OULUN AMMATTIKORKEAKOULU

Jonna Neulikko

TEKOÄLYN HYÖDYNTÄMINEN AUTOMAATIOTESTAUKSESSA

Tekoälypohjaisen koodiavustajan valintaprosessi

TEKOÄLYN HYÖDYNTÄMINEN AUTOMAATIOTESTAUKSESSA

Tekoälypohjaisen koodiavustajan valintaprosessi

Jonna Neulikko
Opinnäytetyö
Syksy 2024
Tietojenkäsittelyn tradenomi
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn tradenomi

Tekijä(t): Jonna Neulikko
Opinnäytetyön nimi: Tekoälyn hyödyntäminen automaatiotestauksessa - tekoälypohjaisen koodiavustajan valintaprosessi
Työn ohjaaja(t): Markku Kekkonen
Työn valmistumislukukausi ja -vuosi: Syksy 2024 Sivumäärä: 36 + 4 liitettä

Tekoälyn kehittyessä erilaisten tekoälypohjaisten ratkaisujen rooli ohjelmistokehityksen tukena kasvaa tulevaisuudessa, mikä edellyttää yrityksiltä jatkuvaa sopeutumista ja uusien teknologioiden omaksumista.

Opinnäytetyön toimeksiantajana oli Esko Systems Oy, jonka tavoitteena oli löytää sopiva tekoälypohjainen koodiavustaja ohjelmistokehityksen ja automaatiotestauksen tueksi.

Esko Systems Oy:n tuottaa Esko-nimistä potilas- ja asiakastietojärjestelmä, joka luokitellaan vaatimuksiltaan A3-luokan tietojärjestelmäksi ja osa tietojärjestelmän sovelluksista on lääkinnällisiä laitteita. Sen vuoksi sovelluksen kehittämisessä ja siihen integroitavien työkalujen valinnassa täytyy noudattaa erilaisia sosiaali- ja terveysalan tietojärjestelmien määräyksiä ja säädöksiä sekä lääkinnällisiä laitteita koskeva lakeja ja standardeja.

Työssä kartoitettiin kolme potentiaalista koodiavustajaa: Tabnine, Codeium ja GitHub Copilot, joiden tiedot kerättiin vertailua varten erilliselle arviointilomakkeelle ja koodiavustajien suorituskykyä testattiin lyhyiden kooditehtävien avulla. GitHub Copilot valittiin lopulta parhaaksi vaihtoehdoksi sen taustalla toimivien vakaiden kehittäjien, luotettavuuden, jatkuvan kehityksen ja kattavan dokumentaation ansiosta. Koodiavustaja vaatii kuitenkin vielä kelpuutusprosessin ensin varsinaista yrityksen sisäistä käyttöönottoa, että sen valinta täyttää viranomaisvaatimukset ja vastaa laadunhallinnan asetuksia.

Asiasanat: Tekoäly, Automaatiotestaus, Koodiavustaja, Sosiaali- ja terveydenhuollon tietojärjestelmät

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology

Author(s): Jonna Neulikko
Title of thesis: Utilizing Artificial Intelligence in Automation Testing - selection process for an AI-based code assistant
Supervisor(s): Markku Kekkonen
Term and year when the thesis was submitted: Autumn 2024
Number of pages: e.g. 36 + 4 appendices

As artificial intelligence continues to develop, the role of various AI-based solutions in supporting software development is expected to grow in the future, requiring companies to continuously adapt and adopt new technologies.

The thesis was commissioned by Esko Systems Ltd, whose goal was to find a suitable AI-based code assistant to support software development and automation testing.

Esko Systems Oy produces the Esko patient and client information system, which is classified as an A3-level system in terms of requirements, and some of its applications are classified as medical devices. Therefore, the development of the application and the selection of tools to be integrated into it must comply with various regulations and standards for health and social care information systems, as well as laws and standards related to medical devices.

The study identified three potential code assistants: Tabnine, Codeium, and GitHub Copilot. Information about these assistants was collected on a separate evaluation form, and their performance was tested with short coding tasks. GitHub Copilot was ultimately chosen as the best option due to its reliable developers, continuous development, and comprehensive documentation. However, the code assistant still requires a validation process before it can be fully implemented within the company to ensure that its selection meets regulatory requirements and complies with quality management standards.

Keywords: Artificial Intelligence, Automation testing, Code assistant, health and social care information systems

SISÄLLYS

Sanasto

Algoritmi	Yksityiskohtainen kuvaus tai ohje, jota seuraamalla tehtävä, prosessi tai ongelmanratkaisu suoritetaan
Asiakastietojärjestelmä	Sosiaalihuollon asiatietojen käsittelyä varten suunniteltu ja valmistettu ohjelma
Bugi	Ohjelman lähdekoodissa oleva virhe
Data	Digitaalisesti tallennettua koneellisesti luettavaa tietoa, joka koostuu merkeistä ja symboleista
GAMP 5	Ohjeistus, joka toimii apuna kelpuutuksessa
Kehitysympäristö	Ohjelmisto, jolla koodari kirjoittaa ohjelmistoa
Kelpuutus	Menettely, jolla varmistetaan, että tuote soveltuu aiottuun käyttötarkoitukseen
Koodiavustaja	Ohjelmointityökalu, joka käyttää tekoälyä ohjelmointiprosessin tukemiseen. Auttaa kehittäjiä kirjoittamaan koodia, korjaa virheitä koodista ja täydentää koodia
Määräys	Oikeusohjeessa säädöstä alempiasteinen yksittäinen käsky tai kielto
Palvelubotti	Ohjelma, joka on suunniteltu keskustelemaan ihmisen kanssa joko puheen tai kirjoitetun tekstin välityksellä
Potilastietojärjestelmä	Terveydenhoidon yksiköissä käytettävä ohjelma, joka tallentaa henkilön terveystietoja
Skaalautuvuus	Tietojärjestelmän kyky mukautua kasvavaan tarpeeseen prosessoida tietoa
Skripti	Peräkkäin toteutettavien erilaisten komentojen muodostama kokonaisuus
Standardi	Julkaisu, johon on kirjattu yhteisesti sovittuja vaatimuksia, suosituksia tai ominaisuuksia
Säädös	Oikeusohjeen sisältävä tekstikokonaisuus
Teknologia	Ihmisen kehittämä keino tai väline, jolla helpotetaan elämää
Vaatimus	Tiukka ja ehdoton ilmaus siitä, kuinka on toimittava tai meneteltävä ja mitä ehtoja tai rajauksia on täytettävä

1 JOHDANTO

Tekoäly on terminä ”vanha keksintö”, mutta muutamien viime vuosien aikana tapahtunut nopea kehitys on tehnyt siitä nykypäivän teknologisen murroksen keskipisteen. Tämä kehitys on tuomassa erilaisia tekoälypohjaisia ratkaisuja arkipäiväiseen käyttöön eri aloille kuten sovelluskehitykseen ja terveydenhuoltoon. Muutoksessa mukana pysyminen on tärkeää ja tarkoittaa organisaatioiden kannalta jatkuvaa sopeutumista, uuden oppimista ja uusien teknologioiden omaksumista.

Opinnäytetyön aihe tuli vastaan opintoihin kuuluvan harjoittelun loppupuolella Esko Systems Oy:ssä, joka tuottaa Esko-nimistä asiakas- ja potilastietojärjestelmää. Yrityksen tavoitteena oli kartoittaa tekoälyn hyödyntämismahdollisuuksia tuotekehityksessä, erityisesti ohjelmistotestauksessa. Aihe oli mielenkiintoinen ja erittäin ajankohtainen, mikä teki siitä luontevan valinnan opinnäytetyöksi.

Tarkoituksena oli löytää tekoälypohjainen työkalu ohjelmistotestauksen tueksi. Alustavassa kartoituksessa huomattiin, että tekoälypohjaisen koodiavustajan valitseminen voisi olla hyödyllisempää, sillä sitä voitaisiin hyödyntää automaatiotestauksen lisäksi ohjelmistokehityksen eri vaiheissa.

Työkalun valintaprosessiin kuului kriteerien määrittely sekä työkalujen arvioinnin suorittaminen erillisen arviointilomakkeen avulla. Lisäksi työkalujen käytettävyyttä testattiin kehitysympäristössä lyhyiden kooditehtävien avulla, mikä antoi tarkempaa tietoa niiden suorituskyvystä ja käytettävyydestä ohjelmistokehityksen ja automaatiotestauksen tukena.

Opinnäytetyö pyrkii vastaamaan seuraamaan tutkimuskysymykseen: *”Miten sosiaali- ja terveysalan tietojärjestelmien erityisvaatimukset vaikuttavat tekoälypohjaisen testityökalun valintaan?”* Tämä kysymys ohjasi työn etenemistä, koska sosiaali- ja terveysalan tietojärjestelmät ovat tarkasti säänneltyjä ja niiden tulee täyttää tiukat säädökset ja standardit.

Opinnäytetyön aineisto kerättiin kesän 2024 aikana ja itse loppuraportti kirjoitettiin syyslukukauden 2024 aikana puhtaaksi. Tekoälyteknologioiden kehittämisen ja kehittymisen nopeuden huomasi opinnäytetyöprosessin aikana siitä, että loppuraportin kirjoittamisen aikana valintaprosessiin poimitut ohjelmat ehtivät päivittyä ja lomakkeille kerätyt tiedot muuttua.

2 TEKOÄLY

Tekoäly tai AI (lyhenne, joka tulee englanninkielisestä termistä Artificial Intelligence) on kattotermi erilaisille ohjelmointitekniikoille, joilla erilaisia ihmisille ominaisia kykyjä ja toimintoja pyritään matkimaan tietokoneella (Järvinen 2024, 18). Tällaisia ominaisuuksia ovat oppiminen, päättely, suunnittelu ja luominen (Euroopan parlamentti 2020). Tavoitteena on, että koneet kykenevät suorittamaan ihmisen kaltaisia älyllisiä toimintoja kuten käännöstyöt, puheentunnistus ja -tuotanto tai päätöksenteko.

Tekoäly eroaa perinteisistä ohjelmistoista monin eri tavoin. Perinteiset ohjelmistot on suunniteltu ja rakennettu tarkasti määriteltyjen sääntöjen ja toimintojen mukaan. Ne pystyvät suorittamaan vain ne tehtävät, joihin ne on erikseen ohjelmoitu, eivätkä ne voi toimia ennalta määriteltyjen sääntöjen ulkopuolella ilman ihmisen tekemiä koodimuutoksia tai päivityksiä. Tekoäly puolestaan kykenee oppimaan ja mukautumaan ympäristöönsä käsittelemänsä datan perusteella. Sen kyky oppia ja mukautua perustuu koneoppimiseen ja datan analysointiin. Tekoäly kykenee itsenäisesti mukauttamaan toimintaansa datasta tekemiensä havaintojen perusteella ilman, että ihminen välttämättä puuttuu prosessiin. (CGI 2024.)

2.1 Tekoälyn muodot

Tekoälyllä on useita eri muotoja, jotka voidaan jakaa eri kategorioihin sen mukaan, millainen on tekoälyn kyky suorittaa erilaisia toimintoja. Luokittelun pohjalta tekoäly voidaan jakaa kolmeen eri pääkategoriaan: kapea tekoäly, yleinen tekoäly ja supertekoäly. Tällä hetkellä kaikki käytössä olevat tekoälyjärjestelmät edustavat kapeaa tekoälyä. Vauhdilla etenevät tutkimus ja kehitys tähtäävät kuitenkin jatkuvasti kohti monipuolisempia ja älykkäämpiä tekoälyn muotoja.

Kapean tekoälyn eri alamuodoista erityisesti generatiiviset tekoälyteknologiat ovat lyhyessä ajassa nousseet merkittävään asemaan. Näitä ovat esimerkiksi tekstin, kuvien, koodin, testidatan tai musiikin luomiseen tarkoitetut järjestelmät.

2.1.1 Kapea tekoäly

Kapea tekoäly (eng. Artificial Narrow Intelligence), josta käytetään myös nimeä ”heikko tekoäly” (eng. Weak Artificial Intelligence) viittaa tekoälypohjaisiin järjestelmiin tai ohjelmiin, jotka on suunniteltu hoitamaan tiettyjä rajattuja tehtäviä. Tavoitteena on, että se näyttää älykkäältä siinä tehtävässä tai tehtävissä, mihin se on luotu. Toisin kuin yleinen tekoäly, joka voi oppia ja mukautua erilaisiin tilanteisiin, kapea tekoäly toimii siihen asetettujen sääntöjen puitteissa ja on täysin niiden sitoma. Se ei pysty menemään näiden sääntöjen yli ja tekemään itsenäisiä ratkaisuja tai säännöistä poikkeavia toimintoja. (Maria 2024.)

Hyvä esimerkki kapean tekoälyn toiminnasta on shakkipeli. Kapea tekoäly ei tiedä shakista mitään. Se ei osaa arvioida itsenäisesti, mikä on hyvä tai huono siirto, eikä esimerkiksi mieti, mikä siirto on strategisesti paras missäkin tilanteessa tai myöskään arvioi pelin kokonaiskuvaa. Sen sijaan se analysoi jokaisen siirron sääntöjen ja ohjelmoijien antamien parametrien perusteella. Tekoäly käy läpi lukuisia mahdollisia siirtoja ja valitsee algoritmin mukaan parhaan mahdollisen vaihtoehdon sekä tekee siirron sen perusteella. Shakkipelin ulkopuolella, sillä ei kuitenkaan ole kykyä päättää, mitä tehdä seuraavaksi. Jopa koneoppimisella varustettu tekoäly voi oppia ja soveltaa oppimaansa vain sen laajuuden mukaan, mitä siihen on ohjelmoitu. (Tekoäly.info 2024.)

2.1.2 Generatiivinen tekoäly

Perinteisen tekoälyn toiminta perustuu siihen, että se luokittelee, ennustaa, optimoi ja tekee analyyseja datan perusteella. Sen tehtävät liittyvät datan tulkitsemiseen ja käsittelyyn ennalta määrättyjen mallien mukaisesti. Generatiivinen tekoäly (eng. Generative Artificial Intelligence) puolestaan edustaa tekoälyn uudempaa suuntausta siten, että sen tavoitteena on tuottaa uutta sisältöä; tekstiä, kuvia, videoita, koodia, testidataa tai ääntä sille syötetyn koulutusdatan pohjalta. (Rouse 2024.)

Generatiivisen tekoälyn toimintaperiaate perustuu erilaisten generatiivisten mallien hyödyntämiseen, jotka ovat toteutettu koneoppisarkkitehtuurilla. Erilaiset generatiivisen tekoälyn mallit oppivat käsittelemään suuria datamääriä, tunnistamaan monimutkaisia rakenteita ja riippuvuuksia sekä luomaan näiden perusteella uutta sisältöä. Tällainen toiminta mahdollistaa erilaisissa ohjelmistoissa ja sovelluksissa esimerkiksi uuden tekstin tuottamisen, kuvien tai videoiden luomisen,

palvelubottien kyvyn vastata kysymyksiin, ja uuden koodin kirjoittamisen. (Feuerriegel ym. 2023. 112–113.)

2.1.3 Yleinen tekoäly

Yleinen tekoäly (eng. Artificial General Intelligence) viittaa teoreettiseen tekoälyn muotoon, koska se on edelleen kehitysvaiheessa, eikä sitä ole vielä pystytty saavuttamaan. Yleisen tekoälyn tavoitteena on, että se kykenee ymmärtämään, oppimaan ja suorittamaan mitä tahansa älyllistä tehtävää ihmisenkaltaisesti. Tutkijapiireissä yleisellä ja vahvalla tekoälyllä viitataan koneen kykyyn kehittää tietoisuus. Vahvaa tekoälyä voi soveltaa tehtävästä toiseen ja se kykenee luovaan ajatteluun ja oppimaan uutta itsenäisesti. Nykyiset tekoälysovellukset ovat vielä kaukana vahvasta tekoälystä, vaikka hyvin kehittynyt palvelubotti voi hetkellisesti onnistua antamaan vaikutelman ihmisenkaltaisesta älykkyydestä. (CGI 2024.)

2.1.4 Supertekoäly

Supertekoäly (eng. Artificial Super Intelligence) on toinen tekoälyn teoreettinen, lähinnä tieteiskirjallisuudesta tuttu, vielä älykkäämpi ja suorituskyvyltään tehokkaampi muoto kuin yleinen tekoäly. Se ylittää ihmisaivojen kyvyt moninkertaisesti jokaisella osa-alueella. On ennustettu, että supertekoäly syntyy sen jälkeen, kun vahva tekoäly on kehittynyt karkeasti ihmisen tasolle, minkä jälkeen tekoäly alkaa itse kehittää itsestään eksponentiaalisesti yhä älykkäämpiä versioita. (Bomstrom & Müller 2014, 1–2.)

2.1.5 Turingin testi

Tekoälyn arvioinnissa käytetty ihmisenkaltaisuuden tunnetuin mittari on Turingin testi, jonka mukaan tekoäly on saavuttanut ihmisen älykkyydessä siinä vaiheessa, kun tekoälyn kanssa keskusteleva ihminen ei enää pysty varmuudella erottamaan sitä, tulevatko vastaukset koneelta vai ihmiseltä. Testin on kehittänyt Alan Turing 1950-luvulla ja testiä sekä sen erilaisia versioita käytetään edelleen, kun halutaan mitata koneen kykyä ”ajatella”. Testissä ihmistuomari käy keskustelua sekä tekoälyn että toisen ihmisen kanssa ilman, että hän tietää, kumpi osapuoli on kumpi. Jos tuomari ei pysty luotettavasti erottamaan tekoälyä ihmisestä, katsotaan tekoälyn läpäisseen testin. (Saygin ym. 2000, 464–467.)

Turingin testiä voidaan käyttää myös generatiivisen tekoälyn eri sovellusten ”ihmismäisyyden” mittaamiseen. Generatiiviset tekoälymallit voivat tuottaa kielellisesti virheetöntä tekstiä, mutta niillä ei ole syvempää ymmärrystä puheen sisällöstä, tarkoituksesta tai merkityksestä. Tämä rajoite korostaa Turingin testin heikkoutta generatiivisen tekoälyn varsinaisen älykkyyden mittaamisessa: testi mittaa enemmän tekoälyn kykyä jäljitellä ihmisen käyttäytymistä kuin sen todellista älyä tai tietoisuutta. (Rouse 2024.)

2.2 Tekoälymenetelmät

Tekoälymenetelmät ovat erilaisia teknologioita ja algoritmeja, minkä avulla tekoälyjärjestelmiä luodaan, koulutetaan ja sovelletaan eri tarkoituksiin. Näitä menetelmiä voidaan jakaa useisiin eri kategorioihin riippuen siitä, mikä on tekoälyjärjestelmän haluttu käyttötarkoitus tai toimintatapa. Tässä kappaleessa kuvataan lyhyesti yleisimmät käytössä olevat menetelmät, joita käytetään tekoälyn kehittämiseen.

2.2.1 Koneoppiminen

Koneoppiminen (eng. Machine Learning) on tekoälyn osa-alue, joka mahdollistaa tietokoneille itsenäisen oppimisen, kehittymisen ja päätöksenteon historialliseen dataan perustuen ilman että niitä on erikseen ohjelmoitu käsittelemään jokaista mahdollista tilannetta. Se on menetelmä, jolla etsitään ratkaisua tietokoneen avulla johonkin tehtävään. Prosessi hyödyntää erilaisia algoritmeja, jotka ohjaavat ratkaisun kehittymistä haluttuun suuntaan. (Sarker 2021, 1–2.)

Koneoppimisprosessi koostuu kolmesta päävaiheesta: ensimmäinen vaihe on syötteen esikäsittely, jossa syötetty raakadata puhdistetaan ja muokataan sellaiseen muotoon, että se on sopivaa algoritmien käytettäväksi. Toisessa vaiheessa malli koulutetaan; algoritmit analysoivat syötteenä olevan datan ja oppivat sen perusteella tekemään ennusteita tai luokitteluita. Kolmannessa vaiheessa malli arvioidaan; virhefunktiota käytetään arvioimaan sitä, kuinka hyvin malli toimii ja tunnistaa mahdolliset parannuskohteet. Virhefunktio auttaa mallia oppimaan ja tekemään tarkempia ennusteita. Viimeisenä optimointiprosessi täydentää näitä vaiheita säätämällä mallin parametrejä ja asetuksia niin, että saavutetaan mahdollisimman hyvä suorituskyky ja tarkkuus. (Tekoälyn monet kasvot 2024; Sarker 2021, 5–10.)

Vaikka koneoppimisprosessi koostuu kolmesta erilaisesta päävaiheesta, se on kuitenkin muokattavissa ja hyödynnettävissä erilaisten mallien avulla erilaisiin tarkoituksiin. Mallien käyttötarkoitus riippuu muun muassa datan määrästä ja käyttökohteesta. Seuraavassa kaaviossa on kuvattuna neljä tyypillisintä koneoppimismallia.

KUVA 1 Koneoppimismallit (mukaillen Haltu 2024b; Sarker 2021, 3–4).

2.2.2 Syväoppiminen

Syväoppiminen (eng. Deep Learning) on koneoppimisen alalaji, joka käyttää monikerroksisia keinotekoisia neuroverkkoja tietojen analysointiin ja käsittelyyn. Nämä neuroverkot koostuvat useista erilaisista toisiinsa kytketyistä solmukerroksista, joista jokainen oppii tunnistamaan piirteitä syötettävästä datasta. (ChatGPT-verkkosivusto 2024.)

Ensimmäinen kerros havaitsee yksinkertaisia ominaisuuksia kuten viivoja tai värejä ja välittää käsittelemänsä tiedot seuraavalle kerrokselle. Jokainen neuroverkon seuraava kerros rakentaa uuden kerroksen edellisen kerroksen oppiman tiedon päälle havaitsemalla yhä abstraktimpia ja monimutkaisempia piirteitä. Lopulta viimeinen kerros antaa mallin lopullisen ennusteen tai luokittelun perustuen aiempien kerrosten tuottamaan tietoon. (Goodfellow ym. 2016, 166–170.)

Koneoppimismallit tarvitsevat jäsenneltyä ja hyvin merkittyä dataa tarkkojen ennusteiden ja päätösten tekemiseen. Mikäli dataa ei ole merkitty oikein tai järjestelty, tavalliset koneoppimismallit eivät pysty käsittelemään sitä tarkasti ja silloin siitä tulee syväoppimisen alue. Syväoppiminen on koneoppimista kehittyneempi lähestymistapa, sillä monitasoisen rakenteensa vuoksi se pystyy tunnistamaan hienovaraisempia piirteitä datasta. Tämä tekee siitä tehokkaan monimutkaisissa tehtävissä esimerkiksi kuvantunnistuksessa ja luonnollisten kielten käsittelyssä. Syväoppimisen hyödyntäminen vaatii kuitenkin suurta laskentatehoa ja suuria määriä dataa, mikä voi tehdä sen hyödyntämisestä haastavaa. (Unite.AI 2023.)

2.3 Kielimallit

Kielimallit (eng. Language Models) ovat tekoälyyn perustuva järjestelmä, mitkä on kehitetty analysoimaan ja tuottamaan inhimillistä kieltä. Näitä malleja käytetään kielen kääntämisessä, vastausten tuottamisessa annetun syötteen perusteella sekä ennustamaan, että mikä sana todennäköisimmin seuraa tiettyä lauseketta. Kielimallit hyödyntävät valtavia datamääriä ja kehittyneitä neuroverkkoja. Kielimallien avulla ne pystyvät ymmärtämään tekstin rakenteita, merkityksiä ja asiayhteyksiä. (Feuerriegel ym. 2023. 114–116.)

Kielen tuottamiseen ja käsittelemiseen luotuja generatiivisia tekoälyratkaisuja kutsutaan suuriksi kielimalleiksi (eng. Large Language Models). Suuret kielimallit ovat neuroverkkopohjaisia kielimalleja, jotka on koulutettu suurilla datamäärillä ja niiden toiminta perustuu syväoppimiseen. Tämä tarkoittaa sitä, että kielimalli hyödyntää ja kehittää itsenäisesti eräänlaista ihmisen aivojen toimintaa jäljittelevää järjestelmää, mikä oppii käsillä olevia sisältöjä, merkityksiä, ja niiden asiayhteyksiä sekä kykenee tuottamaan, ymmärtämään, ja kääntämään tekstiä oppimansa perusteella. (Feuerriegel ym. 2023. 114.)

Suurista kielimalleista tunnettuja ovat esimerkiksi Open AI:n GPT-3 (eng. Generative pre-trained transformer 3) ja Googlen Bard (Kähärä 2024; Haltu 2024b). Näitä malleja käytetään erilaisissa sovelluksissa kuten palveluroboteissa, virtuaaliavustajissa, ja kielenkäännöspalveluissa.

3 AUTOMAATIOTESTAUS

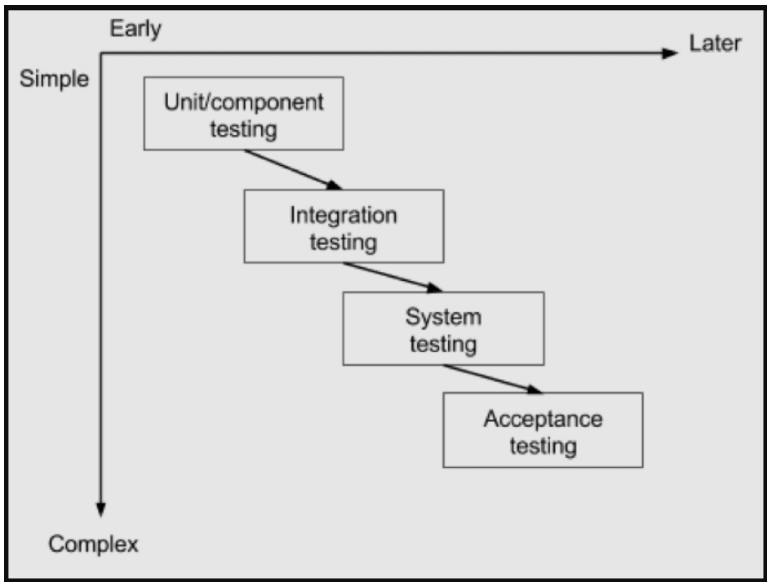
Ohjelmistotestaus on tapa varmistaa ohjelman laatu ja tunnistaa mahdolliset viat ja häiriöt ennen ohjelmiston tai sen uusien versioiden käyttöönottoa. Ohjelmistotestausta voidaan tehdä kahdella tavalla: manuaalisesti ja automaattisesti.

Manuaalisen testauksen suorittaa ihminen, joka istuu tietokoneen ääressä ja tekee testit käsin ennalta määriteltyjen testivaiheiden mukaan. Automaatiotestaus taasen tarkoittaa testien suorittamista automaattisesti ohjelmistojen avulla.

Automaatiotestauksessa käytetään erilaisia ohjelmistotyökaluja, jotka suorittavat testit ja vertaavat testituloksia odotettuihin tuloksiin sekä raportoivat mahdolliset poikkeamat. (Hamilton 2024.) Automaatiotestauksen etuja ovat sen nopeus, toistettavuus ja kyky käsitellä suuria testimääriä lyhyessä ajassa. Haittapuolena on alussa tarvittava investointi työkaluihin ja testiympäristöjen kehittämiseen.

3.1 Testitasot

Ohjelmiston testaaminen voidaan jakaa eri testitasoihin, jotka kohdistuvat ohjelmiston eri osiin ja ohjelmiston kehityksen eri vaiheisiin. Testitasot voidaan jakaa neljään eri kategoriaan suurimmasta ja yksinkertaisimmasta kokonaisuudesta pienimpään ja monimutkaisimpaan kokonaisuuteen.



KUVA 2 Testaustasot (Testsigma 2023).

3.1.1 Yksikkötestaus

Yksikkötestaus (eng. Unit Testing) on alhaisin testauksen taso, jota aletaan tehdä heti ohjelmiston kehityksen alkumetreiltä. Testeillä testataan koodin yhden osan toimivuutta kerrallaan. Yksikkötestien määrä on suuri koska tarkoituksena testeillä on varmistaa, että koodin pienimmätkin erilliset osat toimivat tarkoitetulla tavalla. (Vala 2022.)

Yksikkötestauksen suorittaa kehittäjä sitä mukaa, kun kirjoittaa varsinaista ohjelmakoodia koodialustalla. Tyypillisesti yksikkötestit kirjoitetaan sen jälkeen, kun varsinaista koodia on valmistunut. Joissain ohjelmistokehitysmalleissa kuten ketterässä kehityksessä automatisoitavien yksikkötestien kirjoittaminen voi tapahtua jo ennen sovelluksen koodin kirjoittamista. (ISBQT 2018, 27–28.)

3.1.2 Integraatiotestaus

Integraatiotestaukseen (eng. Integration Testing) siirrytään siinä vaiheessa, kun kaikki yksikkötestit on läpäisty hyväksytysti. Integraatiotestausvaiheessa keskitytään koodin eri osien tai järjestelmien välisen vuorovaikutuksen testaamiseen, jotta saadaan selville, toimivatko ohjelmiston osat tarkoitetulla tavalla yhdessä. (Vala 2022.) Integraatiotestaus voidaan jaotella kahteen eri tasoon: komponentti-integraatiotestaus, joka keskittyy yhteen liitettyjen ohjelmiston palasien vuorovaikutuksiin ja rajapintoihin. Tämä on yleensä ohjelmistokehittäjien vastuulla. Järjestelmäintegraatiotestaus

keskittyy järjestelmien, pakettien ja mikropalveluiden välisiin vuorovaikutuksiin ja rajapintoihin sekä on yleensä testaajien vastuulla. (ISBQT 2018, 28–30.)

3.1.3 Järjestelmätestaus

Järjestelmätestauksessa (eng. System Testing) keskitytään koko järjestelmän testaamiseen, jolloin tutkitaan täyttääkö järjestelmä sille asetetut vaatimukset. Tässä vaiheessa kaikki ohjelmiston osat ja integraatiot on yhdistetty ja testaus pyritään suorittamaan lopullista kohde- tai tuotantoympäristöä vastaavassa tilassa. Järjestelmätestausta tekevät testaajat joko manuaalisesti tai automatisoidusti erilaisten testityökalujen avulla riippuen siitä, minkä tyyppistä järjestelmätestausta testaaja suorittaa. (Vala 2022; ISBQT 2018, 31–32.)

3.1.4 Hyväksymistestaus

Testitasomallin ”alimmalla portaalla” on hyväksymistestaus (eng. Acceptance Testing). Tässä vaiheessa testataan valmista tai lähes valmista tuotetta. Sen tarkoituksena on varmistaa, että ohjelmisto täyttää ennalta määritellyt hyväksymiskriteerit ja on valmis loppukäyttäjien käyttöön. Hyväksymistestaus voidaan myös jaotella eri muotoihin riippuen siitä, mitä ohjelmiston kriteeriä tai toiminnallisuutta halutaan arvioida.

Yleisimmin hyväksymistestaus on toiminnallista testausta, jossa testataan ohjelmiston toimivuutta määriteltyjen skenaarioiden mukaan. Tämä testaus on yleensä manuaalista. Sitä voidaan myös automatisoida erityisesti silloin, jos halutaan toistaa samat testit useissa eri käyttöympäristöissä tai ohjelman eri versioissa. Automatisoidut testit ovat pitkiä ja monimutkaisia koska niillä testataan kokonaisen ohjelmiston toimivuutta. Hyväksymistestausta tekevät tyypillisimmin ohjelmiston tilaaja, loppukäyttäjät tai riippumaton testausryhmä. (ISBQT 2018, 32–35; Vala 2022.)

3.2 Automaatiotestityökalujen valinta

Sopivan automaatiotestityökalun valinta on tärkeä osa testausprosessin onnistumista, koska automaatiotestauksen avulla voidaan havaita virheitä, jotka saattavat jäädä huomaamatta manuaalisessa testauksessa. Oikean työkalun avulla voidaan myös tehostaa testausta, vähentää inhimillisten virheiden riskiä sekä nopeuttaa ohjelmiston julkaisuaikataulua. Lisäksi automaatiotestauksen

käyttö parantaa ohjelmiston laatua ja auttaa varmistamaan, että mahdolliset ongelmat havaitaan aikaisessa vaiheessa, mikä säästää aikaa ja kustannuksia ohjelmistokehitysprosessin aikana. ISTQB-dokumentissa (2018, 74) korostetaan, että automaatiotestauksen työkalujen tulisi tukea nopeaa ja helppoa skriptien luontia sekä ylläpitoa, mikä auttaa minimoimaan kehitysaikaa ja virheenkorjausprosesseja.

Aburaksen (2024, 31–35) tutkimuksessa, jossa vertailtiin eri automaatiotestityökaluja, koottiin testityökalun valintaa varten arviointikriteerit, joita käytettiin apuna työkalun arvioinnissa. Näitä kriteerejä voidaan soveltaa valittaessa työkalua, ja ne sisältävät muun muassa seuraavat tekijät:

- Testauksen taso ja tyyppi: Määritetään, mihin testitasoon työkalu parhaiten sopii. Esimerkiksi JUnit ja NUnit tukevat yksikkötestausta, kun taas Robot Framework ja Selenium ovat paremmin järjestelmätestaukseen sopivia.
- Kehitysympäristö: Valitun työkalun on oltava yhteensopiva käytössä olevien ohjelmistokehitysympäristöjen kanssa.
- Ohjelmointikieli: Työkalun tulisi olla yhteensopiva käytettävien ohjelmointikielien kanssa.
- Kustannukset: Onko kyseessä avoimen lähdekoodin työkalu vai kaupallinen työkalu? Avoimen lähdekoodin työkalut ovat maksuttomia, mutta kaupalliset työkalut tarjoavat laajemman tuen ja toiminnot.

Aburaksen tutkimus (2024, 31–35) korostaa, että automaatiotestityökalun valinnassa on usein tehtävä kompromisseja. Yhtä kaikille sopivaa työkalua ei ole, joten työkalun valinta on tehtävä projektin erityistarpeet huomioiden. Esimerkiksi monimutkaisia, monitasoisia järjestelmiä varten voi olla tarpeen käyttää yhdistelmää useista eri työkaluista, jotka tukevat erilaisia testausvaiheita.

3.3 Tekoäly automaatiotestauksessa

Ohjelmistotestaus on alana jatkuvassa muutoksessa. Kun tavoitteena on laadun varmistaminen ja virheiden varhainen tunnistaminen, tulee testausmenetelmien reagoida herkästi ohjelmistoalan kehitykseen (Haaramo 2020, 4–5).

Tekoälyn rooli automaatiotestauksessa kasvaa, samalla kun erilaiset tekoälyteknologiat kehittyvät ja yleistyvät. Tekoälypohjaisilla ratkaisuilla voidaan helpottaa kehittäjien ja testaajien työkuormaa.

Erilaisia tekoälypohjaisia ratkaisuja automaatiotestaukseen ovat muun muassa ohjelmistot, jotka suorittavat testejä itsenäisesti, tunnistavat bugeja automaattisesti tai ylläpitävät ja päivittävät testiskriptejä ilman manuaalista työtä. (Thiam 2023.)

Foggin (2021) blogikirjoituksessa tuodaan esille tekoälyn tarjoamia konkreettisia etuja automaatiotestaukseen. Tekoälyn avulla voidaan esimerkiksi generoida uusia testitapauksia, analysoida testituloksia ja lokitiedostoja tehokkaammin kuin manuaalisilla menetelmillä. Lisäksi tekoäly pystyy seuraamaan ohjelmiston toimintaa reaaliajassa, mikä mahdollistaa sen, että virhetilanteisiin voidaan reagoida nopeammin. Tämä tekee testausprosessista kattavamman ja vähentää inhimillisten virheiden riskiä joka lopulta parantaa yleistä ohjelmiston laatua.

Hyödyt korostuvat erityisesti suurissa ja monimutkaisissa ohjelmistoprojekteissa, joissa sekä datan määrä että ohjelmiston monimutkaisuus lisääntyvät kehityksen myötä. Koneoppimisen ja ennakoivan analytiikan avulla tekoäly voi analysoida suuria määriä testidataa, tunnistaa kehityssuuntia ja oppia ohjelmiston käyttäytymismalleista. Testauksesta tulee tarkempaa ja paremmin ennakoitavaa, jonka myötä virheet voidaan havaita ja korjata ennen kuin ne pääsevät vaikuttamaan tuotantoon tai ohjelmiston loppukäyttäjiin. (Fogg 2021.)

Tekoäly ei ole rajoittunut vain varsinaisiin automaatiotestiohjelmistoihin. Testauksessa sitä voidaan hyödyntää jo ohjelmistokehitysvaiheessa käyttämällä erilaisia kehitysalustoille integroitavia tekoälypohjaisia koodiavustajia. Koodiavustaja on työkalu, joka auttaa kirjoittamaan, optimoimaan, ennakoimaan ja korjaamaan koodia jo sen kirjoittamisvaiheessa. Sitä voi hyödyntää pyytämällä luomaan yksikkötestejä koodille, kirjoittamaan dokumentaation tai löytämään virheet koodista jo kehitysvaiheessa. Sitä voidaan käyttää myös automaatiotestauksen tukena silloin, kun kehitetään automaatiotestejä testiautomaatiokehyksen kuten Robot Frameworkin tai Robot Frameworkiin integroitavien kirjastojen avulla. (Swimm Team 2024.)

4 SOSIAALI- JA TERVEYDENHUOLLON TIETOJÄRJESTELMÄT

Digitaaliset sosiaali- ja terveysalan tietojärjestelmät ovat olennainen osa nykyistä terveydenhuollon palvelujärjestelmää. Näitä tietojärjestelmiä ovat muun muassa Kanta-palvelut (valtakunnallinen tietojärjestelmäpalvelu), asiakastietojen välityspalvelut, reseptijärjestelmät ja terveydenhuollon potilastietojärjestelmät (Valvira 2024). Järjestelmien tarkoituksena on parantaa hoidon laatua, tehostaa työnkulkuja sekä varmistaa potilas- ja asiakastietojen turvallinen ja luotettava hallinta.

Tietojärjestelmien luokittelu ja vaatimustenmukaisuus perustuvat asiakastietolakiin sekä Terveyden ja hyvinvoinnin laitoksen (THL) määräyksiin. Järjestelmillä on niiden käyttötarkoituksen mukaiset erityisvaatimuksensa, jotka määrittelevät muun muassa tietoturvallisuuden ja yhteen toimivuuden toteutumista. Vaatimusten täyttämisestä ja niiden ylläpidosta vastaa tietojärjestelmäpalvelun tuottaja, vaatimustenmukaisuuden toteutumista valvovat viranomaiset kuten Valvira ja Traficom. (Valvira 2024.)

4.1 A- ja B-luokan tietojärjestelmät

Sosiaali- ja terveydenhuollon tietojärjestelmät luokitellaan A- ja B-luokan järjestelmiin käyttötarkoituksen mukaan. A-luokan järjestelmät on tarkoitettu liitettäviksi Kanta-palveluun, kun taas B-luokan järjestelmiä käytetään vain paikallisesti tai alueellisesti (Mediconsult 2024).

A-luokan tietojärjestelmät jaotellaan edelleen alaluokkiin A1, A2 ja A3 tietojärjestelmän riskitason, kriittisyyden, käyttötarkoituksen, käsiteltyjen asiakastietojen tietojen luonteen ja laajuuden perusteella sekä Kanta-liitettävyyden mukaan. (Valvira 2024; Terveyden ja hyvinvoinnin laitos 2024.) Esko Systems Oy:n tuottama Esko-järjestelmä luokitellaan A3-luokan tietojärjestelmäksi, koska se on Kanta-palveluihin liitetty potilastietojärjestelmä, jossa käsitellään arkaluonteisia potilas- ja asiakastietoja korkealla riskitasolla.

Tietojärjestelmän luokka vaikuttaa myös siihen, miten järjestelmän olennaiset vaatimukset todennetaan eli millaisia sertifioinnin ja rekisteröinnin toimenpiteitä järjestelmälle on suoritettava. Olennaiset vaatimukset liittyvät tietojärjestelmän toiminnallisuuteen, yhteen toimivuuteen ja tietoturvallisuuteen. A3-luokan järjestelmille vaaditaan sertifiointiprosessi, joka sisältää yhteistestauksen

Kelan Kanta-palveluiden kanssa sekä tietoturvallisuuden auditoinnin Traficomin hyväksymän arviointilaitoksen toimesta. (Terveyden ja hyvinvoinnin laitos 2024.)

Korkean riskin järjestelmille asetetaan vaatimuksia myös jatkuvan tietoturvan valvonnan, säännöllisten auditointien ja sertifiointien osalta. Tietoturvavaatimukset perustuvat kansallisiin ja kansainvälisiin säädöksiin, kuten EU:n tietosuoja-asetukseen (GDPR) ja ISO 27001 -standardiin. Suomessa Valvira ja Kela valvovat järjestelmien tietoturvaa ja yhteensopivuutta yhdessä Kanta-palveluiden kanssa. (Kanta 2024.; Valvira 2024; Tolonen & Vepsäläinen 2021).

4.2 Lääkinnälliset laitteet ja ei-lääkinnälliset laitteet

EU:n lääkinnällisten laitteiden asetuksen eli MDR:n (EU/2017/745) mukaan lääkinnälliseksi laitteeksi luokitellaan esimerkiksi instrumentti, laitteisto, väline tai ohjelmisto, jonka valmistaja on tarkoittanut käytettäväksi ihmisillä, joko yksinään tai yhdistelminä, sairauden diagnosointiin, ehkäisyyn, ennakointiin, ennusteen laatimiseen, tarkkailuun, hoitoon tai lievitykseen. Näihin laitteisiin kuuluvat esimerkiksi sydämentahdistimet, verensokerimittarit ja röntgenlaitteet (Fimea 2024a).

Ohjelmisto luokitellaan lääkinnälliseksi laitteeksi, jos se vaikuttaa suoraan terveydentilan seurantaan tai hoitoon. Vaatimuksia sovelletaan myös lääkinnällisiä laitteita ohjaaviin tai niiden toimintaan vaikuttaviin erillisiin ohjelmistoihin. (Fimea 2024b.)

Lääkinnällisiltä laitteilta vaaditaan tiukkoja turvallisuus- ja suorituskykyominaisuuksia, jotka MDR-asetus ja kansallinen lainsäädäntö määrittelevät. Asetus edellyttää valmistajilta kattavaa dokumentointia, laitteen turvallisuuden ja suorituskyvyn osoittamista sekä viranomaishyväksyntää ennen laitteen markkinoille saattamista. MDR:n 8 artiklassa määritellään yhdenmukaistettujen standardien käyttö mutta niiden harmonisointi on vielä kesken. MDR edellyttää kuitenkin uusinta tekniikkaan ja valmistajia kehotetaan noudattamaan standardien uusinta versiota. (EUR-Lex 2017; Hurskainen 2024.)

ISO 13485 -standardi on viitekehys, jota lääkinnällisten laitteiden valmistajien tulee noudattaa laadunhallintajärjestelmän rakentamisessa ja ylläpidossa. Standardi korostaa, että valmistajan on varmistettava kaikkien prosessien, työkalujen ja järjestelmien asianmukainen todentaminen. Tämä

koskee myös ohjelmistoja, joita käytetään lääkinnällisen laitteen kehittämisessä ja ylläpidossa. (Hurskainen 2024.)

Lisäksi ISO 13485 -standardia täydentävät seuraavat alistandardit, jotka ohjaavat lääkinnällisen laitteen kehittämistä:

- ISO 14971: Lääketieteelliset laitteet: Riskienhallinnan soveltaminen lääkinnällisiin laitteisiin
- IEC 62304: Lääketieteellisten laitteiden ohjelmistot: Ohjelmiston elinkaari
- IEC 62366: Lääketieteelliset laitteet: Käytettävyystekniikan soveltaminen lääkinnällisiin laitteisiin
- IEC 82304-1: Terveysohjelmistot, Osa 1: Yleiset tuoteturvallisuusvaatimukset
  (Esko Systems Oy 2024.)

Ei-lääkinnälliset laitteet puolestaan ovat terveydenhuollossa käytettäviä laitteita, jotka eivät täytä lääkinnällisen laitteen määritelmää. Näihin voi kuulua esimerkiksi toimistotarvikkeita, kuten tietokoneita ja tulostimia, joita käytetään potilastietojen hallinnassa, mutta jotka eivät suoraan vaikuta potilaan terveyteen. Vaikka näille laitteille ei aseteta yhtä tiukkoja vaatimuksia kuin lääkinnällisille laitteille, niiden on silti täytettävä tietyt standardit, kuten tietoturvaa ja tietosuojaa koskevat määräykset.

4.3 Sosiaali- ja terveydenhuollon tietojärjestelmien kehittäminen

Sosiaali- ja terveydenhuollon tietojärjestelmät ovat työkaluja, joiden avulla pyritään parantamaan terveyttä, hyvinvointia ja sairauksien hoitoa. Potilastietojärjestelmät tallentavat ja käsittelevät valtavia määriä arkaluontoista tietoa, ja niiden toimivuus vaikuttaa suoraan potilaiden saaman hoidon laatuun sekä hoitohenkilökunnan työskentelyn sujuvuuteen. Järjestelmien kehittäminen edellyttää monialaista yhteistyötä, koska se vaatii kokonaisvaltaista ymmärrystä terveydenhuollon toimintakulttuurista, lainsäädännön vaatimuksista ja teknologian tarjoamista mahdollisuuksista. Kehittämiseen osallistuvat niin hoitoalan ammattilaiset kuin teknologian ja lainsäädännön asiantuntijatkin. (Rytkönen, J. 2022. 132–134.)

Nopeasti kehittyvä teknologia asettaa omat haasteensa järjestelmien suunnittelulle ja käytettävyydelle. Teknologian muutosten myötä järjestelmien on pysyttävä mukana muutoksessa, mikä

asettaa omat vaatimuksensa kehitystyölle. Tulevaisuuden tarpeet on myös otettava huomioon jo kehitysvaiheessa, sillä esimerkiksi tekoäly, koneoppiminen, ennakoiva analytiikka ja robotiikka integroituvat yhä tiiviimmin osaksi terveydenhuollon tietojärjestelmiä.

Tekoälyä hyödyntävät diagnosointityökalut ja potilaiden hoitopolkua tukevat järjestelmät tarjoavat etuja ja vaativat toimiakseen järjestelmiä, jotka kykenevät käsittelemään ja analysoimaan suuria määriä dataa reaaliajassa. Järjestelmien on oltava joustavia ja skaalautuvia, jotta ne voivat mukautua uusiin teknologisiin innovaatioihin ja pystyvät vastaamaan muuttuviin tarpeisiin. (Sosiaali- ja terveysministeriö 2023. 19–23.)

5 TEKOÄLYPOHJAISEN KOODIAVUSTAJAN VALINTAPROSESSI

Tekoälypohjaisen koodiavustajan valintaprosessin aihe sai alkunsa Esko Systems Oy toimeksiantona. Yrityksen tavoitteena oli löytää työkaluja automaatiotestauksen tueksi ja kartoittaa, että löytyisikö tähän sopivia tekoälypohjaisia tuotteita. Alun perin tarkoituksena oli tutkia tekoälypohjaisia testityökaluja ja selvittää, olisiko niiden joukossa automaatiotestaukseen sopivia vaihtoehtoja.

Haasteiksi varsinaisen tekoälypohjaisen testityökalun testikäytössä nousivat erityisesti tietoturvakysymykset, sosiaali- ja terveysalan tietojärjestelmien erityisvaatimukset ja lääkinnällisiä laitteita koskevat säädökset, koska kyseessä oli sosiaali- ja terveydenhuollon tietojärjestelmä, jonka sovelluksista osa luokitellaan lääkinnällisiksi laitteiksi. Näissä ympäristöissä tietoturva- ja yksityisyysvaatimukset ovat tiukat, ja tekoälypohjaisten testityökalun testikäyttö herätti kysymyksiä siitä, missä ja miten dataa käsitellään sekä säilytetään tekoälypohjaisissa sovelluksissa.

Toinen merkittävä haaste oli toimivien testiympäristöjen puute projektin alussa. Testiympäristöt olivat vielä kehitysvaiheessa, mikä hidasti automaatiotestauksen työkalujen kokeilua käytännössä. Koska testiympäristöjä ei ollut vielä valmiina, ei tekoälypohjaisia testityökaluja olisi päästy luotettavasti testaamaan todellisissa olosuhteissa, mikä teki alkuperäisen suunnitelman toteuttamisen haastavaksi.

Aiheeseen syventymisen ja käytännön realiteettien selkiytyessä päätettiin keskittyä tekoälypohjaisen koodiavustajan valintaan. Koodiavustajalla voisi helpottaa kehittäjien työtä yksikkötestauksessa, jossa yksittäisiä koodikomponentteja testataan, sekä tukea testaajia automaatiotestien kirjoittamisessa. Tekoälypohjainen koodiavustaja nopeuttaisi testien kirjoittamista ennakoivalla tekstinsyötöllä, pyydettäessä kirjoittaa yksikkötestit koodille ja korjaa mahdolliset virheet koodissa sekä kirjoittaa koodille dokumentaation. Tämän lisäksi koodiavustajan chat-toiminnon avulla voi pyytää muun muassa kertomaan koodin sisällön ja selittämään koodin eri osien toimintoja. Testaamisen lisäksi kehittäjät hyötyisivät koodiavustajan käytöstä myös sovellusta kehittäessä.

5.1 Koodiavustajien kartoitus

Erilaisia tekoälypohjaisia koodiavustajia kartoitettiin touko-kesäkuussa 2024 Googlen hakukoneen ja erilaisten blogien sekä koodiavustajia vertailevien arviointisivustojen kautta. Touko-kesäkuussa 2024 tyypillisimmin käytössä olleet, eniten esiin nousseet ja parhaiten arvioidut koodiavustajat olivat Tabnine, Codeium ja Github Copilot. Sen vuoksi nämä kolme koodiavustajaa vaikuttivat potentiaalisimmilta työkaluilta yrityksen tarvetta ajatellen.

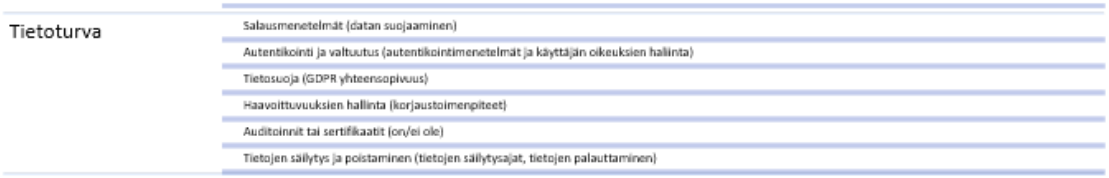
Kaikki työkalut olivat uusia ja suhteellisen tuoreita koska tekoälypohjaisten koodiavustajien kehitys on alkanut vasta viime vuosina. Tämä näkyi erityisesti siinä, että työkalujen toiminnallisuudet ja ominaisuudet perustuivat syväoppimiseen ja kehittyneisiin kielimalleihin, kuten OpenAI:n kehittämiin GPT-mallin eri versioihin.

Työkaluista GitHub Copilot on Microsoftin ja Open AI:n yhteinen tuote. Tabnine ja Codeium ovat molemmat uusien startup-yritysten tuotteita.

5.2 Koodiavustajan ominaisuuksien arviointilomakkeen luominen

Koodiavustajan ominaisuuksien arviointia varten luotiin erillinen arviointilomake (LIITE 1), jonka pohjana käytettiin Esko Systemsin omaa kelpuutuslomaketta. Kelpuutuslomakkeesta poimittiin koodiavustajan arvioinnin kannalta olennaisimmat kriteerit, jotka muokattiin vastaamaan arvioinnin tarpeita. Kelpuutuslomaketta käytettiin hyödyksi arviointilomakkeen luomisessa myös sen vuoksi että mikäli myöhemmin päädytään työkalun käyttöönottoon, niin arviointilomakkeelta löytyisivät myös kelpuutuksen kannalta olennaiset tiedot työkalusta.

Arviointilomakkeen hyväksytettiin yrityksen tietosuojapäälliköllä, jotta varmistettiin sen poimivan työkalun tietoturvaominaisuuksien kannalta olennaiset tiedot (Kuva 3). Tämän jälkeen lomake otettiin käyttöön arviointiprosessia varten.



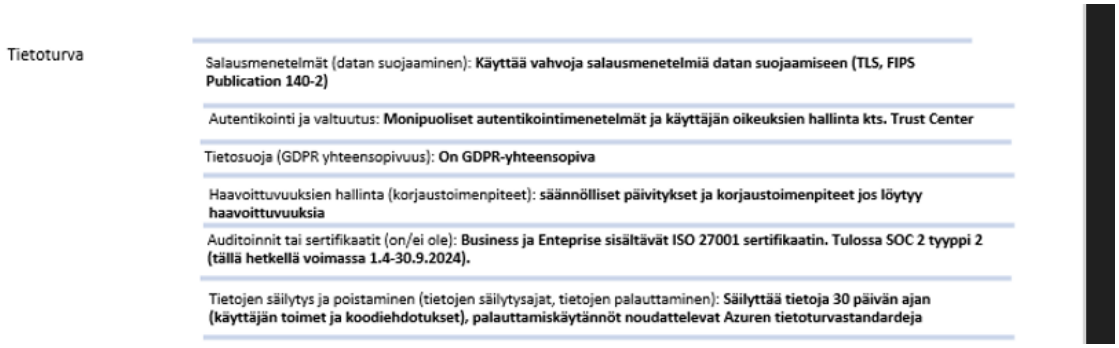
KUVA 3 Tyhjän arviointilomakkeen tietoturvaosio.

5.3 Tietoturva ja tietosuoja

Työkalujen tietosuoja ja -turva ovat olennainen osa niiden arviointia ja käyttöönottoa. Tässä tapauksessa erityisesti siksi, että kyseessä on sosiaali- ja terveysalan tietojärjestelmä.

Arvioinnin kohteena olevista työkaluista Github Copilotilla ja Tabninella, tietoturvaa ja -suojaa koskevat tiedot olivat dokumentaatiosta hyvin löydettävissä. Codeiumin dokumentaatiossa muun muassa salausmenetelmistä kertovia tietoja ei ollut saatavilla ollenkaan.

Kaikki työkalut takasivat paremmat tietosuoja- ja tietoturvaominaisuudet maksetuissa tilauksissa, kun taas ilmaiskäyttäjille ehdot olivat heikommat ja käyttäjien tiedot heikommin salattuja.



KUVA 4 Github Copilot Tietoturva ja tietosuoja

KUVA 5 Codeium Tietoturva ja tietosuoja

KUVA 6 Tabnine Tietoturva ja tietosuoja

Tämän lisäksi sovelluksen ilmaiskäyttöön sisältyy ehto, että sovellus voi käyttää ilmaiskäyttäjien koodia sovelluksen kouluttamiseen.

5.4 Yhteensopivuus olemassa olevien järjestelmien kanssa

Esko Systemsillä käytetään monipuolisesti erilaisia kehitysympäristöjä, riippuen käyttäjän mieltymyksistä ja siitä, minkälaisia teknologioita ja työkaluja henkilön toimenkuva edellyttää. Koodiavustajan valinnassa on kuitenkin olennaista, että se toimii halutussa ympäristössä ja sen kaikki ominaisuudet ovat käytettävissä ilman rajoituksia.

Arvioinnin kohteena olleet koodiavustajat; Tabnine, Codeium ja GitHub Copilot ovat yhteensopivia yleisimmin käytössä olevien kehitysalustojen kanssa. Kaikissa testatuissa koodiavustajissa oli monipuolinen kielituki yleisimmin käytössä oleville ohjelmointikielille.

Koodiavustajien dokumentaatiot olivat osin puutteellisia. Codeiumin dokumentaatio oli arviointiajankohtana puutteellinen ja vaikeasti löydettävissä verkkosivuilta, Tabninella ja Github Copilotilla dokumentaatiot olivat kattavat ja helposti löydettävissä. Tabninen ja Github Copilotin dokumentaatioista löytyivät selkeät käyttö- ja asennusohjeet eri kehitysympäristöihin, mikä helpottaa koodiavustajan käyttöönottoa ja integrointia eri kehitysympäristöihin.

5.5 Koodiavustajien käytettävyys

Koodiavustajien testikäyttö tehtiin tutkimuksen tekijän henkilökohtaisella tietokoneella Visual Studio Code - kehitysympäristössä, johon oli valmiiksi asennettuna Python- ja JavaScript- kieliä tukevat lisäosat. Kehitysympäristöä käytettiin tutkimuksen tekijän opiskelijatunnuksilla. Tällä tavoin koodiavustajien testikäyttö pystyttiin tekemään turvallisesti ilman että ohjelmilla oli pääsyä yrityksen tuottamaan dataan tai arkaluonteisiin tietoihin.

Kaikkien kolmen koodiavustajan käytettävyys testattiin yksinkertaisilla koodinpätkillä, jossa käytiin läpi koodiavustajien ennakoiva koodinsyöttö, kyky optimoida koodia, luoda dokumentaatio koodille ja luoda yksikkötestitapaukset jo olemassa olevalle koodille. Koodiavustajien palvelubottiominaisuus testattiin pyytämällä palvelubottia selittämään koodin sisältö ja koodin tekemät toiminnot. Koodiavustajien arviointilomakkeista (LIITE 2, LIITE 3, LIITE 4) löytyvät linkit jokaisesta koodiavustajasta kuvattuun videoon, joissa esitellään lyhyesti koodiavustajan käyttöä edellä mainituilla tapauksilla.

Käyttäjätuen osalta havaittiin kuitenkin puutteita. Koodiavustajilla oli aktiiviset verkkoyhteisöt ja kattavat usein kysytyt kysymykset-osiot, jotka eivät tarjonneet kunnollista henkilökohtaista tukipalvelua, kuten reaaliaikaista teknistä tukea. Tämä voi aiheuttaa haasteita erityisesti kriittisissä ongelmatilanteissa tai silloin, kun tarvitaan nopeasti ratkaisuja erikoistilanteisiin.

5.6 Koodiavustajan valinta

Arviointilomakkeiden vertailun ja koodiavustajien testikäytön jälkeen Githbub Copilot päädyttiin valitsemaan yrityksen tarpeita parhaiten vastaavaksi työkaluksi. Github Copilot oli käytettävyydeltään toiseksi paras mutta työkalun arviointilomakkeelle kerättyjen taustatietojen perusteella tuote oli luotettavin ja tarjosi laadukkaimman dokumentaation.

Valintaan vaikuttivat erityisesti seuraavat tekijät:

1. Luotettavuus ja jatkuva kehitys: GitHub Copilotia kehittävät Microsoft ja OpenAI, jotka ovat johtavia sekä vakaita toimijoita tekoäly- ja teknologiasektorilla. Tämä varmistaa todennäköisimmin sovelluksen jatkuvat päivitykset ja tietoturvaparannukset sekä mahdollistaa sen, että mikäli tarvitaan vielä tarkempia määrityksiä koskien tietoturvaa tai tietojen suojaamista, niin sopimuksien ehdot voivat olla paremmin neuvoteltavissa.
2. Dokumentaatio ja tukipalvelut: Työkalu tarjoaa kattavan ja hyvin jäsennellyn dokumentaation, sekä käyttäjätuen, joka mahdollistaa avun mahdollisissa ongelmatilanteissa, mutta ei kuitenkaan vastaa ongelmatilanteisiin reaaliajassa.

3. Pitkäaikainen skaalautuvuus: Github Copilotin ominaisuudet mahdollistavat teknologian hyödyntämisen myös pitkällä aikavälillä ja riski sovelluksen kaatumiselle on pieni, kun taustalla ovat alan vakaat toimijat.

5.7 Päätöksenteko ja käyttöönotto

Valintaprosessin päätteeksi ehdotettiin, että Github Copilot otettaisiin testikäyttöön rajatulle käyttäjäryhmälle, johon kuuluu muutamia ohjelmistokehittäjiä ja automaatiotestaajia. Testikäytön ajankohta ajoittuu talvelle ja keväälle 2025, kunhan Esko Systemsin sisäisiä dokumentaatioita on päivitetty ja rakenteilla oleva tekoälypolitiikka on hyväksytty. Testikäytön tavoitteena on arvioida työkalun tuomat hyödyt käytännössä, mahdolliset haasteet sekä vaikutukset työnkulkuun ja tuottavuuteen.

Testikäytön aikana myös kerätään palautetta käyttäjiltä. Näin voidaan löytää mahdolliset ongelmakohdat ja kehitystarpeet.

Testijakson lopussa järjestetään arviointi, jossa käydään läpi kerätty palaute ja analysoidaan testikäytön aikana kerätyt tiedot. Tämän perusteella arvioidaan, että täyttääkö työkalu sille asetetut tavoitteet. Mikäli työkalu täyttää sille asetetut tavoitteet, tehdään päätös siitä, otetaanko GitHub Copilot laajemmin koko yrityksen sisäiseen käyttöön.

6 POHDINTA

Opinnäytetyön tutkimuskysymykseen sain vastauksen työn edetessä ja työn etenemisen myötä ymmärryksen kasvaessa siitä, kuinka vaativaa sosiaali- ja terveydenhuollon ohjelmistojen kehittäminen on. Erilaisten lakien ja standardien tunteminen sekä noudattaminen on tärkeää, jotta tuotteen kehitysprosessi täyttää viranomaisvaatimukset ja vastaa laadunhallinnan asetuksia.

Sosiaali- ja terveydenhuollon tietojärjestelmien erityisvaatimukset eivät suoraan vaikuta koodiavustajien valintaan, mutta niiden käyttöönottoa määrittelevät erilaiset säännöt ja standardit. Tämä johtuu siitä, että koodiavustajaa hyödynnetään kehitystyössä, jossa lopullinen tuote luokitellaan lääkinnälliseksi laitteeksi.

Ohjelmistokehityksessä käytettävät koodiavustajat eivät käsittele potilastietoja, mutta niiden käyttö lääkinnällisten laitteiden kehittämisessä edellyttää ISO 13485 -standardin mukaista laadunhallintajärjestelmää. ISO 13485 standardia sovelletaan A3-luokan tietojärjestelmiin, joita käytetään kriittisissä sosiaali- ja terveydenhuollon järjestelmissä. Standardin määrittelee, että koodiavustaja tulee kelpuuttaa GAMP 5 -menetelmän mukaisesti ennen käyttöönottoa. Kelpuutusprosessi varmistaa sen, että ohjelmistokehityksessä käytettävä työkalu täyttää tarvittavat vaatimukset ja on turvallinen ja luotettava. Kelpuuttamiseen käytetään erillistä kelpuuttamislomaketta, jota käytettiin myös työkalujen ominaisuuksien arviointilomakkeen (LIITE 1) pohjana.

MDR-asetuksen asettamat vaatimukset lääkinnällisille laitteille ja asiakastietolain edellyttämä laadunhallintajärjestelmä korostavat koodiavustajien käyttöönottoon ja käyttöön liittyvien dokumentaatio- ja kelpuutusprosessien merkitystä. Vaikka koodiavustajan ensisijainen rooli on tukea ohjelmistokehitystä ja automaatiotestausta, sen valinta ja dokumentointi vaikuttavat välillisesti koko kehitysprosessin laatuun ja viranomaishyväksynnän edellytyksiin.

Mikäli määräysten mukaisia prosesseja ei noudateta ja kelpuutusta ei suoriteta, riskinä ovat: virheiden ja puutteiden syntyminen, jotka voivat johtaa viivästyksiin kun puutteita joudutaan korjaamaan ja tarvittavat kelpuutukset suorittamaan jälkikäteen, lisäkustannuksiin kun tehdään uudet testaukset ja päivitetään dokumentteja, viranomaiset voivat huomauttaa mikäli vaatimuksia ei ole täytetty tai pahimmillaan laitteen markkinoillepääsyn estymiseen jos lopputuotteen ei katsota täyttävän vaadittuja standardeja ja säädöksiä. Puutteellinen prosessinhallinta heikentää myös asiakkaiden

luottamusta, millä on haitallinen vaikutus liiketoimintaan ja kilpailukykyyn. Tämä osoittaa, kuinka tärkeää on vaatimuksenmukaisuuksien noudattaminen lääkinnälliseksi laitteeksi luokiteltavan tuotteen kehityksen kaikissa vaiheissa.

Opinnäytetyön loppuraportin kirjoittamisen aikana osa tekoälypohjaisten koodiavustajien arviointilomakkeisiin kerätyistä tiedosta oli jo ehtinyt vanhentua ja muuttua uusien sovellusversioiden ja -päivitysten myötä. Tämä havainnollistaa hyvin sitä, miten nopeaa tekoälysovellusten kehitys tällä hetkellä on. Nopea kehitys tarjoaa käyttäjille jatkuvasti uusia ja kehittyneempiä versioita mutta samalla se vaatii käyttäjäorganisaatiolta ketteryyttä ja säännöllistä päivitysten tarkastelua, jotta käytössä olevien työkalujen dokumentaatio säilyy ajantasaisena.

Työn aikana hyödynsin itsekin tekoälyä monin tavoin, kuten opinnäytetyön rakenteen suunnittelussa, sisältöjen muokkaamisessa ja ideoiden etsimisessä. Tämä konkretisoi tekoälyn käytön monipuolisuutta ja sen sovellettavuutta monenlaisiin tehtäviin. Kokemus tekoälyn käytöstä vahvisti ymmärrystäni sen mahdollisuuksista ja myös rajoitteista, erityisesti kun tekoälytyökaluja halutaan hyödyntää erityisvaatimuksia edellyttävissä sovelluksissa tai niiden kehittämisessä.

Tulevaisuudessa erilaisten tekoälypohjaisten ratkaisujen rooli ohjelmistokehityksessä ja automaatiotestauksessa todennäköisesti kasvaa entisestään, kun tekoälyn kehittyessä ohjelmistokehityksen menetelmät monipuolistuvat. Tuleva kehitys tarjoaa merkittäviä mahdollisuuksia, mutta samalla se tuo mukanaan uusia muutostarpeita lakeihin ja asetuksiin, että pystytään pysymään tekoälyn kehityksessä mukana eritysvaatimuksia, sääntöjä ja standardeja noudattavien tietojärjestelmien kehitystyössä.

LÄHTEET

Aburas, Ali 2024. Choosing the right automated software testing tools. IEEE Xplore. Julkaisupäivä 30.7.2024. Hakupäivä 15.8.2024. IEEE-tietokanta. Vaatii käyttöoikeuden.

Bostrom, Nick & Müller, Vincent C. 2014. Future Progress in Artificial Intelligence: A Survey of Expert Opinion. Hakupäivä 1.8.2024. https://www.nickbostrom.com/papers/survey.pdf

CGI 2024. Tekoälyn perusteet. Hakupäivä 1.5.2024. https://www.cgi.com/fi/fi/blogi/tekoalyn-perusteet

ChatGPT-verkkosivusto Suomi 2024. Koneoppimisen ja syväoppimisen vertaileva analyysi. Chat GPT Suomi -tiimi 6.4.2024. Hakupäivä 4.8.2024. https://chat-gpt-suomi.fi/koneoppiminen-vs-syvaoppiminen/

Esko Systems Oy 2024. Lait, asetukset ja standardit. Sisäinen laadunhallintajärjestelmän dokumentti.

EUR-Lex 2017. Euroopan parlamentin ja neuvoston asetus (EU) 2017/745, annettu 5 päivänä huhtikuuta 2017, lääkinnällisistä laitteista, direktiivin 2001/83/EY, asetuksen (EY) N:o 178/2002 ja asetuksen (EY) N:o 1223/2009 muuttamisesta sekä neuvoston direktiivien 90/385/ETY ja 93/42/ETY kumoamisesta (ETA:n kannalta merkityksellinen teksti.). Dokumentti 32017R0745. Euroopan Unioni. Hakupäivä 1.12.2024. https://eur-lex.europa.eu/legal-content/FI/TXT/?uri=CELEX%3A32017R0745

Euroopan parlamentti 2020. Mitä on tekoäly ja mihin sitä käytetään? Julkaistu 4.9.2020. Päivitetty 20.6.2023. Hakupäivä 20.10.2024. https://www.europarl.europa.eu/topics/fi/article/20200827STO85804/mita-tekoaly-on-ja-mihin-sita-kaytetaan

Feuerriegel Stefan, Hartmann Jochen, Janiesch Christian & Zschech Patrick 2023. Generative AI. Business & Information System Engineering. Julkaistu 12.9.2023. https://link.springer.com/article/10.1007/s12599-023-00834-7

Fimea 2024a. Lääkinnällisen laitteen määritelmä. Hakupäivä 28.11.2024. https://fimea.fi/laakinnalliset_laitteet/mita-ovat-laakinnalliset-laitteet-/laakinnallisen-laitteen-maaritelma

Fimea 2024b. Ohjelmistot. Hakupäivä 28.11.2024. https://fimea.fi/laakinnalliset_laitteet/mita-ovat-laakinnalliset-laitteet-/erikoislaiteryhmat/ohjelmistot

Fogg, Erik 2021. The Impact of AI on Software Quality Assurance. Embedded Computing Design. Blogikirjoitus 12.2.2021. Hakupäivä 8.8.2024. https://embeddedcomputing.com/technology/ai-machine-learning/ai-dev-tools-frameworks/the-impact-of-ai-on-software-quality-assurance

Goodfellow Ian, Bengio Joshua & Courville Aaron 2016. Deep Learning. MIT Press. Hakupäivä 4.8.2024. http://imlab.postech.ac.kr/dkim/class/csed514_2019s/DeepLearningBook.pdf

Haaramo, Eeva 2020. Testauksen uudet trendit. Tivia News 12/2020. Hakupäivä 3.8.2024. https://tivia.fi/web/content/11963?unique=7b9bc774b55ba0b2a77b6380a7723d039343b80f

Hamilton, Thomas 2024. Automation testing. Päivitetty 19.8.2024. Hakupäivä 25.8.2024 https://www.guru99.com/fi/automation-testing.html

Haltu 2024a. Mitä on koneoppiminen? Malleja, hyötyjä ja haasteita. Blogikirjoitus 29.1.2024. Hakupäivä 3.8.2024. https://www.haltu.fi/blogi/koneoppiminen

Haltu. 2024b. Suuret kielimallit (LLM) - pohja Chat GPT:lle ja muille AI-sovelluksille. Blogikirjoitus. Julkaistu 13.2.2024. Hakupäivä 9.8.2024. https://www.haltu.fi/blogi/suuret-kielimallit-llm

Hurskainen Sinikka 2024. ISO 13485 vaatimukset Eskon laadunhallinnalle. Esko Systems Oy.

ISTQB 2018. Sertifioitu testaaja. Perustason sertifikaattisisältö. (käännösversio 10.10.2018, alkuperäisversio 4.6.2018) Hakupäivä 9.8.2024. https://tivia-jasenyhdistykset.fi/fistb-testi/wp-content/uploads/sites/30/2020/12/CTFL-2018-Sertifikaattisisalto-20181010-1-Valmis.pdf

Järvinen, Petteri 2023. Tekoäly ja minä: ihmisenä tekoälyn aikakaudella. Helsinki: Kustannusosakeyhtiö Tammi.

Kanta 2024. Sertifiointi ja olennaiset vaatimukset. Verkkosivu. Sivua päivitetty 23.9.2024. Hakupäivä 28.9.2024. https://www.kanta.fi/jarjestelmakehittajat/sertifiointi-ja-olennaiset-vaatimukset

Kähärä, Sohvi. 2024. Mitkä ihmeen suuret kielimallit? Delingua 13.3.2024. Hakupäivä 9.8.2024. https://delingua.fi/artikkeli-mitka-ihmeen-suuret-kielimallit/

Maria, Teresa 2024. Heikko tekoäly. Techopedia 3.9.2024. Hakupäivä 10.9.2024. https://www.techopedia.com/fi/sanasto/heikko-tekoaly

Mediconsult 2024. Sote-tietojärjestelmien A- ja B-luokka – tunnetko erot? Hakupäivä 3.5.2024. https://www.mediconsult.fi/blogi/sote-tietojarjestelmien-a-ja-b-luokka-tunnetko-erot

Rouse, Margaret 2024. Generatiivinen tekoäly. Techopedia. Päivitetty 3.9.2024. Hakupäivä 11.11.2024. https://www.techopedia.com/fi/sanasto/generatiivinen-tekoaly

Rytkönen Jenni, Kinnunen Ulla-Mari & Martikainen Sanna 2022. Sosiaali- ja terveydenhuollon tietojärjestelmäkehittäjien kokemuksia yhteistyöstä käyttäjien kanssa. Finnish Journal of eHealth and eWelfare 2022;14(2). Julkaistu 7.5.2022.

Sarker, Iqbal H. 2021 Machine Learning: Algorithms, real-world applications and research directions. SN Computer Science (2021) 2:160. Julkaistu 22.3.2021.

Saygin Pinar Ayse, Cicekli Ilyas & Akman Varol 2000. Turing Test: 50 years Later. Minds and Machines 10.

Sosiaali- ja terveysministeriö 2023. Digitaalisuus sosiaali- ja terveydenhuollon kivijalaksi. Sosiaali- ja terveydenhuollon digitalisaation ja tiedonhallinnan strategia 2023–2035. Sosiaali- ja terveysministeriön julkaisuja 2023:32. Hakupäivä 28.9.2024. https://julkaisut.valtioneuvosto.fi/bitstream/handle/10024/165288/STM_2023_32.pdf?sequence=1&isAllowed=y

Swimm Team 2024. AI Code Assistants: key Capabilities and 13 Tools to Know About. AI tools for developers. Hakupäivä 8.8.2024. https://swimm.io/learn/ai-tools-for-developers/ai-code-assistants-key-capabilities-and-5-tools-to-know-about

Tekoälyn monet kasvot 2024. Koneoppiminen. Hakupäivä 1.8.2024. https://tekoalynmonetkasvot.fi/koneoppiminen/

Terveyden ja hyvinvoinnin laitos 2024. Tiedonhallinta sosiaali- ja terveysalalla. Olennaiset vaatimukset ja sertifiointi. Päivitetty 6.9.2024. Hakupäivä 23.11.2024. https://thl.fi/aiheet/tiedonhallinta-sosiaali-ja-terveysalalla/tiedonhallinnan-ohjaus/olennaiset-vaatimukset-ja-sertifiointi

Thiam, Alexander 2023. Top 18 AI Testing Tools in 2024. Code Intelligence. Hakupäivä 20.9.2024. https://www.code-intelligence.com/blog/ai-testing-tools

Tolonen Petri & Vepsäläinen Pekka 2021. ISO 27001 Tietoturvallisuuden hallintajärjestelmän kypsyysarviointi. Esitelmä Huoltovarmuuskeskus 23.12.2021.

Unite.AI 2023. Machine learning vs. deep learning - key differences. Julkaistu 6.1.2023. Hakupäivä 8.8.2024. https://www.unite.ai/machine-learning-vs-deep-learning-key-differences/

Vala 2022. Mitä on ohjelmistotestaus ja mitä hyötyä siitä on? Valagroup 10.11.2022. Hakupäivä 10.8.2024 https://www.valagroup.com/fi/blogi/mita-on-ohjelmistotestaus-ja-mita-hyotya-siita-on/

Valvira 2024. Sosiaali- ja terveydenhuollon tietojärjestelmien luokittelu. Hakupäivä 1.11.2024. https://valvira.fi/sosiaali-ja-terveydenhuolto/tietojarjestelmien-luokittelu

LIITTEET

Tyhjä arviointilomake liite 1

Tabnine arviointi liite 2

Codeium arviointi liite 3

Github Copilot arviointi liite 4

TYHJÄ ARVIOINTILOMAKE LIITE 1

Tekoälypohjaisen työkalun arviointilomake

Ohjelmiston nimi ja versio	
Ohjelmiston valmistaja	
URL ja käyttöohjeet	
Ohjelmiston käyttötarkoitus	
Tuki ja ylläpito (dokumentaatio)	
Integroitavuus ja yhteensopivuus (testausympäristöt)	
Ominaisuudet (käyttäjäystävällisyys, käyttöohjeet, raportointi/analytiikka, automaatio)	
Tietoturva	Salausmenetelmät (datan suojaaminen) Autentikointi ja valtuutus (autentikointimenetelmät ja käyttäjän oikeuksien hallinta) Tietosuoja (GDPR yhteensopivuus) Haavoittuvuuksien hallinta (korjaustoimenpiteet) Auditoinnit tai sertifikaatit (on/ei ole) Tietojen säilytys ja poistaminen (tietojen säilytysajat, tietojen palauttaminen)
Hinnoittelu (kustannukset)	
Yhteenveto (hyödyt/haitat)	
Demovideo URL	

Muuta tietoa ohjelmistosta:

Tekoälypohjaisen työkalun arviointilomake

Ohjelmiston nimi ja versio	Tabnine (v5.8.0) Viimeisin päivitys 19.7.2024
Ohjelmiston valmistaja	Tabnine Ltd. (aiemmin Codota), (käytettyTabninen kehittämiä "Deep Tabnine" gpt2-malliin perustuvaa algoritmia Public Code Algorithm ja Private Code Algorithm algoritmeja)
URL	URL: https://www.tabnine.com/
Ohjelmiston käyttötarkoitus	Tekoälyyn perustuva ohjelmointityökalu, jonka tarkoituksena on nopeuttaa ja tehostaa koodin kirjoittamista. Työkalu tarjoaa koodiehdotuksia ja täydentää koodin kirjoittamisen yhteydessä koodirivejä tai -pätkiä automaattisesti. Tarjoaa työkalun mukana myös chat ominaisuuden jossa voi pyytää esimerkiksi valmista koodia, koodiehdotuksia tai koodin korjaamista.
Tuki ja ylläpito (dokumentaatio)	Dokumentaatio: https://docs.tabnine.com/main Turvallisuus: https://trust.tabnine.com/ Tuki: https://support.tabnine.com/hc/en-us
Integroitavuus ja yhteensopivuus (testausympäristöt)	VS Code, JetBrains IDEs (IntelliJ, PyCharm, WebStorm, PhpStorm, Android Studio, GoLand, CLion, Rider, DataGrip, RustRover, RubyMine, DataSpell, Aqua, AppCode), Eclipse, Visual Studio
Ominaisuudet (käyttäjäystävällisyys, käyttöohjeet, raportointi/analytiikka, automaatio)	Käyttöohjeet: https://docs.tabnine.com/main/getting-started/quickstart#getting-started-with-tabnine Käyttäjäystävällinen ja selkeä käyttöliittymä, voi testata ilmaisella Basic tilauksella Raportointi/analytiikka: tuottaa käyttäjätilastoja jotka analysoivat Tabninen käyttöä, tuottaa tietoa koodin suorituskyvystä, analysoi koodia ja tuottaa raportteja jotka auttavat parantamaan koodin laatua Automaatio: tekee reaaliaikaisia koodiehdotuksia ja täydentää koodia perustuen käyttäjän kirjoittamaan koodiin, tunnistaa virheitä koodista
Tietoturva	Salausmenetelmät (datan suojaaminen): **E2E-salaus ja TLS** Autentikointi ja valtuutus (autentikointimenetelmät ja käyttäjän oikeuksien hallinta): **OAUTH, 2FA** Tietosuoja (GDPR yhteensopivuus): **on GDPR yhteensopiva** Haavoittuvuuksien hallinta (korjaustoimenpiteet): **säännölliset turvatestaukset ja päivitykset** Auditoinnit tai sertifikaatit: **ISO 9001, SOC 2 (type II)** Tietojen säilytys ja poistaminen (tietojen säilytysajat, tietojen palauttaminen): **ei säilytä tietoja ollenkaan**
Hinnoittelu (kustannukset)	Kolme erilaista tilausvaihtoehtoa: Basic: ILMAINEN, (käyttö rajoitettua) Pro: $12 USD/käyttäjä/kuukausi Enterprise: $39 USD/käyttäjä/kuukausi
Yhteenveto (hyödyt/haitat)	+ joustavuus (voi käyttää pilvessä, paikallisesti omalla koneella tai itse isännöidyllä palvelimella), ei jaa eikä tallenna koodia, chat ominaisuus joka mahdollistaa monipuolisemmat toiminnot - yritystilausvaihtoehto kallis, ei tue kaikkia kieliä (suppea), käyttäjän täytyy osata lukea koodia
Demovideo URL	Linkki demovideoon: tabnine test.mkv Linkki tabninen demoalustalle: ei ole omaa demoalustaa

Muuta tietoa ohjelmistosta:

Hyvä ja kattava dokumentointi. Sertikaatit, jotka varmentavat ohjelmiston turvallisuutta. Suppein kielivalikoima. Ei omaa demoversiota saatavilla, jolla ohjelmaa voisi testata.

Tekoälypohjaisen työkalun arviointilomake

Ohjelmiston nimi ja versio	Codeium v1.9.86 (viimeisin päivitys 24.7.2024)
Ohjelmiston valmistaja	Codeium Inc. (Varun Mohan and Douglas Chen), käyttää "Deep Tabnine" algoritmia joka perustuu OpenAI:n GPT-2 arkkitehtuuriin.
URL	https://codeium.com/
Ohjelmiston käyttötarkoitus	Tekoälyyn perustuva ohjelmointityökalu, jonka tarkoituksena on nopeuttaa ja tehostaa koodin kirjoittamista. Työkalu tarjoaa koodiehdotuksia ja täydentää koodin kirjoittamisen yhteydessä koodirivejä tai -pätkiä automaattisesti. Tarjoaa työkalun mukana myös chat ominaisuuden jossa voi pyytää esimerkiksi valmista koodia, koodiehdotuksia tai koodin korjaamista.
Tuki ja ylläpito (dokumentaatio)	Dokumentaatio: https://help.codeium.com/hc/en-us (suppea, ei varsinaista develper tai team tilauksen dokumentaatiota saatavilla Enterprise-tilauksen dokumentaatio: https://help.codeium.com/hc/en-us/articles/16974311736859-Welcome-to-Codeium-Enterprise-Docs-Portal Turvallisuus: https://codeium.com/security Tuki: https://help.codeium.com/hc/en-us (HUOM! tuki saatavilla Developer käyttäjille ainoastaan discordin ja githubin yhteisöjen kautta, teams ja enterprise käyttäjille tarjolla Codeiumin oma tukikeskus)
Integroitavuus ja yhteensopivuus (testausympäristöt)	VS Code, JetBrains IDEt, Neovim, Jupyter, Visual Studio jne
Ominaisuudet (käyttäjäystävällisyys, käyttöohjeet, raportointi/analytiikka, automaatio)	Selkeä ja helppokäyttöinen käyttöliittymä, helppo asentaa ja helppo käyttää. Codeiumin omalla sivulla hyvät demot ja kokeiluversio jolla voi testata työkalua. Käyttöohjeet: https://help.codeium.com/hc/en-us/categories/17058258214939-User-Guides Raportointi ja analytiikka: tarjoaa mahdollisuuden käytön seurantaan tiimeittäin, ajanjaksoittain tietoa ajan säästämisestä jne Automaatio: automaatisoidut koodiehdotukset/koodin täydennys ja virheiden korjaus
Tietoturva	Salausmenetelmät (datan suojaaminen): **tietoja ei julkisesti saatavilla (oletuksena käyttää vahvoja salausmenetelmiä)** Autentikointi ja valtuutus: **autentikointitoken varmistaa käyttäjän henkilöllisyyden ja oikeudet, Enterprise tilauksessa käyttäjä voi itse määritellä omat autentikointi- ja valtuutusprosessit** Tietosuoja (GDPR yhteensopivuus): **on GDPR yhteensopiva** Haavoittuvuuksien hallinta: **säännölliset päivitykset ja korjaustoimenpiteet jos löytyy haavoittuvuuksia** Auditoinnit tai sertifikaatit: **SOC 2 (taso II)** Tietojen säilytys ja poistaminen: **henkiötietoja säilytetään niin kauan kuin tarpeen, kun tietoja ei enää tarvita ne poistetaan. Rekisteröity käyttäjä voi pyytää tietojensa poistoa.**
Hinnoittelu (kustannukset)	Kolme erilaista tilausvaihtoehtoa: Developer ILMAINEN, (vanhempi chat GPT, tuki ainoastaan discordin ja githubin yhteisöjen kautta) Teams: $19 USD/käyttäjä/kuukausi Enterprise: hinta neuvoteltavissa
Yhteenveto (hyödyt/haitat)	+ nopeuttaa koodin kirjoittamista, chat ominaisuus joka mahdollistaa monipuolisemman käytön, SOC 2 (taso II) sertifikaatti, ei kouluta malliaan käyttäjien datalla, yksittäisille käyttäjille ilmainen (suppea) käyttö, saanut käyttäiöltä parhaat arviot - käyttäjän täytyy osata lukea koodia (voi tehdä joskus virheellisiä ehdotuksia), omat taidot "ruostuvat" jos nojaa liikaa tekoälyyn, Codeium suhteellisen uusi toimija (luotettavuus?), ei kattavaa dokumentaatiota saatavilla, ilmaiskäyttäjien rajoitettu tuki (github tai discord yhteisöt)
Demovideo URL	Linkki demovideoon: codeium test.mkv Linkki Codeiumin demoalustalle: https://codeium.com/playground

Muuta tietoa ohjelmistosta:

Saanut käyttäjiltä hyvät arvioinnin eri alustoilla. Ongelmana kutienkin se, ettei avointa ja kattavaa dokumentaatiota eikä julkista tietoa mm. salausmenetelmistä ole saatavilla.

Tekoälypohjaisen työkalun arviointilomake

Ohjelmiston nimi ja versio	**Github Copilot** v1.213.0 (viimeisin päivitys 15.7.2024)
Ohjelmiston valmistaja	Github, OpenAI & Microsoft (käytetty OpenAI Codex algoritmia)
URL	Demosivu: https://github.com/features/copilot
Ohjelmiston käyttötarkoitus	Tekoälyyn perustuva ohjelmointityökalu, jonka tarkoituksena on nopeuttaa ja tehostaa koodin kirjoittamista. Työkalu tarjoaa koodiehdotuksia ja täydentää koodin kirjoittamisen yhteydessä koodirivejä tai -pätkiä automaattisesti. Tarjoaa työkalun mukana myös chat ominaisuuden jossa voi pyytää esimerkiksi valmista koodia, koodiehdotuksia tai koodin korjaamista.
Tuki ja ylläpito (dokumentaatio)	Dokumentaatio: https://docs.github.com/en/copilot Turvallisuus: https://resources.github.com/copilot-trust-center/ Tuki: https://docs.github.com/en/support/contacting-github-support TAI https://docs.github.com/en/support/contacting-github-support/using-copilot-in-github-support
Integroitavuus ja yhteensopivuus (koodieditorit)	Visual Studio, JetBrains IDEs, Neovim, Visual Studio Code
Ominaisuudet (käyttäjäystävällisyys, käyttöohjeet, raportointi/analytiikka, automaatio)	Käyttäjäystävällisyys: Helppokäyttöinen ja selkeä käyttöliittymä, nopea asentaa, tarjoaa demosivulla lyhyitä mallivideoita joiden avulla voi tutustua ominaisuuksiin Käyttöohjeet: https://docs.github.com/en/copilot/using-github-copilot Raportointi/analytiikka: tarjoaa raportointi ja analytiikkatyökaluja joilla voi seurata mm. työkalun hyödyntämistä, vaikutuksia, kuukausikustannuksia ym. Automaatio: täydentää koodia automaattisesti ja tarjoaa ehdotuksia reaaliaikaisesti
Tietoturva	Salausmenetelmät (datan suojaaminen): **Käyttää vahvoja salausmenetelmiä datan suojaamiseen (TLS, FIPS Publication 140-2)** Autentikointi ja valtuutus: **Monipuoliset autentikointimenetelmät ja käyttäjän oikeuksien hallinta kts. Trust Center** Tietosuoja (GDPR yhteensopivuus): **On GDPR-yhteensopiva** Haavoittuvuuksien hallinta (korjaustoimenpiteet): **säännölliset päivitykset ja korjaustoimenpiteet jos löytyy haavoittuvuuksia** Auditoinnit tai sertifikaatit (on/ei ole): **Business ja Enteprise sisältävät ISO 27001 sertifikaatin. Tulossa SOC 2 tyyppi 2 (tällä hetkellä voimassa 1.4-30.9.2024).** Tietojen säilytys ja poistaminen (tietojen säilytysajat, tietojen palauttaminen): **Säilyttää tietoja 30 päivän ajan (käyttäjän toimet ja koodiehdotukset), palauttamiskäytännöt noudattelevat Azuren tietoturvastandardeja**
Hinnoittelu (kustannukset)	Sovelluksella on kolme erilaista tilausvaihtoehtoa riippuen käyttötarpeesta. **Copilot Individual $10 USD ($100 USD/vuosi),** tarkoitettu yksityiskäyttöön. **Copilot Business $19 USD/käyttäjä/kuukausi**, tarkoitettu yrityskäyttöön. **Copilot Enterprise $39USD/käyttäjä/kuukausi**, tarkoitettu yrityskäyttöön siten että sovellus on kustomoitavissa yrityksen tarpeisiin.
Yhteenveto (hyödyt/haitat)	+ nopeuttaa koodin kirjoittamista, chat ominaisuus joka mahdollistaa monipuolisemman käytön, käyttää lähdekoodia ainoastaan ehdotusten luomiseen (ei tallenna eikä jaa sitä muille käyttäjille) - käyttäjän täytyy osata lukea koodia (voi tehdä joskus virheellisiä ehdotuksia), omat taidot "ruostuvat" jos nojaa liikaa tekoälyyn, tietosuoja (kerää käyttäjien sitoutumistietoja, kehotteita jne myös enterprise ja business tilauksissa)
Demovideo URL	Linkki demovideoon: Github copilot test.mkv Linkki Copilotin omaan demoympäristöön (ohjeet): https://github.com/github/copilot-codespaces-demo

Muuta tietoa ohjelmistosta:

Laaja ja avoin dokumentaatio saatavilla. Integroitu Azure SQL tietokantaan (tehostaa tietokantakyselyitä ja tietokantojen hallintaa). Tällä hetkellä SOC 2 taso 2 arviointi menossa, päättyy 30.9.2024 jonka jälkeen SOC 2, taso II sertifikaattikelpoinen.