiheessa tulkintojen tekemiseen. (Hliš, Četina, Beranič, & Pavlič, 3–6)

Jotta pystyisin vastaamaan tutkimuskysymyksiini, haluan omassa työssäni keskittyä kyselyyn ja sen tulosten tulkintaan. En paneudu tutkijoiden systemaattisen kirjallisuuskatsauksen toteutuksen kuvaamiseen. Tutkimuskysymykseni kannalta oleelliset asiat liittyvät tekoälytyökalujen käyttöön, en siis myöskään käy läpi eroavaisuuksia perinteisten avustajien (mm. listaehdotus kaikista saatavilla olevista vaihtoehdoista kontekstista riippumatta) ja älykkäiden avustajien välillä.

Tutkimuksessa analysoitiin kahdeksan ohjelmistokehityksen tekoälytyökalua ja niiden käyttötarkoituksia: GitHub Copilot, Tabnine, Kite, IntelliCode, IntelliCode Compose, CACHECA, Pythia ja Galois. Työni kannalta oleelliset tekoalytyökalut ovat Copilot ja Tabnine, joten tarkastelen vain niiden osalta esiin nousseita ominaisuuksia.

Kuvassa 6 on ote taulukosta, jossa verrataan kahdeksaa työkalun käyttöominaisuutta. Olen tarkoituksella rajannut tähän vain tutkimieni Copilotin ja Tabninen ominaisuudet.

Table 4. The accessible intelligent assistants.

		GitHub Copilot	Tabnine
Current word completion		✓	✓
Generating a program continuation:	Word	✓	✓
	Line	✓	✓
	Section of source code	✓	
API exploration			✓
API proposal		✓	✓
Naming variables		✓	
Source code generation based on natural language		✓	✓
Learning the model on your own source code (for enterprises)		✓	✓

Kuva 6. GitHub Copilot ja Tabnine käyttöominaisuuksien vertailu. (Hliš, Četina, Beranič, & Pavlič, 10)

Kuten kuvasta 6 näemme Copilot ja Tabnine toiminnallisuudet ovat pitkälti samat, eroja nousee kuitenkin muutaman toiminnallisuuden osalta. Copilotilla pystytään tekemään miltei kaikki listatut

toiminnot, kuitenkin API:en (*application programming interface,* suomeksi sovellusohjelmointirajapinta) etsiminen ja hyödyntäminen ei onnistunut. Tabnine kykeni tutkimaan APE:ja mutta sillä oli rajoitteita muiden toimintojen kanssa: se ei kykene generoimaan kokonaista koodin osaa tai nimeämään muuttujia. (Hliš, Četina, Beranič, & Pavlič, 10, 14).

Älykkäät lähdekoodin täydennystyökalut ovat merkittävästi parantaneet ohjelmistokehitystä. Silti niiden käyttöön liittyy haasteita, jotka vaikuttavat myös niiden tehokkuuteen. Yksi merkittävä haaste on automaattisten testausmenetelmien sovittaminen tekoälytyökalun ehdotusten kanssa. Haastavaa on myös tekoälytyökalun generoiman koodin muodollinen varmennus ja validointi. Nykyiset tekoälytyökalut eivät myöskään täysin kykene ymmärtämään tai ennustamaan kehittäjän aikomuksia. Edellä mainituista syistä älykkään lähdekoodin täydennyksessä tarvitaan yhä jatkuvaa tutkimusta ja kehitystä. (Hliš, Četina, Beranič, & Pavlič, 11).

Kyselyssä älykkäiden avustajien käyttökelpoisuus arvioitiin kohtalaiseksi, ja käyttäjäkokemus arvioitiin keskiverroksi. Mielenkiintoista oli, että työkalujen käytöstä pidettiin (sai korkeamman arvosanan) vaikka aina niiden käytön ei koettu tehostavan työntekoa (käytännöllinen näkökulma sai alemman arvosanan). 82 % vastaajista ilmoitti jatkavansa älykkäiden avustajien käyttöä. Kokeneet ohjelmistokehittäjät kokivat avustajan käytännöllisemmäksi ja pitivät sen käyttämisestä. Kokemattomammat käyttäjät, jotka luottavat enemmän tekoälyyn ja sen ehdotuksiin, saattavat hämmentyä epärelevanttien ehdotusten keskellä. Työkalujen käyttökokemukseen vaikutti siis olennaisesti ehdotusten soveltuvuus annetussa kontekstissa. Työkalujen tehokkuudessa on parannettavaa, erityisesti ehdotusten käyttökelpoisuudessa. Huomattiin myös, että yleisesti ottaen kehittäjät, joilla oli ennestään kokemusta näistä työkaluista arvioivat ne positiivisesti, kun taas henkilöt, joilla vähemmän kokemusta suhtautuivat tekoälytyökaluihin kriittisemmin. (Hliš, Četina, Beranič, & Pavlič, 11–14).

Vaikka tekoälytyökalut eivät tutkimuksen mukaan merkittävästi paranna olemassa olevan koodin laatua tai vähennä virheitä, ne voivat nopeuttaa koodin kirjoittamista ymmärtämällä entistä paremmin kehittäjän kontekstia ja tarjoamaan relevantteja koodin täydennyksiä. Käyttäjäkokemuksen suhteen korostui tarve käytännölliselle käyttöliittymälle. (Hliš, Četina, Beranič, & Pavlič, 16).

4.3.2 Tekoälytyökalujen vaikutus ohjelmointiin, työprosesseihin ja työllisyyteen

Barke, James & Polikarpova:n (2023) tutkimuksessa tutkitaan GitHub Copilotin käyttöä ja pyritään saamaan kattava käsitys ohjelmoijien vuorovaikutuksesta Copilotin kanssa. Tutkimukseen valittiin 20 eritaustaista ohjelmoijaa ja heitä pyydettiin toteuttamaan neljä erilaista ohjelmointitehtävää.

Tehtävät oli suunniteltu tutkimaan, miten ohjelmoijat käyttivät Copilotia eri tilanteissa ja eri ohjelmointikielillä (Python, Rust, Haskell, Java). Ohjelmoijista yhdeksällä oli jo ennestään kokemusta Copilotin käytöstä. Copilotiin perehtymiseen annettiin ensin harjoitustehtävä. Tutkimukseen osallistujia kehotettiin käyttämään Copilotia, ja heitä pyydettiin tehtävän jälkeen kertomaan työskentelystään. Grounded Theory -menetelmää (GT) käytettiin teorioiden luomiseen ilman aiempaa teoreettista tietoa. Tiedon keruu ja analyysi vuorottelivat koko prosessin ajan. Tutkimuksen aikana erottui kaksi erilaista tilaa, joissa ohjelmoijat olivat vuorovaikutuksessa Copilotin kanssa: kiihdytystila ja tutkimustila. (Barke, James & Polikarpova, 78:1, 78:4–78:7).

Kiihdytystilassa ohjelmoija käyttää Copilotia suorittaakseen suunnitellut kooditoiminnot nopeasti, käyttämällä Copilotin automaattisesti ehdottamaa ehdotusta koodin kirjoittamisessa. Kiihdytystila auttaa ohjelmoijaa työskentelemään nopeammin. Kiihdytystilassa ohjelmoija yleensä hyväksyy Copilotin ehdotuksen ilman sen suurempaa epäröintiä ja jatkaa jouhevasti työskentelyä. Kiihdytystilassa usein Copilotia käytetään pienempiin koodin osiin/ongelmiin, jolloin Copilotin ehdotukset ovat usein helposti hyväksyttävissä. Mikäli taas Copilotin ehdotus oli liian pitkä ja monirivinen, ohjelmoija herkemmin hylkäsi nämä ehdotukset. Näillä pitkillä ehdotuksilla oli myös häiritsevä vaikutus ohjelmoijan työskentelyyn keskeyttäen flow-tilan. Copilotin ehdotuksen arvioinnissa ohjelmoijat tarkastelivat mm. tiettyjä avainsanoja (funktiokutsuja tai muuttujanimiä) tai rakenteita. Mikäli ehdotuksessa ei ollut odotettuja avainsanoja tai rakenteita ohjelmoijat nopeasti hylkäsivät ehdotukset. Toisaalta he hyväksyivät lähes oikeat ehdotukset, jos olivat parannettavissa pienellä korjauksella. (Barke, James & Polikarpova, 78:7–78:10).

Tutkimustilassa kehittäjä turvautuu Copilotiin suunnitellessaan kooditoimintojaan esim. tuntemattoman syntaksin, oikean API:n etsimisessä tai oikean algoritmin löytämisessä. Tutkimustilaan tutkimukseen osallistuja siirtyi, kun ei ollut tarkkaa tietoa, mistä ja miten aloittaa. Toinen tapa, jolla osallistujat aloittivat tutkimustilassa, oli törmätessään toimimattomaan koodiin, jolloin he tutkivat Copilotin avulla erilaisia tapoja korjatakseen ongelman. Osa osallistujista luotti Copilotiin niin paljon, että antoivat sen usein ohjata ennen kuin yrittivät ratkaista tehtävää ensin itse. Yhden osallistujan kohdalla tämä tarkoitti käytännössä sitä, että tehtävän suorittaminen edistyi itseasiassa hitaammin. Melkein jokainen osallistuja kirjoitti ainakin yhden luonnollisen kielen kommentin käyttääkseen Copilotia. Ohjelmoijat kokivat, että prompteja oli vaivattomampaa kirjoittaa Copilotille luonnollisen kielen kommentteina ja oli paremmin hallittavissa kuin koodinmuodossa kirjoitettuna. He olisivat valmiita lisäämään tarpeeksi yksityiskohtaista tietoa kommentteihin, jotta Copilotilla olisi tarpeeksi kontekstia tuottaakseen varteenotettavia ehdotuksia. (Barke, James & Polikarpova, 78:10–78:11).

Tutkimustilassa osallistujat olivat enemmän valmiita käyttämään aikaa vuorovaikutukseen (kommentit) Copilotin kanssa, sekä Copilotin useiden eri ehdotusten läpikäymiseen, arvioimiseen ja vertaamiseen, toisin kuin kiihdytystilassa. Osallistujat myös saattoivat valita vain osan ehdotuksesta ja ottaa toisen osan toisesta eri ehdotuksesta (*cherry-pick*). Copilotin kanssa työskenneltiin myös tietyn työvaiheen toteuttamiseksi, kun ei ollut tarkkaa tietoa, kuinka toteuttaa se. Luotettavimpana ja toimivimpana pidettiin monesti sitä ehdotusta, jossa toistui sama ominaisuus. Osa osallistujista koki useiden erilaisten ehdotusten vertailun aikaa vievänä ja kuluttavana. (Barke, James & Polikarpova, 78:11–78:13).

Ohjelmoijat tarkistavat Copilotin ehdotukset huolellisemmin tutkimustilassa verrattuna kiihdytystilaan. Tässä tapauksessa koodia mm. tarkasteltiin myös dokumentaatioon nojaten, ja se testattiin. Tutkimustilassa osallistujat olivat halukkaita hyväksymään ehdotuksia muokkaamalla niitä omaan tarkoitukseensa paremmaksi. Sudenkuoppana havaittiin, että virheiden havaitseminen Copilotin generoimissa koodiehdotuksissa oli hankalampaa. (Barke, James & Polikarpova, 78:13–78:15).

Ohjelmoijat vaihtelivat näiden kahden tilan välillä tehtävän eri vaiheissa ja siirtyvät sulavasti tilasta toiseen (Barke, James & Polikarpova, 78:7). Tutkimuksessa ehdotetaan lopuksi vielä tekoälytyökalujen käytön tehostamiseksi hienosäätöä niin ihmisen tuottamaan inputiin, joka syötetään tekoälylle, kuin tekoälyn tuottamaan outputiin. (Barke, James & Polikarpova, 78:20).

Kuhail, Mathew, Khalil, Berengueres & Shah:n (2024) tutkimus toteutettiin kaksiosaisena: ensimmäisessä osiossa arvioitiin ChatGPT:n koodaamisen tehokkuutta, toisessa osioissa kysyttiin ohjelmoijien kokemuksista ja näkemyksestä tekoälytyökalujen käytöstä sekä mahdollisista huolista mm. työllisyyteen liittyen (Kuhail, Mathew, Khalil, Berengueres & Shah, 1).

Tutkimuksessa kiteytetään hyvin, kuinka ohjelmistokehityksen tekoälytyökalujen tehokkuutta on tutkittu: tarkasteltu esim. ChatGPT:n kykyä ratkaista koodausongelmia, toisissa tutkimuksissa on taas verrattu tekoälytyökalun kyvykkyyttä ihmisohjelmoijan suoritukseen. Tämä tutkimus eroaa aiemmista, tai tarkemmin sanottuna, ottaa tutkimukseensa mukaan uusia muuttujia, kuten ongelman monimutkaisuuden ja suosion, vaikutuksena ChatGPT:n onnistumisprosenttiin. (Kuhail, Mathew, Khalil, Berengueres & Shah, 3–4).

Kehittäjien näkemykset tekoälytyökaluista ovat vaihtelevia: jotkut kehittäjät luottavat näihin työkaluihin ja kokevat ne hyödyllisiksi tuottavuuden lisäämisessä ja koodin laadun parantamisessa, toiset suosivat niiden käyttöä yksinkertaisissa tehtävissä, kuten yksikkötestien kirjoittamisessa, mutta eivät monimutkaisemmissa tehtävissä, jotka liittyvät liiketoimintalogiikkaan. On havaittu myös, että tekoälytyökalujen käyttö luo uusia tehtäviä, mikä johtaa työntekijöiden keskittymiseen luovempiin

tehtäviin. Tämä voi puolestaan parantaa tuottavuutta ilman työpaikan korvaamisen riskiä. Nämä tutkimukset ovat tunnistaneet kiinnostavia tekoälytyökalujen käyttöön liittyviä muuttujia kuten hyödyllisyys, rajoitukset, luottamus, tuottavuus ja työpaikan turvallisuus. Yleensä nämä muuttujat ovat yhteydessä toisiinsa. Tässä tutkimuksessa kartoitetaan ohjelmoijien käsityksiä tekoälytyökaluista niiden kyvyistä ja rajoituksista sekä näiden työkalujen aiheuttamista mahdollisista uhista luottamukselle, tuottavuudelle ja työpaikan turvallisuudelle. (Kuhail, Mathew, Khalil, Berengueres & Shah, 4).

Tutkimuksen ensimmäisessä osiossa analysoitiin ChatGPT:n tehokkuutta koodiongelmien avulla. Osiossa oli kolme vaihetta: 1) ongelman valinta: ChatGPT:lle annettin 180 eri vaikeustason koodiongelmaa LeetCode-alustasta myös ongelman yleisyys huomioiden, 2) ongelman lähettäminen sellaisenaan, ilman muutoksia tai ohjeita, LeetCode-alustalta ChatGPT:lle, 3) aineiston keräämisessä tarkasteltiin seuraavia asioita: a) läpäisikö ratkaisu testitapaukset b) ratkaisun suorituskyky verrattuna ratkaisuihin LeetCode:ssa c) ratkaisujen muistinkäyttö verrattuna ratkaisuihin LeetCode:ssa. ChatGPT:n tehokkuutta arvoitiin kolmen tekijän perusteella: onnistumisprosentti koodausongelmien ratkaisemisessa, sekä ajallinen tehokkuus ja muistinkäyttö verrattuna ihmisen ratkaisuihin. (Kuhail, Mathew, Khalil, Berengueres & Shah, 5).

Tutkimuksen toisessa osiossa, eli kyselyssä, analysoidaan 99 ohjelmoijan näkemyksiä tekoälytyökalujen käytöstä arvioiden luottamusta, tuottavuutta sekä työllisyyteen liittyvää uhkaa. (Kuhail, Mathew, Khalil, Berengueres & Shah, 5–7).

Tutkimustulokset osoittavat, että 540:sta mittauksesta 57 % ongelmista läpäisi kaikki testitapaukset, kun taas 39,4 % epäonnistui. Mittauksissa, joissa testattiin yhteensä 180 ongelmaa, 44,4 % läpäisi kaikki testitapaukset, kun taas 51,1 % epäonnistui. ChatGPT:n suoritus oli keskimäärin 58,5–59,1 % parempi kuin ihmisen ratkaisut. ChatGPT:n käyttämä muisti oli alle 69,6–69,7 % ihmisten ratkaisuista. Yleisesti ottaen helppoja ongelmia ratkaistiin useimmiten (86,1 %) kaikilla testitapauksilla, kun taas vaikeampien ongelmien läpäisyprosentti oli alhaisempi. Ongelman yleisyys näyttää olevan yhteydessä onnistumisprosenttiin, mutta ei merkittävissä määrin. Tulosten perusteella ei havaittu merkittäviä eroja ongelmien vaikeustason tai yleisyyden vaikutuksesta aikatehokkuuteen tai muistinkäyttöön. (Kuhail, Mathew, Khalil, Berengueres & Shah, 7–10).

Useat kyselyyn osallistujat käyttivät erilaisia tekoälytyökaluja työssään, joista yleisimpiä olivat ChatGPT 71,7 % ja Copilot 32,3 % ja käyttötaajuus asteikolla 1 (harvoin)–5 (usein) oli 2,08. Tekoälytyökaluja käytetään mm. koodin luomiseen, koodin selittämiseen, virheiden löytämiseen koodista, dokumentoinnin kirjoittamiseen ja koodin suorituskyvyn parantamiseen sekä mm. uusien ratkaisujen löytämiseen. Osallistujien keskimääräinen koettu tuottavuus tekoälytyökalujen avulla oli 2,53 asteikolla 1–5. Osallistujat kokivat, että työkalut tuottavat joskus virheellistä koodia. (Kuhail, Mathew, Khalil, Berengueres & Shah, 10).

Työllisyysuhan osalta 47,5 % ei pitänyt tekoälytyökaluja lainkaan työpaikan uhkana, ja osallistujat kokivat tällä hetkellä keskimäärin vain pienen uhan (1,89 asteikolla 1–5). Tulevaisuuden näkymät aiheuttivat kuitenkin enemmän huolta: 29,3 % jonkinlainen uhka ja 14,1 % kohtalainen uhka. (Kuhail, Mathew, Khalil, Berengueres & Shah, 10).

Vaikka tekoälytyökalut voivat tarjota merkittäviä hyötyjä, niiden käytössä ilmenee yhä puutteita ja rajoitteita, esim. virheellistä koodia, joten ohjelmoijien rooli kriittisenä tarkistajana säilyy erittäin keskeisenä. Tutkimus osoittaa, että yli kaksi kolmasosaa osallistujista ei näe välitöntä työpaikan turvallisuusuhkaa tekoälytyökaluihin liittyen. Kuitenkin tulevaisuuden osalta useammalla herää huoli työllisyydestä. Osallistujien havaittu ero nykyisen ja tulevan työpaikan työllisyysuhan välillä on merkittävä. Kyselyssä nousi esiin huoli siitä, että aloittelevien ohjelmoijien työtehtävät ja työpaikat voitaisiin korvata tekoälytyökaluilla. Toisaalta osallistujat, jotka eivät kokeneet uhkaa työllisyyden puolesta olivat sitä mieltä, että tekoälytyökalut eivät kykene korvaamaan ihmistä, koska tarvitaan ihmisen alakohtaisia tietoa ja luovuutta. Osallistujat, jotka kokivat suurempaa tuottavuutta tekoälytyökalujen kanssa, luottivat näihin työkaluihin enemmän, mutta tunsivat myös enemmän uhkaa työpaikkansa korvautumisesta. (Kuhail, Mathew, Khalil, Berengueres & Shah, 16–17).

Tekoälyn vaikutuksesta työllisyyteen ja työmarkkinoihin puhutaan paljon. On spekuloitu, että tekoälytyökalut korvaisivat erilaisia työpaikkoja, esim. ohjelmistokehittäjien tehtävät voitaisiin automatisoida. Tutkimuksessa korostetaan, että vaikka tekoälytyökalut helpottavatkin ohjelmoijia monissa tehtävissä, ohjelmoijat haluavat pysyä prosessissa vahvasti mukana, eivätkä liioin luota näihin työkaluihin. Lisäksi kokonaisvaikutus työllisyyteen riippuu siitä, miten työtehtävät voidaan tulevaisuudessa määritellä ja jakaa uudelleen. (Kuhail, Mathew, Khalil, Berengueres & Shah, 16).

4.3.3 Tekoälyn käytön eettiset haasteet työelämässä ja yhteiskunnassa

Kun Piñeiro-Martín, García-Mateo, Docío-Fernández & López-Pérez:n (2023) artikkeli julkaistiin heinäkuussa 2023 tekoälytyökalujen (virtual assistants) eettisiksi ongelmiksi ja uhiksi tunnistettiin uskottavat tiedonkalastelut, virheellisen tiedon levittäminen, yksityisyysongelmat, puolueellinen sisältö, plagiointi sekä jo aiemmin tutkimuksessani esiin nostettu työpaikkojen väheneminen. (Piñeiro-Martín, García-Mateo, Docío-Fernández & López-Pérez, 2)

Euroopan Komissio tarjoaa suuntaviivoja tekoälyn eettisesti hyväksyttyjen ratkaisujen suunnittelussa. Euroopan tasolla tulisi kuitenkin pohtia lainsäädäntöä nimenomaan virtuaaliavustajien käyttöön, jotka hyödyntävät LLM-pohjaista ohjelmistoa kuten ChatGPT. (Piñeiro-Martín, García-Mateo, Docío-Fernández & López-Pérez, 2–3).

Artikkelissa luodaan ensin katsaus vuoden 2023 tekoälypohjaisten ratkaisujen säädöksiin Euroopassa. Tämän jälkeen pohditaan LLM-malleilla toimivien virtuaaliavustajien käyttöön liittyviä riskejä

ja oikeudellisia/eettisiä kysymyksiä. Lopuksi annetaan näkemys sekä suosituksia virtuaaliavustajien kehitykseen ja käyttöön.

Euroopan unionin perusoikeuskirja määrittelee kuusi keskeistä eettistä periaatetta, joiden mukaan jokaisen tekoälyjärjestelmän tulisi toimia. Näihin periaatteisiin kuuluu ensinnäkin ihmisen toimijuuden kunnioittaminen. Toinen periaate on yksityisyyden ja tietojen hallinnan kunnioittaminen. Kolmantena periaatteena on oikeudenmukaisuus. Neljäs periaate on yksilön, yhteiskunnallisen ja ympäristöllisen hyvinvoinnin edistäminen. Viides periaate, tekoälyn toiminnan läpinäkyvyys. Lopuksi, tekoälyjärjestelmän tulee olla vastuullinen ja sen toimintaa tulee valvoa. (Piñeiro-Martín, García-Mateo, Docío-Fernández & López-Pérez, 3)

Edellä mainittujen eettisten periaatteiden pohjalta Euroopan komissio on luonut itsearviointilistan (ALTAI), joka auttaa arvioimaan, kuinka hyvin tekoälyjärjestelmät noudattavat näitä periaatteita. Arvioinnissa keskitytään muun muassa tekoälyn vaikutukseen päätöksentekoon ja tiedon luotettavuuteen, erityisesti hallusinaatioiden välttämisen kannalta. Lisäksi tarkastellaan, miten tekoälyavustajien käyttö voi vaarantaa ihmisen autonomiaa, jos käyttäjät turvautuvat niihin liiallisesti. Turvallisuus- ja yksityisyysnäkökohdat ovat myös arvioinnin keskiössä, erityisesti kun kyse on virtuaaliavustajista ja käyttäjän oikeudesta peruuttaa suostumuksensa tekoälyavusteisessa ympäristössä. Arviointilistassa kiinnitetään huomiota myös siihen, että käyttäjiä tiedotetaan tekoälyn teknisistä rajoituksista ja mahdollisista riskeistä. Lopuksi, listassa korostetaan eri sidosryhmien osallistumista tekoälyjärjestelmien kehittämiseen. (Piñeiro-Martín, García-Mateo, Docío-Fernández & López-Pérez, 3–4).

Tekoälypohjaisten ohjelmien käytön eettisiä riskejä ei suinkaan ole tarkasteltu ainoastaan Euroopassa vaan myös IEEE *Global Initiative on Ethics of Autonomous and Intelligence Systems* useiden satojen osallistujien voimin toteutettiin asiakirja, jonka avulla halutaan varmistaa, että kaikki sidosryhmät, jotka ovat mukana kehittämässä ohjelmistoa, toimivat priorisoiden eettisiä arvoja ja ihmisyyttä vaalien. Myös UNESCO:n suosituksissa tekoälyn eettisissä kysymyksissä jäsenmaita kehotetaan kiinnittämään erityistä huomioita koulutukseen, tieteeseen, kulttuuriin, viestintään ja informaatioon. Näiden suositusten käyttöönotto perustuu kuitenkin vielä vapaaehtoisuuteen Euroopassa. (Piñeiro-Martín, García-Mateo, Docío-Fernández & López-Pérez, 4).

Vuonna 2021 Euroopan Komissio teki ensimmäisen ehdotuksen tekoälyn sääntelemiseen. Sääntelyesitys seuraa riskiperusteista lähestymistapaa luokitellen tekoälyjärjestelmät viiteen eri kategoriaan. Tulevina vuosina ehdotukseen on tullut tarkennuksia sisämarkkina- ja kuluttajansuojavaliokunta sekä kansalaisvapauksien valiokunnalta. Olennaista on myös, että käyttäjälle tiedotetaan avoimesti, kun kyseessä on tekoälyjärjestelmä. (Piñeiro-Martín, García-Mateo, Docío-Fernández & López-Pérez, 4–5).

Vaikka artikkelissa paneudutaankin terveydenhoidossa käytettäviin virtuaaliassistentteihin ohjelmistokehityksessä käytettävien työkalujen sijaan, voidaan mielestäni esiin nousevia eettisiä haasteita ja uhkia kuitenkin soveltaa yhtä hyvin myös ohjelmistokehitystyössä käytössä oleviin työkaluihin. Alla on listaus esiin nousseista eettisistä huolista. Olen listaukseen sisällyttänyt sellaiset asiat, jotka ovat myös ohjelmistokehitystyökalujen käytössä relevantteja.

Kuten jo työni luvussa 3.2 pohdittiin, yhteneväisyyksiä löytyy tässä artikkelissa esiin nousseiden eettisten huolien kanssa:

1. Tietosuoja ja tietoturva
    - käyttäjätietojen kerääminen ja säilyttäminen
    - tietojen väärinkäyttö ja yksityisyydensuojan vaarantuminen
2. Vinoumat ja syrjintä
    - LLM-mallit, jotka siis pohjautuvat suuren määrään dataa, voivat vahvistaa tiedon vinoumia ja syrjintää
    - Tasa-arvon ja monimuotoisuuden huomiointi
3. Väärän tiedon ja hallusinaatioiden (täysin keksitty ja virheellinen tieto) levittäminen
    - LLM:t voivat generoida epäkoherenttia tai virheellistä tietoa
    - tarvitaan keinoja näiden virheellisten tietojen tunnistamiseen ja torjumiseen
4. Ihmisen (liiallinen) riippuvuus tekoälypohjaisista avustajista
    - kuinka ja miten paljon virtuaaliavustajat vaikuttavat ihmisen päätöksentekoon
5. LLM-pohjaisten assistenttien toiminnan läpinäkyvyys
    - ihmisen tulee ymmärtää mihin virtuaalisten avustajien päätöksenteko perustuu, kuinka algoritmi toimii sekä kuinka annettua dataa käytetään päätöksenteossa
6. Koulutusdata ja tekijänoikeudet
    - kuinka varmistua siitä, että kouluttamiseen käytetty data ei ole tekijänoikeuksien alaista tai että data olisi ennen käyttöönottoa tarkasteltu ja suodatettu eettiset periaatteet huomioiden
7. Sidosryhmien aktiivinen osallistuminen LLM-pohjaisten avustajien kehittämisessä

(Piñeiro-Martín, García-Mateo, Docío-Fernández & López-Pérez, 7–9).

Esiin nousseiden eettisten huolien pohjalta artikkelissa tehtiin parannusehdotuksia tekoälypohjaisten avustajien käyttöön.

Tekoälyjärjestelmät tulisi tarkistuttaa kolmansilla osapuolilla, jotta varmistutaan siitä, että eettiset periaatteet toteutuvat. Varmistutaan järjestelmän läpinäkyvyydestä ja ymmärretään sen toiminta päätöksentekoprosessissa ja pystytään myös tunnistamaan virheellinen tieto.

Myös koulutusdatan julkaisemista suositellaan läpinäkyvyyden lisäämiseksi. On varmistuttava siitä, että yksityisyys ja tietosuoja toteutuvat tietojenhallinnassa.

Käytetyn koulutusdatan monimuotoisuudesta varmistuminen sekä eri sidosryhmien (myös kolmansien osapuolten) osallistuminen mallien kehittämisessä. Näin voitaisiin myös tunnistaa ja torjua tiedon vinoumia.

Käyttäjille olisi aina selkeästi informoitava, että kyseessä on tekoälytyökalu ja varmistuttava siitä, että heiltä on saatu suostumus heidän tietojensa keräämiseen ja käyttämiseen. Käyttäjien on oltava tietoisia siitä, miten heidän tietojaan käytetään, ja mahdollisuus halutessaan evätä suostumuksensa.

Koska tekoälytyökalut kehittyvät jatkuvasti, tarvitaan eettisten ongelmien tunnistamiseen jatkuvaa arviointia ja käytetyn datan keräämistä sekä yhteistyötä kehittäjien ja eri sidosryhmien (mm. yksityisyyden ja tietosuojan ammattilaisten) välillä koko projektin elinkaaren aikana.

(Piñeiro-Martín, García-Mateo, Docío-Fernández & López-Pérez, 10–12).

5 Yhteenveto

Tutkimuksissa tarkasteltiin alalla käytetyimpiä tekoälytyökaluja koodin generoimiseen, ja tuotettua koodia arvioitiin koodiongelmien ratkaisemisen kautta. Tekoälyn tuottamaa koodia verrattiin myös ihmisen tuottamaan koodiin, jolloin pystyttiin analysoimaan missä määrin tekoälytyökalu olisi parempi ko. tehtävässä ihmiseen verrattuna. Lisäksi tarkasteltiin ohjelmoijien asenteita tekoälytyökalujen käyttöön, sekä miten ne voivat tehostaa ohjelmoijan työtä. Tämän analyysin pohjalta annettiin myös näkemys siihen, voisiko tekoälytyökalulla korvata ihmisohjelmoijan.

Useat käyttävät erilaisia tekoälytyökaluja työssään jonkin verran (2,08 asteikolla 1 (harvoin)–5 (usein)), joista yleisimpiä ovat Chat-GPT 71,7 % ja Copilot 32,3%. Tekoälytyökaluja käytetään mm. koodin luomiseen, koodin selittämiseen, virheiden löytämiseen koodista, dokumentoinnin kirjoittamiseen ja koodin suorituskyvyn parantamiseen sekä mm. uusien ratkaisujen löytämiseen.

ChatGPT sai parhaimmat pisteet koodiongelmien ratkaisemisessa. ChatGPT:n vuorovaikutus ihmisen kanssa oli onnistunut, sillä mm. 16 virheellisestä koodista se onnistui annetun palautteen avulla ratkaisemaan 14. Mutta optimaalinen toimivuus on siis vahvasti sidottuna ihmiskoodaajan panokseen. Ihmisen antamalla promptilla, jopa sanavalinnoilla, on merkitystä LLM:n generoimaan koodiratkaisuun, jolloin ihmisen vastuulla on koodin valinta ja sen hyväksyminen. Joten vaikka ohjelmoija säästää aikaa koodin tuottamisessa, työaikaa menee koodin tarkasteluun ja soveltuvuuden arviointiin.

Myös Copilot suoriutui ohjelmointitehtävistä hyvin (50 %), onnistuen tuottamaan jossain määrin vähemmän koodirivejä kuin ihminen, ja jonkin verran vähemmän kompleksista koodia luoden laadukkaampaa koodia kuin ihminen. Koodin suorituskykyä mitattaessa vain alle neljäsosassa tapauksista tekoälyn ratkaisu suoritettiin nopeammin kuin ihmisen kirjoittama. Muistinkäytön suhteen ihmisen koodi käytti puolestaan vähemmän muistia (42 % kerroista). Puolet kerroista tekoälyn tuottama koodi oli ylläpidettävämpää, mutta luonnollisesti sama juttu toisin päin. Yleisesti ottaen helppoja ongelmia ratkaistiin useimmiten (86,1 %), kun taas vaikeampien ongelmien läpäisyprosentti oli tekoälyllä alhaisempi.

Mitä tulee tekoälyn tuottaman koodin korjaamiseen ja ajankäyttöön sen suhteen, havaittiin, että koodia kannattaa korjata uudelleen kirjoittamisen sijaan, tällä tavoin voidaan säästä jopa 71.31 % aikaa. Voidaan siis todeta, että generatiiviset tekoälytyökalut suoriutuvat vaihtelevasti ohjelmointikielestä ja koodausongelmasta riippuen. Tutkimukset todistavat myös sen, ettemme voi vielä täysin luottaa siihen, että tekoäly tuottaisi aina oikeaa, tehokasta ja ylläpidettävää ohjelmakoodia. Tekoäly voi kuitenkin nopeuttaa ohjelmointia, koska vaikka ohjelmakoodi olisi toimimatonta, se voidaan saada korjattua pienellä vaivalla ja lyhyessä ajassa.

Ohjelmoijien asennetta tarkastellessa huomattiin mm., että käyttökelpoisuus arvioitiin kohtalaiseksi, ja käyttäjäkokemus arvioitiin keskiverroksi. Työkalujen käytöstä pidettiin, vaikka aina niiden käytön ei koettu tehostavan työntekoa. Yleisesti ottaen kehittäjät, joilla oli ennestään kokemusta näistä työkaluista arvioivat ne positiivisesti, kun taas henkilöt, joilla vähemmän kokemusta suhtautuivat tekoälytyökaluihin kriittisemmin. Kokeneet ohjelmistokehittäjät kokivat avustajan käytännöllisemmäksi ja pitivät sen käyttämisestä. Kokemattomammat käyttäjät saattavat hämmentyä epärelevanttien ehdotusten keskellä. Työkalujen käyttökokemukseen vaikutti siis olennaisesti ehdotusten soveltuvuus annetussa kontekstissa, sillä tekoälytyökalun generoiman koodin muodollinen varmennus ja validointi on haastavaa.

Kehittäjien asenteet tekoälytyökaluja kohtaan siis vaihtelevat: jotkut luottavat niihin ja kokevat ne hyödyllisiksi tuottavuuden lisäämisessä ja koodin laadun parantamisessa, toiset suosivat niiden käyttöä vain yksinkertaisissa tehtävissä. On havaittu myös, että tekoälytyökalujen käyttö luo uusia tehtäviä, mikä johtaa työntekijöiden keskittymiseen luovempiin tehtäviin. Tämä voi puolestaan parantaa tuottavuutta ilman työpaikan korvaamisen riskiä.

Vaikka tekoälytyökalut eivät välttämättä merkittävästi paranna olemassa olevan koodin laatua tai vähennä virheitä, ne voivat nopeuttaa koodin kirjoittamista, kun ne ymmärtävät entistä paremmin kehittäjän kontekstia ja onnistuvat tarjoamaan relevantteja koodin täydennyksiä.

Ohjelmoijat kokivat tekoälytyökalujen lisäävän jonkin verran tuottavuutta (2,53 asteikolla 1–5) mutta kokivat myös, että työkalut tuottavat joskus virheellistä koodia.

Tekoälytyökalujen käyttöä ilmoitti jatkavansa valtaosa (82 %).

Tekoälytyökalujen käyttötarkoituksissa havaittiin kaksi tilannetta/tilaa, jossa ohjelmoija on vuorovaikutuksessa niihin: kiihdytystila ja tutkimustila. Ko. tutkimuksessa ohjelmoijia kehotettiin käyttämään tukena Copilotia.

Kiihdytystilassa ohjelmoija käyttää Copilotia suorittaakseen suunnitellut kooditoiminnot nopeasti, käyttämällä Copilotin automaattisesti ehdottamaa ehdotusta koodin kirjoittamisessa. Kiihdytystila auttaa ohjelmoijaa työskentelemään nopeammin. Kiihdytystilassa ohjelmoija yleensä hyväksyy Copilotin ehdotuksen ilman sen suurempaa epäröintiä ja jatkaa jouhevasti työskentelyä. Kiihdytystilassa usein Copilotia käytetään pienempiin koodin osiin/ongelmiin, jolloin Copilotin ehdotukset ovat usein helposti hyväksyttävissä. Mikäli taas Copilotin ehdotus oli liian pitkä, ei sisältänyt tiettyjä avainsanoja tai rakenteita, ohjelmoija herkemmin hylkäsi nämä ehdotukset. Toisaalta he hyväksyivät lähes oikeat ehdotukset, jos ne olivat parannettavissa pienellä korjauksella.

Tutkimustilassa ohjelmoija käyttää Copilotia suunnitellakseen kooditoimintojaan esim. tuntemattoman syntaksin, oikean API:n etsimisessä tai oikean algoritmin löytämisessä, toisin sanoen, kun ei ole tarkkaa tietoa mistä ja miten aloittaa. Tutkimustilassa myös yritettiin löytää ratkaisu toimimattomaan koodiin. Luonnollisen kielen promptien kirjoittaminen koettiin vaivattomana ja useat ohjelmoijat olisivat valmiita lisäämään tarpeeksi yksityiskohtaista tietoa kommentteihin, jotta Copilotilla olisi tarpeeksi kontekstia tuottaakseen varteenotettavia ehdotuksia.

Tutkimustilassa osallistujat olivat enemmän valmiita käyttämään aikaa vuorovaikutukseen Copilotin kanssa, sekä Copilotin useiden eri ehdotusten läpikäymiseen, arvioimiseen ja vertaamiseen, toisin kuin kiihdytystilassa. Osa osallistujista koki useiden erilaisten ehdotusten vertailun kuitenkin aikaa vievänä ja kuluttavana. Tutkimustilassa osallistujat olivat halukkaita hyväksymään ehdotuksia muokkaamalla niitä omaan tarkoitukseensa paremmiksi. Sudenkuoppana havaittiin, että virheiden havaitseminen Copilotin generoimissa koodiehdotuksissa oli hankalampaa.

Ohjelmoijat vaihtelevat näiden kahden tilan välillä ja siirtyvät sulavasti tilasta toiseen.

Vaikka tekoälytyökalut voivat tarjota merkittäviä hyötyjä, niiden käytössä ilmenee yhä puutteita ja rajoitteita, joten ohjelmoijien rooli kriittisenä tarkistajana säilyy erittäin keskeisenä. Tutkimukset osoittavat, että yli kaksi kolmasosaa osallistujista ei näe välitöntä uhkaa työpaikan suhteen tekoälytyökaluihin liittyen. Toisaalta pohdittiin myös, että tekoälytyökalut eivät kykenisi korvaamaan ihmistä, koska tarvitaan ihmisen toimialatuntemusta ja luovuutta. Osallistujat, jotka kokivat suurempaa tuottavuutta tekoälytyökalujen kanssa, luottivat näihin työkaluihin enemmän, mutta tunsivat myös enemmän uhkaa työpaikkansa korvautumisesta.

Tekoälyn vaikutuksesta työllisyyteen ja työmarkkinoihin puhutaan paljon. Vaikka tekoälytyökalut helpottavatkin ohjelmoijia monissa tehtävissä, ohjelmoijat eivät luota sokeasti näihin ja haluavatkin pysyä prosessissa vahvasti mukana. Lisäksi, kuten jo työni teoriaosuudessakin oli puhe (ks. luku 3.1) tutkimuksissa tuodaan esiin tekoälytyökalujen vaikutus uusien työpaikkojen, ammattien ja alojen luojina. Siispä vaikka jotkut työtehtävät voitaisiin automatisoida, kokonaistyöllisyyden vaikutus riippuu siitä, miten työtehtäviä pystytään määrittelemään ja jakamaan uudelleen.

Eettisten haasteiden tarkastelussa ja ehkäisyssä tulee kiinnittää entistä enemmän huomiota tietosuojaan ja -turvaan, vinoumien ja syrjinnän ennaltaehkäisyyn sekä LLM:n virheellisen tiedon tunnistamiseen ja torjumiseen. On lisäksi pohdittava tekoälytyökalujen vaikutusta ihmisen päätöksentekoon, ja ymmärrettävä mihin tekoälytyökalujen ratkaisut pohjautuvat. Myös koulutusdataa tulisi tarkastella eettiset periaatteet huomioiden.

Koska tekoälytyökalut kehittyvät jatkuvasti, tarvitaan eettisten ongelmien tunnistamiseen jatkuvaa arviointia ja käytetyn datan keräämistä sekä yhteistyötä kehittäjien ja eri sidosryhmien (mm. yksityisyyden ja tietosuojan ammattilaisten) välillä koko LLM:n pohjaisten avustajien kehittämisen elinkaaren aikana.

Tekoälytyökalut ovat lisääntyneet ja tulevat vastedeskin lisääntymään, jolloin on syytä jatkaa tutkimusta niiden vaikutuksesta ohjelmistoteollisuuteen, jotta pystytään käsittämään, mihin tekoälytyökalut kykenevät, sekä toisaalta mitkä ovat niiden rajoituksia. Koska tekoälyavusteinen koodaaminen lisääntyy ja uusia teknologioita tulee valtavalla vauhdilla, tarvitaan ymmärrystä siitä, millaisiin tehtäviin niiden käyttö soveltuu parhaiten, kuinka niiden käyttö voidaan optimoida, jotta voidaan saavuttaa haluttu lopputulos koodin laatu ja taloudellinen ajankäyttö huomioiden. Vastaavasti on kyettävä tunnistamaan myös näiden työkalujen tekniset rajoitteet ja eettiset haasteet.

Kuten jo aiemmissa tutkimuksissa on käynyt ilmi, tekoälytyökalut eivät, ainakaan tämänhetkisen kapasiteettinsa ja suorituksensa perusteella, kykene täysin korvaamaan ihmisen panosta. Ihmisohjelmoijan rooli on keskeinen täsmällisten promptien antamisessa, sekä generoidun koodin laadun, toimivuuden ja soveltuvuuden arvioinnissa välttämätön. Lisäksi vaikka tekoälytyökaluilla saataisiinkin korvattua manuaalista koodin kirjoittamista, ja tämänkaltaiset työtehtävät vähentyisivätkin automatisoitumisen myötä, samanaikaisesti tarvitaan lisää työtehtäviä tekoälytyökalujen toimintojen valvontaan.

6 Pohdinta

Yleisesti ottaen tekoäly kehittyy kehittymistään, ja ihmisen ja tekoälyn yhteistyötä on syytä tutkia vastedeskin ja ymmärtää minkälaisiin tehtäviin sitä voi ja kannattaa käyttää apuna ja milloin ihmisen työpanoksella ja erityisesti päätöksenteolla on väliä. Ihmisen vastuu ja velvollisuus, kun onkin, ainakin tämänhetkisten tutkimusten perusteella, ymmärtää miten tekoäly toimii ja mihin sen päätöksenteko perustuu eikä sokeasti luottaa saamaansa ratkaisuun tai dataan.

Itse olen tämän tutkimuksen myötä saanut laajan katsauksen tekoälytyökalujen käytöstä ohjelmistokehityksessä. Olen kuitenkin saanut huomata kuinka, ennakkokäsityksen vastaisesti, kaikki kehittäjät eivät suhtaudu ainoastaan myönteisesti tekoälytyökalujen käyttöön. Tekoälytyökaluihin kriittisesti suhtautuvien huolenaiheina ovat mm. tekoälytyökalujen ehdotusten relevanttius, annettujen ehdotusten arviointiin kuluva aika sekä myös uhkakuvat nykyisiin työtehtäviin ja työpaikkoihin, varsinkin junioritasolla. Toki tekoälytyökalujen todettiin myös tehostavan työntekoa ja niistä koettiin saavan hyötyä niin pienien kuin suurempienkin koodinosien työstämiseen. Tarkastelemani työkalut erosivat toiminnallisuuksiltaan hieman toisistaan. Suosituimmaksi osoittautui ChatGPT, toisena GitHub Copilot ja kolmantena Tabnine. GitHub Copilotin ja Tabninen käyttö keskittyy keskeisesti koodin täydennykseen. Copilotin muista kahdesta erottavana toimintona on kyky täydentää lähdekoodin kokonaisia osia sekä nimetä muuttujia. Tabninen vahvuus puolestaan on kyky tutkia API-rajapintoja. ChatGPT:n erottaa muista kahdesta sen keskustelunomainen vuorovaikutuksellisuus ihmisen kanssa mahdollistaen koodiongelman laaja-alaisemman tarkastelun.

Löydökset ovat väistämättä ajatuksia herättäviä, varsinkin alanvaihtajana ja alalle pyrkivänä. On syytä kysyä, ollaanko jo siinä vaiheessa, kun ohjelmointialalle pyrkivillä on vaikeammat olosuhteet päästä alalle, koska tekoälytyökaluilla saadaan tehokkaasti ja näppärästi tehtyä vastaavat työtehtävät. Kenties jo muutaman vuoden sisällä, kun esittää tämän tutkimuskysymyksen, vastaus voi olla täysin selkeä.

Lähteet

Aaltonen, M. 2019. Tekoäly - ihminen ja kone. Helsinki: Alma Talent Oy.

Alto, V. 2023. Modern Generative AI with ChatGPT and OpenAI Models : Leverage the Capabilities of OpenAI's LLM for Productivity and Innovation with GPT3 and GPT4. Birmingham: Packt Publishing, Limited.

Barke, S., James, M. B., & Polikarpova, N. (2023). Grounded Copilot: How Programmers Interact with Code-Generating Models. Proceedings of ACM on programming languages, 7(OOPSLA1), 85-111. Luettavissa: https://doi.org/10.1145/3586030. Luettu 23.4.2024.

Coello, C. E. A., Alimam, M. N., & Kouatly, R. (2024). Effectiveness of ChatGPT in Coding: A Comparative Analysis of Popular Large Language Models. Digital, 4(1), 114-125. Luettavissa: https://doi.org/10.3390/digital4010005. Luettu 23.4.2024.

Dean, D.J. 12.3.2024. Principal Cloud Solution Architect. GPT-4 use cases, new vision capabilities, speech capabilities, as well as some architecture perspectives. Microsoft. Esitys. Microsoft Teams.

Dhamani, N., Engler, M. 2024. Introduction to Generative AI. New York: Manning Publications Co. LLC.

European Union Agency for Fundamental Rights 2021. Tienviittoja tulevaisuuteen : tekoäly ja perusoikeudet : tiivistelmä. Publications Office of the European Union. Luettavissa: https://data.europa.eu/doi/10.2811/27576. Luettu 14.4.2024.

Forbes Technology Council 2022. 15 Jobs And Tasks Tech Experts Believe Will Be Automated Within A Decade. Forbes. Luettavissa: https://www.forbes.com/sites/forbestechcouncil/2022/02/18/15-jobs-and-tasks-tech-experts-believe-will-be-automated-within-a-decade/. Luettu 14.4.2024.

Github Copilot 2024. Your AI pair programmer. Luettavissa: https://github.com/features/copilot. Luettu 4.4.2024.

Gombrich E. 2024. 11 Best AI Tools for Coding in March 2024. Luettavissa: https://techvibe.ai/ai-tools-for-coding/. Luettu 21.3.2024.

Gombrich E. 2024. Pros & Cons GitHub Copilot / Pros & Cons Tabnine. Luettavissa: https://techvibe.ai/ai-tools-for-coding/. Luettu 21.3.2024.

Gupta R. 2023. ChatGPT vs Software Developers: Is Generative AI the End of the Road for Developers? Luettavissa: https://www.turing.com/blog/will-generative-ai-replace-software-developers/. Luettu 14.4.2024.

Haas, J. 2023. Why AI + No-Code is the Future. Bubble Blog. Luettavissa: https://bubble.io/blog/bubbleai-no-code/. Luettu 14.4.2024.

Hliš, T., Četina, L., Beranič, T., & Pavlič, L. 2023. Evaluating the Usability and Functionality of Intelligent Source Code Completion Assistants: A Comprehensive Review. Applied sciences, 13(24), 13061. Luettavissa: https://doi.org/10.3390/app132413061. Luettu 25.4.2024.

Idrisov, B., & Schlippe, T. 2024. Program Code Generation with Generative Ais. Algorithms, 17(2), 62. Luettavissa: https://doi.org/10.3390/a17020062. Luettu 25.4.2024.

Jeuring, J., Roel, G. & Hieke, K. 2023. What Skills Do You Need When Developing Software Using ChatGPT? Luettavissa: https://arxiv.org/pdf/2310.05998.pdf. Luettu 2.4.2024.

Kananen, H., Puolitaival, H. 2019 Tekoäly : bisneksen uudet työkalut. Helsinki: Alma Talent.

Kohonen, T., 1990. The self-organizing map. Proceedings of the IEEE, 78(9), pp. 1464-1480. Luettavissa: https://doi.org/10.1109/5.58325. Luettu 7.10.2024.

Kuhail, M. A., Mathew, S. S., Khalil, A., Berengueres, J., & Shah, S. J. H. 2024. “Will I be replaced?” Assessing ChatGPT's effect on software development and programmer perceptions of AI tools. Science of computer programming, 235, . Luettavissa: https://doi.org/10.1016/j.scico.2024.103111. Luettu 29.4.2024.

McCallum, S. 2023. ChatGPT banned in Italy over privacy concerns. BBC News. Luettavissa: https://www.bbc.com/news/technology-65139406. Luettu 14.4.2024.

Megatrendit Sitra 2023. Luettavissa: https://www.sitra.fi/julkaisut/megatrendit-2023. Luettu 14.4.2024.

Nanda, A. 2023. The Future of Low Code and No Code: How Generative AI is Revolutionizing Software Development. Linkedin Blog. Luettavissa: https://www.linkedin.com/pulse/future-low-code-how-generative-ai-revolutionizing-software-atul-nanda. Luettu 14.4.2024.

Open AI Blog. Luettavissa: https://openai.com/blog/chatgpt. Luettu 2.4.2024.

OpenAI ChatGPT 2023. Luettavissa: https://openai.com/chatgpt. Luettu 2.4.2024.

Peng, S.;Kalliamvakou, E.;Cihon, P.;& Demirer, M. 2023. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. Tutkimus. Luettavissa: https://arxiv.org/pdf/2302.06590.pdf. Luettu 11.4.2024.

Piñeiro-Martín, A., García-Mateo, C., Docío-Fernández, L. & López-Pérez, M. d. C. 2023. Ethical Challenges in the Development of Virtual Assistants Powered by Large Language Models. Electronics (Basel), 12(14), 3170. Luettavissa: https://doi.org/10.3390/electronics12143170. Luettu 5.5.2024.

Rahmaniar, W. 2023. ChatGPT for Software Development: Opportunities and Challenges. TechRxiv. Luettavissa: https://doi.org/10.36227/techrxiv.23993583.v1. Luettu 2.4.2024.

Roshi E. 2023. Best AI tools for developers 2024. Luettavissa: https://codeless.co/best-ai-tools-for-developers./. Luettu 21.3.2024.

Salo, I. 2023. Luova tekoäly mullistaa kaiken : ChatGPT näyttää tietä. Helsinki: Kauppakamari.

Stack Overflow. Luettavissa: https://stackoverflow.com/. Luettu 26.3.2024.

Stack Overflow kysely. Luettavissa: https://survey.stackoverflow.co/2023/. Luettu 26.3.2024.

Tabnine 2023. Luettavissa: https://www.tabnine.com. Luettu 11.4.2024.

Tabnine AI Code Assistant Guide 2023. Luettavissa: https://www.tabnine.com/blog/ai-code-assistant-guide. Luettu 11.4.2024.

Taulli, T. 2023. Generative AI: How ChatGPT and Other AI Tools Will Revolutionize Business. Berkeley, CA: Apress L. P.

Vilkka, H. 2023. Kirjallisuuskatsaus metodina, opinnäytetyön osana ja tekstilajina. Helsinki: Art House.

Witts. Parhaimmiksi ja käytetyimmiksi todettuja koodaajien työkaluja vuonna 2024. Luettavissa: https://expertinsights.com/insights/the-top-5-ai-code-generation-solutions-for-programmers/. Luettu 21.3.2024.

Yepis, E. 2023. Hype or not? AI’s benefits for developers explored in the 2023 Developer Survey. Luettavissa: https://stackoverflow.blog/2023/06/14/hype-or-not-developers-have-something-to-say-about-ai/?cb=1. Luettu 30.3.2024.

York A. 2024. Developer to boost coding efficiency. Luettavissa: https://clickup.com/blog/ai-tools-for-developers/. Luettu 21.3.2024.