Antton Koivisto

Etänä toteutettavan pariohjelmoinnin hyödyt ja haitat

Tietotekniikan kandidaatintutkielma

29. huhtikuuta 2023

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Antton Koivisto

Yhteystiedot: antton.v.koivisto@student.jyu.fi

Ohjaaja: Tytti Saksa

Työn nimi: Etänä toteutettavan pariohjelmoinnin hyödyt ja haitat

Title in English: Benefits and disadvantages of remote pair programming

Työ: Kandidaatintutkielma

Opintosuunta: Tietotekniikka

Sivumäärä: 21+0

Tiivistelmä: Tässä kandidaatintutkielmassa selvitetään etänä toteutettavan pariohjelmoinnin hyötyjä sekä haittoja verrattuna samassa tilassa toteutettavaan pariohjelmointiin ja ohjelmointiin yksin.

Avainsanat: Pariohjelmointi, Etänä toteutettava pariohjelmointi

Abstract: This bachelor's thesis examines benefits and disadvanteges of remote pair programming compared to normal pair programming and programming alone.

Keywords: Pair programming, remote pair programming, distributed pair programming

Sisällys

1 Johdanto

Etänä toteutettava pariohjelmointi nousi ajankohtaiseksi keväällä 2020, kun koronapandemian tuomat rajoitukset pakottivat monet yritykset ja oppilaitokset etätyöskentelyyn. Tällöin pariohjelmointia ei voitu toteuttaa normaaliin tapaan, eli samassa tilassa vierekkäin istuen. Etänä toteutettava pariohjelmointi ei kuitenkaan ole vasta pandemian aikana keksitty asia, vaan aihetta on tutkittu jo paljon aiemminkin, ainakin vuodesta 2002 alkaen (Baheti, Gehringer ja Stotts 2002). Kuten kaikkeen muuhunkin etätyöskentelyyn, myös pariohjelmointiin etänä liittyy niin hyötyjä kuin myös haittojakin. Etätyöskentely tuo paljon mahdollisuuksia esimerkiksi paikkariippumattoman työskentelyn muodossa, mutta ongelmiakin on.

Tämän kirjallisuuskatsauksen tarkoituksena on tuoda esiin etänä toteutettavan pariohjelmoinnin haittoja sekä hyötyjä. Tarkoituksena on myös selvittää, voidaanko etänä toteutettavalla pariohjelmoinnilla korvata tavallinen pariohjelmointi esimerkiksi etätyöskentelyyn pakottavan pandemian aikana. Tutkimuksessa keskitytään pariohjelmointiin opiskelujen yhteydessä, ja työmaailman pariohjelmointi jätetään jatkotutkimusaiheeksi. Vertailua tehdään samassa tilassa toteutettavaan pariohjelmointiin sekä yksin toteutettavaan ohjelmointiin. Tutkimuksessa sivutaan myös hieman tulevaisuuden mahdollisuuksia, ja pohditaan miten tekoälyllä voidaan vaikuttaa etänä toteutettavaan pariohjelmointiin.

Tutkielman toisessa luvussa esitellään pariohjelmointia tarkemmin ja käydään läpi sen historiaa sekä hyötyjä ja haittoja. Kolmannessa luvussa esitellään etänä toteutettavan pariohjelmoinnin kahta erilaista toimintatapaa. Neljännessä ja viidennessä luvussa perehdytään varsinaiseen aiheeseen, eli etänä toteutettavan pariohjelmoinnin hyötyihin sekä haittoihin. Kuudes luku on yhteenveto, jossa tiivistetään tutkielman havainnot.

2 Pariohjelmointi

Pariohjelmoinnissa kaksi henkilöä istuvat yhdessä saman tietokoneen ääressä yhteisen ohjelmointitehtävän parissa. Henkilöillä on yleensä roolit, joita vaihdellaan välillä. Ensimmäistä näistä voidaan kutsua kuskiksi (engl. *driver*), joka kirjoittaa koodia. Toinen puolestaan tarkkailee sekä neuvoo vierestä, ja häntä kutsutaan navigaattoriksi (engl. *navigator*) (Williams ym. 2002). Joissakin yhteyksissä tästä neuvovasta osapuolesta on käytetty myös termiä *observer* (Williams ja Upchurch 2001), joka kääntyy suomeksi tarkkailijaksi. Termi *navigator* lienee kuitenkin parempi kuvaus tuosta roolista, sillä myös vieressä istuvan osapuolen tulisi olla aktiivinen, eikä vain katsoa vierestä. (Beck 2000, s. 100). Myös suurin osa tämän tutkielman lähdekirjallisuudesta käyttää navigator-termiä.

Luvussa 2.1 käsitellään pariohjelmoinnin historiaa ja luvussa 2.2 pariohjelmoinnin hyötyjä sekä haittoja. Monet näistä hyödyistä ja haitoista toistuvat myös etänä toteutettavassa pariohjelmoinnissa, minkä vuoksi niitä on hyvä käsitellä tässä tutkielmassa.

2.1 Pariohjelmoinnin historiaa

Pariohjelmoinnin kokeiluja on tiedossa ainakin vuodesta 1978 lähtien, mutta varsinaiseen suosioon se nousi 2000-luvun alkupuolella ketterän kehityksen myötä (Alves De Lima Salge ja Berente 2016). Ketterän kehityksen luoja Kent Beck nosti pariohjelmoinnin ketterän kehityksen yhdeksi merkityksellisimmistä tekijöistä kirjassaan *Exreme Programming Explained: Embrace Change* (Beck 2000).

Pariohjelmointia on vuoden 2000 jälkeen tutkittu lukuisissa tutkimuksissa erilaisista näkökulmista. Monissa näistä on tutkittu opiskelijoiden suoriutumista ohjelmointikursseilla esimerkiksi vertailemalla pariohjelmoinnin ja yksin toteutetun ohjelmoinnin vaikutuksia koepisteisiin. Viimeisimpinä vuosina tutkimusta on tehty myös esimerkiksi pariohjelmoinnista virtuaalitodellisuudessa (Dominic ym. 2020) sekä tekoälyn avulla toimivan virtuaaliparin hyödyntämisestä pariohjelmoinnin helpottamiseksi (Robe ja Kuttal 2022).

Etänä toteutettavaa pariohjelmointia on tutkittu selvästi pienemmällä laajuudella, mutta en-

simmäisiä tutkimuksia aiheesta on jo 2000-luvun alkuvuosista lähtien esimerkiksi Bahetin ym. (2002) toteuttamana. Jo tuossa tutkimuksessa havaittiin, että etänä toteutettava pariohjelmointi on varteenotettava vaihtoehto samassa tilassa toteutettavalle pariohjelmoinnille (Baheti, Gehringer ja Stotts 2002).

2.2 Pariohjelmoinnin hyödyt ja haitat

Pariohjelmoinnin merkittäviä hyötyjä ovat ainakin koodin laadun parantuminen ja virheiden vähentyminen (Williams ja Upchurch 2001). Koodin mahdolliset virheet myös löytyvät nopeammin ja ongelmat ratkeavat helpommin kuin yksin ohjelmoitaessa (Cockburn ja Williams 2001). Pariohjelmoinnin seurauksena molemmilla tekijöillä on ymmärrys koodista, jolloin kaikki tieto ei ole vain yhden henkilön varassa (Cockburn ja Williams 2001). Alves De Lima Salge ja Berente (2016) kävivät läpi erilaisia tutkimuksia pariohjelmoinnista, ja tulivat samaan tulokseen koodin laadun paranemisesta.

Ongelmia pariohjelmointiin saattaa aiheuttaa sopivan parin löytäminen sekä esimerkiksi kommunikointi. Parit eivät myöskään aina ole samalla tasolla aiempien ohjelmointitaitojen suhteen. Tästä voi seurata vapaamatkustamista, eli tilanteita joissa parin heikompitasoinen osapuoli pääsee helpommalla kun osaavampi osapuoli tekee suurimman osan työstä (Yuan ja Cao 2019). Simonin ja Hanksin (2008) tutkimuksessa osa opiskelijoista koki etteivät he ymmärtäneet kirjoittamaansa koodia niin hyvin kuin yksin ohjelmoidessa. Tällainen liittyy varmasti jossain määrin juuri vapaamatkustamiseen. Vapaamatkustaminen ei ole aina heikompitasoisen osapuolen valinta päästä helpolla, vaan osaavampi osapuoli voi joko tietoisesti tai tiedostamatta ottaa dominoivan roolin. Osalla Simonin ja Hanksin (2008) haastattelemista opiskelijoista oli juuri tällaisia kokemuksia pariohjelmoinnista.

Parien eritasoisuus sekä erilaiset osaamisalueet eivät kuitenkaan aina ole pelkästään haitaksi. Heikompitasoinen voi oppia paljon työskennellessä osaavamman kanssa, mutta tämä vaatii osaavammalta mentorointia sekä kärsivällisyyttä (Kangas 2004). Kankaan mukaan pariohjelmointi toimii kuitenkin parhaiten samantasoisten ohjelmoijien muodostamissa pareissa, kunhan kyseessä ei ole kahden aloittelijan muodostama pari.

Kokonaisuudessaan voidaan edellä listattujen asioiden perusteella todeta pariohjelmoinnin

olevan hyödyllinen ja toimiva työskentelytapa. Hyötyjä on selvästi enemmän kuin haittoja, mutta kaikkiin tilanteisiin pariohjelmointi ei kuitenkaan sovellu. Jos parityöskentely ei kahden henkilön välillä toimi, voi olla parempi työskennellä yksin tai yrittää löytää henkilöille sopivammat parit.

3 Etänä toteutettava pariohjelmointi

Etänä toteutettava pariohjelmointi voidaan jakaa karkeasti kahteen erilaiseen toimintatapaan, joista ensimmäisessä hyödynnetään näytönjakoa ja toisessa yhteisen ohjelmoinnin mahdollistavaa kehitysympäristöä (Hanks 2002). Tässä luvussa käsitellään lyhyesti näitä kahta erilaista toimintatapaa.

3.1 Pariohjelmointi näytönjakoa hyödyntämällä

Pariohjelmointi näytönjakoa hyödyntämällä muistuttaa hyvin paljon tavallista pariohjelmointia. Näytön näkymä jaetaan toiselle osapuolelle jonkin näytönjaon mahdollistavan sovelluksen avulla, ja näin parin molemmat jäsenet pääsevät näkemään koodin. Tällaisia näytönjaon mahdollistavia ohjelmia ovat esimerkiksi Zoom ja Microsoft Teams (Adeliyi ym. 2021). Näytönjakoa hyödyntävät sovellukset eivät yleensä ole pariohjelmointiin suunniteltuja sovelluksia, vaan ne ovat yleisesti käytössä etätyöskentelyssä. Näytönjaon lisäksi näitä ohjelmia käytettäessä parin jäsenet voivat keskustella sekä nähdä toistensa kasvot, jos tietokoneessa on mikrofoni ja kamera mahdollistamassa nämä. Englanniksi tätä toimintatapaa voidaan kutsua nimellä *remote pair programming* ja lyhentää muotoon *RPP* (Schenk, Prechelt ja Salinger 2013).

3.2 Pariohjelmointi jaetussa kehitysympäristössä

Toinen tapa etänä toteutettavaan pariohjelmointiin on jaetun kehitysympäristön käyttäminen. Schenkin ym. (2013) mukaan tästä toimintatavasta käytetään englanniksi nimitystä *distributed pair programming*, joka lyhennetään muotoon *DPP*. Jaetun kehitysympäristön käyttäminen ei kuitenkaan täysin vastaa pariohjelmoinnin alkuperäisen määritelmän mukaista tekemistä, koska molemmilla voi olla mahdollisuus muokata koodia. Tällöin perinteiset pariohjelmoinnin roolit eivät välttämättä toteudu (Schenk, Prechelt ja Salinger 2013). Joissain jaetuissa kehitysympäristöissä on mahdollisuus lukita muokkausoikeus kerralla vain toiselle käyttäjälle, jolloin rooleja voidaan toteuttaa normaaliin tapaan.

Yksi nykyään käytössä oleva jaettu kehitysympäristö on yleisesti käytetyn tekstieditori Visual Studio Coden laajennus Live Share (Microsoft 2021). Live Sharen avulla parin molemmat jäsenet voivat tarkastella ja muokata samaa ohjelmakoodia omilla laitteillaan. Myös suosittuun kehitysympäristöön Eclipseen on kehitetty lukuisia pariohjelmoinnin mahdollistavia lisäosia, kuten Ripple, Sangam ja Saros (da Silva Estácio ja Prikladnicki 2015). Merkittävä ero näytönjaolla toteutettuun pariohjelmointiin on se, että nämä sovellukset ja niiden lisäosat ovat varta vasten pariohjelmointia varten suunniteluja.

4 Hyödyt ja mahdollisuudet

Etänä toteutettava pariohjelmointi parantaa samassa tilassa toteutettavan pariohjelmoinnin tavoin suoriutumista ohjelmoitaessa verrattuna yksin tehtävään ohjelmointiin (Beasley ja Johnson 2022). Beasleyn ja Johnsonin tutkimuksessa etänä toteutettavaan pariohjelmointiin osallistuneet opiskelijat suoriutuivat kurssitehtävistä paremmin kuin yksin ohjelmoineet. Hanksin (2005) tutkimuksessa etänä toteutettavaa pariohjelmointia hyödyntäneet opiskelijat pärjäsivät yhtä hyvin kuin samassa tilassa pariohjelmoineet. Pariohjelmoinnin hyödyt siis toistuvat myös etänä pariohjelmoitaessa.

Xinogalosin ym. (2017) tutkimuksessa opiskelijat kokivat ohjelmoinnin oppimisen mukavampana ja koodin ongelmakohtien korjaamisen sujuvampana verrattuna yksin tehtävään ohjelmointiin. Ongelmien tehokkaan selvittämisen lisäksi myös luottamus omien ratkaisujen oikeellisuuteen voi parantua yhdessä ohjelmoitaessa (Xinogalos ym. 2017). Hanksin (2005) tutkimuksessa ei havaittu eroja etänä pariohjelmoineiden ja samassa tilassa pariohjelmoineiden parien välillä tuon saman luottamuksen osalta. Itseluottamuksen lisääntyminen omiin ratkaisuihin on ymmärrettävää, sillä pariohjelmoidessa omalta parilta saa jatkuvasti palautetta ja hyväksyntää ratkaisuihin. Pariohjelmointi voi lisätä rohkeutta kysymysten kysymiseen luennoilla, ja Beasley ja Johnson (2022) huomasivat tämän vaikutuksen erityisesti naisissa. Pariohjelmoidessa kommunikointi oman parin kanssa on jatkuvaa, joten on luonnollista että rohkeus lisääntyy. Yhteistyö myös madaltaa kynnystä tehdä koodiin muutoksia, verrattuna ohjelmoimiseen yksin (Rajpal 2018). Ohjelman rikkoutumista pelätään siis vähemmän kun sitä tehdään yhdessä.

Etänä toteutettu pariohjelmointi mahdollistaa muun etätyöskentelyn tavoin paikkariippumattoman työskentelyn. Työskentely etänä voi esimerkiksi vähentää matkustelua, mutta myös saada aikaan tuntemuksia yksinäisyydestä. Verrattuna itsenäiseen työskentelyyn muista eristyksissä, etänä toteutettavalla pariohjelmoinnilla voidaan vähentää eristymisen kokemusta (Rajpal 2018). Adeliyin ym. (2021) tutkimuksessa etänä pariohjelmoineet tutkittavat kokivat sosiaalisen vuorovaikutuksen paranemista ja yhteenkuuluvuuden tunnetta. Pariohjelmoidessa pääsee jakamaan omia ajatuksia ja tuntemuksia ohjelmointiin liittyen, mikä varmasti lisää yhteenkuuluvuuden tunnetta. Pariohjelmointi parantaa opiskelijoiden kommunikointitaitoja,

ja näistä parantuneista taidoista sekä muiden kanssa työskentelyn harjoittelemisesta on hyötyä työelämässä (Beasley ja Johnson 2022). Pariohjelmointi valmistaa opiskelijoita hyvin työelämää varten, eikä tämä koske pelkästään tavallista pariohjelmointia, vaan myös etänä toteutettavaa pariohjelmointia.

Etänä toteutettava pariohjelmointi on opiskelijoiden keskuudessa pidetty toimintatapa. Xinogalosin ym. (2017) tutkimuksessa yli 80 prosenttia opiskelijoista kertoi kokemuksen etänä toteutettavasta pariohjelmoinnista olleen positiivinen. Adeliyin ym. (2021) haastattelemista opiskelijoista noin 90 prosenttia suosittelisi etänä toteutettavaa pariohjelmointia muille opiskelijoille ja laajempaan käyttöön ylipäätään. Galdon ym. (2022) tutkimuksen tutkittavista opiskelijoista noin 70 prosenttia piti etänä toteutettavasta pariohjelmoinnista, eikä kenenkään suhtautuminen siihen ollut täysin negatiivista.

Adeliyin ym. (2021) tutkimuksessa opiskelijat kokivat pariohjelmoinnin tuovan positiivista painetta saada asiat hoidettua ajoissa ja pysyä mukana opiskelutahdissa. Samantyyppisiä havaintoja on myös työelämästä: Pariohjelmointi kotoa käsin voi helpottaa keskittymistä ja saada asennoitumaan työntekoon paremmin, verrattuna ohjelmointiin yksin (Smite ym. 2021). Pariohjelmointihetki voi auttaa rytmittämään päivää ja tekemään työskentelystä tehokkaampaa.

Etänä toteutettavan pariohjelmoinnin tutkimuksia on viime vuosina laajennettu mielenkiintoisiin suuntiin. Robe ja Kuttal (2022) veivät tutkimuksessaan etänä toteutettavan pariohjelmoinnin suuntaan jota ei ehkä voi enää kutsua pariohjelmoinniksi. Tutkimuksessa selvitettiin tekoälybotti PairBuddy:n toimivuutta pariohjelmoinnin toisena osapuolena. Tulokset olivat hyviä, sillä suurin osa tutkituista piti PairBuddy:n kanssa työskentelystä. PairBuddy myös osasi vastata kysymyksiin lähes 80 prosentin varmuudella.

Tekoäly pariohjelmoinnin toisena osapuolena voi mahdollistaa tehokkaamman ohjelmoinnin, kun kahden ihmisen työajan sijaan käytetään vain yhden ihmisen aikaa. Pariohjelmointi nimittäin voi viedä kokonaisuudessaan enemmän aikaa kuin ohjelmointi yksin (Cockburn ja Williams 2001). Todennäköisesti ihmisten välisellä pariohjelmoinnilla on tulevaisuudessakin paikkansa, sillä Roben ja Kuttalin (2022) tutkimuksessa ihminen oli pidetympi keskustelukumppani kuin tekoäly. Tutkimuksessa havaittiin tutkittavien suhtautuneen aluksi epäileväs-

ti PairBuddy:n toimivuuteen, mikä näkyi tekoälybotille esitettyjen kysymysten pienempänä määränä alkuvaiheessa ja kasvuna tutkimuksen loppua kohden.

Tekoälystä voisi olla paljon hyötyä ohjelmoinnin opettamisessa ainakin tulevaisuudessa. Luvussa 2.2 todettiin, että kokeneemman seurassa ohjelmoitaessa oppiminen voi olla tehokasta. Kokeneemman ohjelmoijan seuraa ei kuitenkaan aina ole saatavilla ja opettavan osapuolen pitäisi myös olla riittävän kärsivällinen ja kannustava. Tekoälyllä tällaisia ongelmia ei samaan tapaan tulisi, joten erilaisten tekoälybottien hyödyntäminen voisi soveltua ainakin tulevaisuudessa erittäin hyvin ohjelmoinnin opetukseen.

Etänä toteutettavaa pariohjelmointia on kokeiltu myös virtuaalitodellisuudessa, vr-laseja hyödyntämällä. Tulokset olivat lupaavia, sillä koodin ongelmakohdat onnistuttiin löytämään ja korjaamaan nopeammin kuin pariohjelmoidessa näytönjakoa hyödyntäen (Dominic ym. 2020). Tekoälyn hyödyntämisen ohella tämä voisi olla toinen tulevaisuuden mahdollinen kehityssuunta etänä toteutettavan pariohjelmoinnin kehittämiseen.

5 Haitat ja ongelmakohdat

Edellisessä luvussa todettiin etänä toteutettavan pariohjelmoinnin olevan toimiva menetelmä, ja huomattiin sen eroavan tavallisesta pariohjelmoinnista hyvin vähän. Pariohjelmoinnin etäversioon liittyy kuitenkin myös joitakin haasteita. Tehtävien tekeminen voi viedä enemmän aikaa verrattuna samassa tilassa toteutettavaan pariohjelmointiin (Satratzemi, Stelios ja Tsompanoudi 2022), ja etätyöskentelyyn tarkoitettujen työkalujen käyttö voidaan kokea vaikeaksi (Smite ym. 2021). Kognitiivinen kuorma on korkeampi verrattuna samassa tilassa toteutettavaan pariohjelmointiin, muttei kuitenkaan suurempi kuin yksin ohjelmoitaessa (Tsai, Yang ja Chang 2015). Tsain ym. mukaan korkeampi kognitiivinen kuorma johtuu pääasiassa teknisistä ongelmista.

Xinogalosin ym. (2017) tutkimuksessa tekniset ongelmat olivat etänä toteutettavaa pariohjelmointia eniten vaikeuttanut tekijä. Galdon ym. (2022) tutkimuksessa yli puolet osallistujista raportoi teknisistä ongelmista tutkimusjakson aikana. Mielenkiintoinen havainto tuosta tutkimuksesta oli se, että toisin kuin aikuiset, tutkitut noin 12-vuotiaat eivät jääneet jumiin ongelmatilanteisiin, vaan ratkaisivat ongelmat esimerkiksi pyytämällä pariaan toistamaan asiansa ääniyhteyden katketessa. Teknisiä ongelmia esiintyy kuitenkin myös pariohjelmoidessa samassa tilassa sekä yksin ohjelmoidessa, joten täysin eroon niistä ei pääse mitenkään.

Etänä toteutettavaa pariohjelmointia varten on vuosien varrella kehitetty monia jaettuja kehitysympäristöjä. Monet näistä ovat jotakin yksittäistä tutkimusta varten tehtyjä ja Adeliyi ym. (2021) kritisoivat näitä prototyypeiksi jääneitä sekä huonosti toimivia kehitysympäristöjä. Yksi esimerkki tällaisesta on Ripple (Boyer ym. 2008), josta ei enää nykyään löydy tutkimusartikkelin lisäksi muuta tietoa. Myöskään tutkimusartikkelissa mainittu verkko-osoite ei ole enää käytössä. Monissa jaetuissa kehitysympäristöissä ei myöskään ole huomioitu opiskelijoiden tarpeita, tai ne eivät ole aloittelijoille sopivia (Ying ja Boyer 2020).

Heikosti toimivien jaettujen kehitysympäristöjen vuoksi Adeliyi ym. (2021) päättivät hyödyntää yleisesti käytössä olevia viestintäsovelluksia tutkimuksessaan etänä toteutettavasta pariohjelmoinnista. Näistä käyttöön valikoitui lopulta jo aiemmin mainittu Microsoft Teams, jota käytetään moneen muuhunkin käyttötarkoitukseen etänä toteutettavan pariohjelmoinnin

lisäksi. Ongelmia liittyy kuitenkin myös pariohjelmointiin näytönjakoa hyödyntämällä. Erityisesti huono verkkoyhteys voi aiheuttaa ongelmia (Schenk, Prechelt ja Salinger 2013). Toinen haaste liittyy roolien vaihtamiseen, sillä kirjoitettava ohjelmakoodi pitää olla saatavilla parin molempien jäsenten laitteilla, jotta roolien vaihdon tapahduttua voidaan jatkaa työskentelyä siitä, mihin jäätiin. Hyvä ratkaisu tähän ongelmaan on tietokoneen etäkäyttö, minkä osa viestintäsovelluksista mahdollistaa (Adeliyi ym. 2021).

Etänä pariohjelmoidessa kommunikointi on erilaista verrattuna tavalliseen pariohjelmointiin, sillä parin toisen osapuolen eleitä ja ilmeitä ei välttämättä pääse näkemään. Tällainen sanaton viestintä antaa paljon tärkeää tietoa molemmille osapuolille tavallisessa pariohjelmoinnissa (Schenk, Prechelt ja Salinger 2013). Näitä eleitä voidaan päästä näkemään jakamalla videokuvaa. Schenkin ym. (2013) tutkimuksessa tutkittavat eivät kuitenkaan hyödyntäneet videokuvan jakamista, sillä se ei hyödyttänyt merkittävästi ja joissain tilanteissa häiritsi keskittymistä. Videon jakaminen ei myöskään ole kannattavaa, jos käytettävän tietokoneen näyttö on pieni, jolloin näytölle ei välttämättä saa mahtumaan samanaikaisesti sekä videokuvaa että kirjoitettavaa koodia hyvin. Myös heikko verkkoyhteys voi aiheuttaa ongelmia videokuvan jakamiseen.

Videokuvan sijaan tärkeimmäksi kommunikaation keinoksi etänä toteutettavassa pariohjelmoinnissa koetaan ääni (Satratzemi, Stelios ja Tsompanoudi 2022). Ääneenkin voi liittyä ongelmia, kuten huonolaatuisuus tai heikko äänenvoimakkuus. Kaikki hiljaiset huokaukset ja muut ääntelyt eivät välttämättä kuulu toiselle osapuolelle yhtä hyvin kuin vierekkäin istuessa (Schenk, Prechelt ja Salinger 2013). Pienetkin ääntelyt voivat kertoa paljon toisen osapuolen tuntemuksista, ja jos videokuvakin on pois käytöstä, voi kommunikaatiosta jäädä puuttumaan merkittävä osa. Keskittyminenkin voi herpaantua helpommin kun äänen kuulee kaiuttimista, eikä vieressä istuvalta parilta (Ho ym. 2004).

Verrattuna tavalliseen pariohjelmointiin, etänä toteutettavassa pariohjelmoinnissa ei ole niin helppoa toteuttaa pariohjelmointia, jos sille tulee hetkellinen tarve (Smite ym. 2021). Tällaisia spontaaneja tilanteita voi tulla työpaikoilla, kun oman koodin kanssa tarvitsee apua. Tällöin on helppoa mennä esimerkiksi viereiseen huoneeseen kysymään apua, mutta puolestaan etänä työskennellessä asiasta pitää sopia ja valmistella etäyhteys pariohjelmoinnin mahdollistamiseksi. Ylipäätään pariohjelmointi etänä voidaan kokea vaikeaksi. Smiten ym.

(2021) tutkimuksessa monet tutkitut eivät hyödyntäneet pariohjelmointia ollenkaan koronapandemian aikana etätyöskennellessä, vaikka olivat pariohjelmoineet aiemmin paikan päällä työskennellessä.

Xinogalos ym. (2017) havaitsivat ongelman sopivan ajan löytämisessä pariohjelmoinnin toteuttamiseksi, mikä on luonnollista, kun yritetään saada kahden ihmisen aikatauluja sopimaan yhteen. Sama ongelma liittyy todennäköisesti myös tavalliseen pariohjelmointiin, ja etänä toteutettava pariohjelmointi voi jopa helpottaa tätä ongelmaa, kun parin jäsenten ei tarvitse olla samassa paikassa pariohjelmoinnin mahdollistamiseksi.

6 Yhteenveto

Tässä kirjallisuuskatsauksessa perehdyttiin etänä toteutettavaan pariohjelmointiin käymällä läpi sen hyötyjä ja haittoja. Etänä toteutettavaan pariohjelmointiin liittyy paljon mahdollisuuksia, mutta myös joitakin haasteita, kuten tekniset ongelmat, korkeampi kognitiivinen kuorma sekä sopivan parin löytämisen haasteet. Kokonaisuudessaan etänä toteutettavan pariohjelmoinnin ongelmat vaikuttavat olevan yhdistelmä pariohjelmoinnin ja etätyöskentelyn ongelmia. Sama pätee myös hyötyihin, joita ovat ohjelmakoodin laadun paraneminen, ongelmien sujuvampi korjaaminen, lisääntynyt itseluottamus omiin ratkaisuihin, yksinäisyyden ja eristymisen tunteen väheneminen sekä mahdollisuus työskennellä paikkariippumattomasti. Etänä toteutettava pariohjelmointi on myös erittäin pidetty toimintatapa, ja monissa tilanteissa parempi vaihtoehto kuin kokonaan yksin tekeminen.

Opiskelijoiden omat mieltymykset tulee kuitenkin huomioida, sillä pariohjelmointi ei välttämättä sovi kaikille tai kaikki eivät halua tehdä sitä. Pariohjelmointi valmistaa hyvin työelämää varten, joten opiskelijoiden olisi hyvä päästä kokeilemaan pariohjelmointia tai pariohjelmointia etänä edes vähän. Pariohjelmointi, erityisesti etänä toteutettuna, tulee järjestää huolellisesti, jotta kokemus ei olisi negatiivinen. Opastuksen tulee olla huolellista, työkalujen toimintavarmoja sekä toimiva ja myös parien muodostamisessa tulee olla tarkkana.

Johdannossa pohdittiin, voidaanko etänä toteutettavalla pariohjelmoinnilla korvata samassa tilassa toteutettava pariohjelmointi. Tutkimuksen havaintojen perusteella se onnistuu, ja pariohjelmoinnin hyödyt saavutetaan hyvin myös etänä toimiessa. Pariohjelmointia ei ainakaan kannata lopettaa, vaikka jouduttaisiinkin siirtymään etätyöskentelyyn samassa tilassa toimimisen sijasta, esimerkiksi pandemian tähän pakottaessa.

Luvuissa 3.1 ja 3.2 esiteltiin etänä toteutettavan pariohjelmoinnin kaksi erilaista tapaa, näytönjako ja jaetut kehitysympäristöt. Molemmissa on omat hyvät ja huonot puolensa, eikä kumpaakaan näistä voi nostaa selvästi paremmaksi vaihtoehdoksi. Jaetuista kehitysympäristöistä voisi nostaa esiin kysymyksen, onko etänä toteutettavan pariohjelmoinnin tarpeen olla puhdasta pariohjelmointia rooleineen ja niiden vaihtoineen. Voisiko jaetussa kehitysympäristöissä toimiessa muokata koodia samanaikaisesti ja pyrkiä eroon toimintatavasta, joka on

alun perin kehitetty erilaista työskentelyasetelmaa varten? Tätä asiaa olisi hyvä tutkia tulevaisuudessa.

Monet tutkimukset ovat vertailleet etänä toteutettavaa pariohjelmointia ohjelmointiin yksin. Tulevaisuudessa olisi hyvä tehdä vertailua vielä enemmän pariohjelmoinnin ja etänä toteutettavan pariohjelmoinnin välillä, sekä selvittää etänä toteutettavan pariohjelmoinnin eri variaatioiden eroa. Mielenkiintoisia mahdollisuuksia tarjoavat virtuaalitodellisuuden sekä tekoälyn hyödyntäminen. Näihinkin olisi hyvä keskittyä tulevaisuudessa, jotta etänä toteutettavasta pariohjelmoinnista voidaan saada vielä toimivampaa ja tehokkaampaa.

Lähteet

Adeliyi, Adeola, Michel Wermelinger, Karen Kear ja Jon Rosewell. 2021. “Investigating Remote Pair Programming In Part-Time Distance Education”. Teoksessa *Proceedings of the 2021 Conference on United Kingdom & Ireland Computing Education Research.* UKICER ’21. Glasgow, United Kingdom: Association for Computing Machinery. ISBN: 9781450385688. https://doi.org/10.1145/3481282.3481290.

Alves De Lima Salge, Carolina, ja Nicholas Berente. 2016. “Pair Programming vs. Solo Programming: What Do We Know After 15 Years of Research?” Teoksessa *2016 49th Hawaii International Conference on System Sciences (HICSS),* 5398–5406. https://doi.org/10.1109/HICSS.2016.667.

Baheti, Prashant, Edward Gehringer ja David Stotts. 2002. “Exploring the Efficacy of Distributed Pair Programming”. Teoksessa *Extreme Programming and Agile Methods — XP/Agile Universe 2002,* 208–220. Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45672-4_20.

Beasley, Zachariah J., ja Ayesha R. Johnson. 2022. “The Impact of Remote Pair Programming in an Upper-Level CS Course”. Teoksessa *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1,* 235–240. ITiCSE ’22. Dublin, Ireland: Association for Computing Machinery. ISBN: 9781450392013. https://doi.org/10.1145/3502718.3524772.

Beck, Kent. 2000. *Extreme programming explained: embrace change.* Addison-Wesley Professional, Reading, MA, Yhdysvallat.

Boyer, Kristy Elizabeth, August A. Dwight, R. Taylor Fondren, Mladen A. Vouk ja James C. Lester. 2008. “A Development Environment for Distributed Synchronous Collaborative Programming”. Teoksessa *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education,* 158–162. ITiCSE ’08. Madrid, Spain: Association for Computing Machinery. ISBN: 9781605580784. https://doi.org/10.1145/1384271.1384315.

Cockburn, Alistair, ja Laurie Williams. 2001. “The Costs and Benefits of Pair Programming”. Teoksessa *Extreme Programming Examined,* 223–243. USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0201710404.

da Silva Estácio, Bernardo José, ja Rafael Prikladnicki. 2015. “Distributed Pair Programming: A Systematic Literature Review”. *Information and Software Technology* 63:1–10. ISSN: 0950-5849. https://doi.org/10.1016/j.infsof.2015.02.011.

Dominic, James, Brock Tubre, Charles Ritter, Jada Houser, Colton Smith ja Paige Rodeghero. 2020. “Remote Pair Programming in Virtual Reality”. Teoksessa *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME),* 406–417. https://doi.org/10.1109/ICSME46990.2020.00046.

Galdo, Aisha Chung, Mehmet Celepkolu, Nicholas Lytle ja Kristy Elizabeth Boyer. 2022. “Pair Programming in a Pandemic: Understanding Middle School Students’ Remote Collaboration Experiences”. Teoksessa *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1,* 335–341. SIGCSE 2022. Providence, RI, USA: Association for Computing Machinery. ISBN: 9781450390705. https://doi.org/10.1145/3478431.3499324.

Hanks, Brian. 2002. “Tool support for distributed pair programming”. Teoksessa *Workshop on Distributed Pair Programming. Extreme Programming and Agile Methods-XP/Agile Universe.* Citeseer.

———. 2005. “Student Performance in CS1 with Distributed Pair Programming”. Teoksessa *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education,* 316–320. ITiCSE ’05. Caparica, Portugal: Association for Computing Machinery. ISBN: 1595930248. https://doi.org/10.1145/1067445.1067532.

Ho, Chih-Wei, Somik Raha, Edward Gehringer ja Laurie Williams. 2004. “Sangam: A Distributed Pair Programming Plug-in for Eclipse”. Teoksessa *Proceedings of the 2004 OOPSLA Workshop on Eclipse Technology EXchange,* 73–77. eclipse ’04. Vancouver, British Columbia, Canada: Association for Computing Machinery. ISBN: 9781450377980. https://doi.org/10.1145/1066129.1066144.

Kangas, Maija. 2004. “The impact of individual differences on pair programming”. Teoksessa *Seminar in software engineering, Spring*. http://www.soberit.hut.fi/T-76.5650/Spring_2004/Papers/M.Kangas_76650_final.pdf.

Microsoft. 2021. *Use Microsoft Live share to collaborate with Visual Studio Code.* Viitattu 6.2.2023. Microsoft. https://code.visualstudio.com/learn/collaboration/live-share.

Rajpal, Mark. 2018. “Effective Distributed Pair Programming”. Teoksessa *Proceedings of the 13th International Conference on Global Software Engineering,* 6–10. ICGSE ’18. Gothenburg, Sweden: Association for Computing Machinery. ISBN: 9781450357173. https://doi.org/10.1145/3196369.3196388.

Robe, Peter, ja Sandeep Kaur Kuttal. 2022. “Designing PairBuddy—A Conversational Agent for Pair Programming”. *ACM Trans. Comput.-Hum. Interact.* (New York, NY, USA) 29, numero 4 (toukokuu). ISSN: 1073-0516. https://doi.org/10.1145/3498326.

Satratzemi, Maya, Xinogalos Stelios ja Despina Tsompanoudi. 2022. “Distributed Pair Programming in Higher Education: A Systematic Literature Review”. *Journal of Educational Computing Research,* https://doi.org/10.1177/07356331221122884.

Schenk, Julia, Lutz Prechelt ja Stephan Salinger. 2013. “Distributed-Pair Programming can work well and is not just Distributed Pair-Programming”. *CoRR* abs/1311.6249. http://arxiv.org/abs/1311.6249.

Simon, Beth, ja Brian Hanks. 2008. “First-Year Students’ Impressions of Pair Programming in CS1”. *J. Educ. Resour. Comput.* (New York, NY, USA) 7, numero 4 (tammikuu). ISSN: 1531-4278. https://doi.org/10.1145/1316450.1316455.

Smite, Darja, Marius Mikalsen, Nils Brede Moe, Viktoria Stray ja Eriks Klotins. 2021. “From collaboration to solitude and back: remote pair programming during Covid-19”. Teoksessa *Agile Processes in Software Engineering and Extreme Programming: 22nd International Conference on Agile Software Development, XP 2021, Virtual Event, June 14–18, 2021, Proceedings,* 3–18. Springer International Publishing Cham.

Tsai, Chia-Yin, Ya-Fei Yang ja Chih-Kai Chang. 2015. “Cognitive Load Comparison of Traditional and Distributed Pair Programming on Visual Programming Language”. Teoksessa *2015 International Conference of Educational Innovation through Technology (EITT),* 143–146. https://doi.org/10.1109/EITT.2015.37.

Williams, Laurie, ja Richard L. Upchurch. 2001. “In support of student pair-programming”. Teoksessa *Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education.* ACM, helmikuu. https://doi.org/10.1145/364447.364614.

Williams, Laurie, Eric Wiebe, Kai Yang, Miriam Ferzli ja Carol Miller. 2002. “In Support of Pair Programming in the Introductory Computer Science Course”. *Computer Science Education* 12 (3): 197–212. https://doi.org/10.1076/csed.12.3.197.8618.

Xinogalos, Stelios, Maya Satratzemi, Alexander Chatzigeorgiou ja Despina Tsompanoudi. 2017. “Student perceptions on the benefits and shortcomings of distributed pair programming assignments”. Teoksessa *2017 IEEE Global Engineering Education Conference (EDUCON),* 1513–1521. https://doi.org/10.1109/EDUCON.2017.7943050.

Ying, Kimberly Michelle, ja Kristy Elizabeth Boyer. 2020. “Understanding Students’ Needs for Better Collaborative Coding Tools”. Teoksessa *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems,* 1–8. CHI EA ’20. Honolulu, HI, USA: Association for Computing Machinery. ISBN: 9781450368193. https://doi.org/10.1145/3334480.3383068.

Yuan, Hans, ja Yingjun Cao. 2019. “Hybrid Pair Programming - A Promising Alternative to Standard Pair Programming”. Teoksessa *Proceedings of the 50th ACM Technical Symposium on Computer Science Education,* 1046–1052. SIGCSE ’19. Minneapolis, MN, USA: Association for Computing Machinery. ISBN: 9781450358903. https://doi.org/10.1145/3287324.3287352.