Tietojenkäsittelytiede **29** Lokakuu 2009 sivut 37–48
Toimittaja: Jorma Tarhio

Ohjelmoinnin opetus pariohjelmoinnin näkökulmasta

Juha Partanen & Heli Karjalainen
Oulun yliopisto
Tietojenkäsittelytieteiden laitos, Kajaanin yksikkö
juha.partanen@gmail.com, heli.karjalainen@oulu.fi

Tiivistelmä

Selvitämme kirjallisuuteen perustuen, minkälaisen ratkaisun pariohjelmointi tarjoaa ohjelmoinnin oppimisen ongelmiin. Annamme yleiskuvan pariohjelmoinnin eduista ja haitoista opetuksen kontekstissa. Yhtenä keskeisenä havaintona esille on noussut se, että pariohjelmoinnilla saavutetaan tiiviimpää ja virheettömämpää ohjelmakoodia nopeammin kuin yksin ohjelmoitaessa. Lisäksi ohjelmointiparit oppivat toisiltaan ja kysymykset ohjelmoinnin opettajalle ovat selkeämpiä ja jäsentyneempiä kuin ohjelmoitaessa yksin.

1 Johdanto

Pariohjelmoinnissa ohjelmakoodia kirjoitetaan kahden ohjelmoijan toimesta, pareittain. Tämän menetelmän oletetaan tuottavan virheettömämpää ja parempaa koodia kuin, että ohjelmoijat työskentelisivät yksin [14, s. 88–91]. Pariohjelmointi on kahden ohjelmoijan välistä jatkuvaa vuorovaikutusta, jossa ohjelmakoodia tuotetaan, analysoidaan, suunnitellaan ja testataan [5, s. 42]. Prestonin [24, s. 84–85] mukaan pariohjelmointi ja sen opettaminen (pariohjelmointipedagogiikka, pair programming pedagogy) perustuu yhteistoiminnalliseen oppimiseen (collaborative learning). Pariohjelmointipedagogiikka tutkii ohjelmoijaparien muodostumista persoonallisten ominaisuuksien perusteella sekä vertailee parien taitotasoja ja ponnistuksia, persoonallisuustyyppejä ja itsearvioinnin tasoa.

Yhteistoiminnallisessa oppimisessa paras ryhmän koko vaihtelee kahdesta kuuteen oppijaan. Pariohjelmoinnissa tämä ryhmän koko on kaksi. Preston [24, s. 90–91] sanoo menestyksellisen yhteistoiminnallisen oppimisen perustuvan viiteen ominaisuuteen. Prosessia ohjataan pienin askelin kohti yhteistä päämäärää. Pienryhmät tuovat oppimiseen joustavuutta eri kokemustason omaavien jäsenten välillä. Opiskelijoiden yhteistyössä he kysyvät ja oppivat toisiltaan. Keskinäinen riippuvuus tarkoittaa sitä, että parit tai ryhmät jakavat samat resurssit, pariohjelmoinnissa siis yhteisen tietokoneen. Rooleina pariohjelmointitilanteessa ovat ohjelmakoodin kirjoittaja (driver) ja tarkkailija (navigator/copilot). Mahdollista on vielä lisätä kolmas rooli, jossa toimitaan reaaliaikaisena laadunvarmistajana. Vastuullisuus ja

yhteisvastuu saadaan onnistumaan painottamalla henkilökohtaisten kokeiden ja testien arvosanoja enemmän kuin ryhmän tuotosta.

Pariohjelmoinnilla ohjelmoinnin oppimisen menetelmänä on sekä etuja että haasteita. Näiden etujen ja haasteitten välillä ohjelmoinnin opettaja joutuu tasapainoilemaan.

Artikkeli koostuu seitsemästä luvusta. Johdannon jälkeen esittelemme pariohjelmoinnin avulla ohjelmoinnin opetuksesta tehtyjä tutkimuksia. Kolmannessa luvussa käsittelemme oppimista ja ohjelmoinnin opetusta. Neljäs luku käsittelee ohjelmoinnin oppimisen haasteita. Viidennessä luvussa esittelemme ketterät menetelmät ja kuudennessa luvussa pariohjelmoinnin. Seitsemäs luku on pohdinta, ja kahdeksannessa luvussa esittelemme artikkelimme johtopäätökset.

2 Tutkimuksia pariohjelmoinnista

Pariohjelmointia ja sen opetusta on tutkittu yliopisto- ja ammattikorkeakoulutasoisten ohjelmointikurssien osalta. Pariohjelmoijien ja yksin työskennelleiden saamia arvosanoja ja kokemuksia sekä tuotettujen ohjelmistokoodien laatua ja määrää sekä ohjelmointiin käytettyä aikaa ja työmäärää on vertailtu näissä tutkimuksissa.

McDowell ym. [22] jakoivat opiskelijat kahteen ryhmään. Toisessa ryhmässä opiskelijat toteuttivat yksinohjelmointia ja toisessa pariohjelmointia. Pariohjelmointiryhmässä opiskelijat saivat ehdottaa kolmea mahdollista pariehdokastaan ja näiden ehdotusten perusteella tutkijat muodostivat lopulliset parit. Myös Williamsin ym. tutkimuksessa [34] opiskelijat saivat itse valita työparinsa, joiden kanssa he työskentelivät kurssin loppuun saakka.

DeCluen [8] ja Mendesin ym. [23] tutkimuksissa opiskelijat vaihtoivat pareja muutaman viikon välein. Katiran ym. tutkimuksessa [16] vaihdettiin myös pareja. Näissä tutkimuksissa opiskelijat suorittivat uuden projektin tai harjoitustyön aina eri parin kanssa. Tutkimuksissa mitattiin parien yhteensopivuutta.

Tuoreimpia tutkimuksia pariohjelmoinnin käytöstä opetuksessa on Kotavuopion tutkimus [20], jossa tutkittiin Oulun ja Kajaanin ammattikorkeakoulujen ohjelmoinnin kursseilla kokeiltua pariohjelmointia. Tässä tutkimuksessa käytettiin avoimia kysymyksiä, koska haluttiin selvittää opiskelijoiden henkilökohtaisia tuntemuksia pariohjelmoinnista.

Dybå ym. [9] tarkastelivat 15 tutkimusta, jotka vertailevat pariohjelmoinnin ja yksinohjelmoinnin tehokkuutta. Näistä tutkimuksista neljässä tutkimus kohdistui ammattilaisohjelmoijiin ja yhdessätoista ohjelmoinnin opiskelijoihin Euroopassa tai Pohjois-Amerikassa. Dybån ym. [9] tutkimuksessa keskityttiin vertailemaan ohjelmointitavan vaikutusta projektin laatuun, kestoon ja työmäärään.

3 Yhteistoiminnallinen oppiminen ja pariohjelmointi

Historiallisesti ihminen on tarvinnut tietoa ympäristöstään ja omasta suhteestaan siihen. Luontevasti tämä tapahtuu uteliaisuuden kautta. Tällöin oppimista tapahtuu syy- ja seuraussuhteita muodostamalla. Osa oppimisprosessia on myös vastaanotetun informaation kerääminen, tallennus ja jäsentely kokonaisuudeksi. [25, s. 50]

Aikuisen ihmisen käsitetään oppivan pääasiassa näköaistinsa välityksellä. Seu-

raavaksi tärkeimmät aistit oppimisen kannalta ovat kuuloaisti ja tuntoaisti. Oppimista tehostaa se, että oppija saa tietoa samanaikaisesti useamman aistin kautta. Opetus on sitä konkreettisempaa ja sitä paremmin opittu asia pysyy myös muistissa, mitä useampaan aistiin opetus pohjautuu. [32, s. 47]

Oppijoilla voi olla oppimistapahtumassa erilaisia oppimistyylejä. Aktiiviselle osallistujalle osallistuminen ja toiminta ovat tärkeitä, hän on ihmisläheinen ja reagoi tunneperäisesti. Harkitseva tarkkailija keskittyy havainnointiin ja on selvillä ryhmän tapahtumista ja sisäisistä suhteista. Looginen ajattelija pyrkii ilmiöiden syiden ja seurausten ymmärtämiseen etsimällä teoriaa tai mallia joka selittää hänen havaintonsa. Kokeileva toteuttaja ideoi ja ottaa riskejä ja kokeilee mielellään uusia toimintamalleja. Parhaisiin oppimistuloksiin päästään käyttämällä kaikkia oppimistyylejä tarkoituksenmukaisesti. [32, s. 50–51]

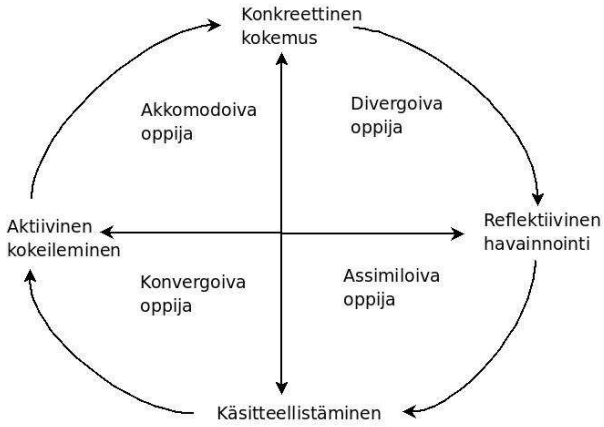
Opiskelulla voi olla sekä yksilöllisiä että yhteisöllisiä tavoitteita. Yksilölliset tavoitteet ovat niitä, mitä oppijoiden tulisi osata työskentelyn jälkeen. Yhteisölliset tavoitteet taas ilmaisevat, mitä yhteisöllisiä taitoja ryhmän tulisi työskentelyn aikana saavuttaa. Yksilölliset ja yhteisölliset tavoitteet täydentävät opetuksellisessa mielessä toisiaan vaikka ajankäytöllisesti ne usein kilpailevat keskenään. [32, s. 56]

Kolb [18] jakaa oppimistyylin välittömään kokemiseen (concrete experience), reflektoivaan havaitsemiseen (reflective observation), abstraktiin käsitteellistämiseen (abstract conseptualization) sekä aktiiviseen kokeilemiseen (active experimentation). Kuvassa 1 on kuvattu eri oppimistyylien suhdetta toisiinsa. Oppija käyttää näitä tyylejä oppimistilanteissa mahdollisuuksiensa ja tottumustensa mukaan. Kolb [19] luonnehtii oppimistapahtuman kehäksi, jossa kuljetaan eri tyylien välillä. Tätä kaikkia oppimisen tyylejä hyväksikäyttävää mallia kutsutaan kokemukselliseksi oppimiseksi.

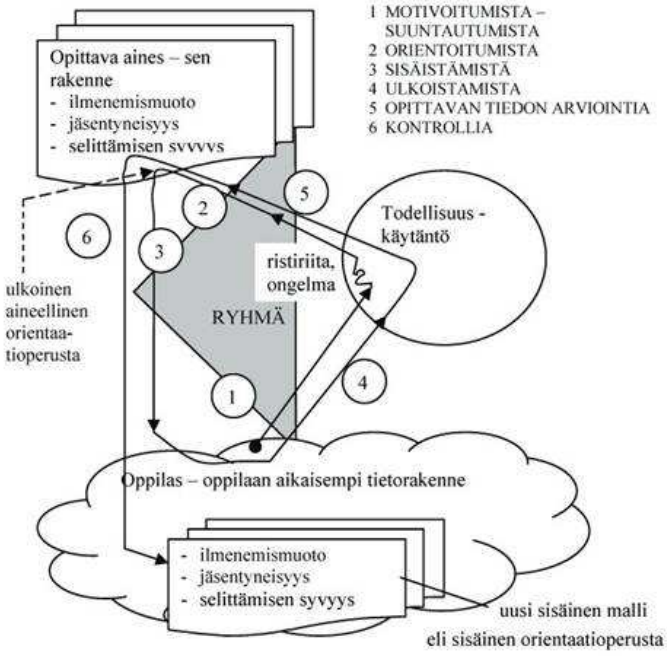
Oppijan tarkkaavaisuutta, valikointia ja tulkintaa suuntaavat hänen toiminta ja aikaisempi tietorakenne. Opiskelumotivaatio voi olla tilapäistä (uutuudenviehätys), vieraantunutta (palkkion toivossa tapahtuvaa) tai sisällöllistä (mielenkiinto sisältöä ja soveltamismahdollisuuksia kohtaan). Kuvassa 2 esitellään oppimisen täydennetty malli [21, s. 14–15] ja siinä motivoitumisella (1) tarkoitetaan mielenkiinnon heräämistä sisältöä kohtaan. Orientoitumisvaiheessa (2) luodaan ongelman ratkaisemiseen tarvittava periaateja tietorakennemalli. Sisäistämisvaiheessa (3) muokataan uuden tiedon avulla omaa ajattelu- ja toimintamallia. Ulkoistamisvaiheessa (4) sovelletaan opittua jonkun todellisen ongelman ratkaisuun. Mallia testatessa (5) arvioidaan tiedon pätevyyttä tarkastelemalla kriittisesti tiedon toimivuutta, ongelmakohtia ja soveltamisen rajoja. Kontrollointivaiheessa (6) kontrolloidaan opiskeluprosessia, arvioidaan oppimista ja sen tuloksia sekä tietoisesti pyritään parantamaan oppimismenetelmiä. [21, s. 14–15], [10, s. 49]

Sahlberg ja Shahran [27, s. 367] kuvaavat yhteistoiminnallista oppimista lähestymistavaksi, jossa opettaja on oppilaiden valmentaja, muodostaa ryhmiä, ohjaa vuorovaikutusprosesseja, kannustaa sekä arvioi toimintaa. Opettajan tehtävänä ei siis ole tehdä itseään tarpeettomaksi. Tavallinen ryhmätyö ei ole yhteistoiminnallista oppimista, koska siinä ryhmän jäsenellä ei ole yksilöllistä vastuuta. Yhteistoiminnallisessa oppimisessa jokaisen opiskelijan tulee ottaa vastuuta opiskelutoverin oppimisesta ja vasta toissijaisesti omasta oppimisestaan.

Yhteistoiminnallisessa oppimismene-



Kuva 1: Kolbin luokittelun neljä oppimistyyliä oppimisprosessin kehälle sijoitettuina [19, s. 77-78].



Kuva 2: Oppimisen täydennetty malli [21, s. 14].

telmässä oppilaat jaetaan 2–4 hengen ryhmiin. Ryhmien tulisi toiminnan kannalta olla heterogeenisia tiedollisilta, taidollisilta, sosiaalisilta ja kielellisiltä valmiuksiltaan. Ryhmän jäsenet ovat vastuussa muitten jäsenten menestymisestä ja samalla omasta oppimisestaan. [28, s. 67–68]

Yhteistoiminnallisen oppimisen edellytyksiä ovat Sahlbergin ja Shahranin [27] mukaan ryhmän jäsenten positiivinen keskinäinen riippuvuus, kannustava kasvokkain tapahtuva vuorovaikutus, yksilöllinen vastuu, sosiaaliset taidot ja ryhmässä tapahtuva prosessointi. Opiskelijoitten tulisi siis sitoutua ryhmän muihin jäseniin ja ymmärtää yhteisen onnistumisen olevan myös heidän oman onnistumisensa. Ryhmän toisten jäsenten tukeminen ja auttaminen mahdollistavat positiivisen riippuvuuden hengen ja hedelmällisen yhteistyön. Yksilöllistä vastuuta voidaan korostaa palauttamalla ryhmän jäsenten yksilölliset arviot sekä koko ryhmälle että opiskelijalle yksilönä, jolloin ryhmän jäsenet tietävät kohdistaa tukensa ja rohkaisunsa sen tarpeessa oleville. Sivustakatsojaksi yhteistoiminnallisessa oppimisessa ei voi jäädä, vaan jokaisen työpanos vaikuttaa lopputulokseen. Ryhmän sisäisessä keskustelussa arvioidaan sitä, kuinka hyvin tavoitteet on saavutettu ja onko ryhmän työskentely ollut tehokasta.

4 Ohjelmoinnin oppimisen haasteita

Ohjelmointia opiskelevien on opittava laatimaan ongelman ratkaiseva algoritmi ja koodattava se käytössä olevalla ohjelmointikielellä oikein toimivaksi suorituskelpoiseksi ohjelmaksi. Ala-Mutkan [1] mukaan ohjelmoinnin opettajan on syytä perehtyä niihin ongelma-alueisiin, joita ohjelmoinnin oppimisessa on.

Ohjelmoinnin opiskelijalla voi olla puutteellinen käsitys ohjelmoinnin olemuksesta. Opiskelija kuvittelee ohjelmoinnin opiskelun pelkäksi ohjelmointikielen opetteluksi eikä osaa panostaa muihin työvaiheisiin ja taitoihin. Tyydytään siihen, että ohjelmakoodi menee läpi kääntäjästä. Opiskelijat eivät ole tottuneet ohjelmoinnin vaatimaan systemaattisuuteen ja tarkkuuteen vaan yrittävät esimerkiksi korjata virheitä yrityksen ja erehdyksen kautta. Tämä saattaa tuottaa enemmän uusia virheitä kuin eliminoida entisiä. Ohjelmointiharjoituksissa tulisikin korostaa ohjelmakoodin lisäksi myös suunnitteludokumenttien ja testiaineistojen merkitystä. [1], [26, s. 137– 172]

Ohjelmointikielen ja -ympäristön valinta saattaa vaikeuttaa ohjelmoinnin oppimista, mikäli oppilas samalla joutuu keskittämään huomionsa ohjelmointiympäristöön. Työelämä vaikuttaa ohjelmointikielen valintaan jo alkeiskursseilta. Suosittujen teollisuuskielten – Java ja C++ – käyttäminen ohjelmoinnin peruskursseilla vaatii opettajalta huolellisuutta opetuksen suunnittelussa. Erityisesti Javalle on kehitetty monia apuvälineitä ja ympäristöjä helpottamaan ohjelmoinnin ensiaskeleiden ottamista [1].

Abstraktin ohjelmointikäsitteen oppiminen on yksi ohjelmoinnin oppimisen ongelmia. Joillekin ohjelmoinnissa käytetyille abstraktioille, esimerkiksi rekursiolle, ei reaalimaailmassa ole vastinetta ja näiden ymmärtäminen voi olla vaikeaa. Toisaalta opettajan käyttämät metaforat voivat saada aikaan virheellisiä tulkintoja. Javalle ja C++:lle on kehitetty apuvälineitä, jotka auttavat ymmärtämään yhteyden ohjelmakoodin ja koneessa tapahtuvan toiminnan välillä visualisoimalla ohjelman suoritusta [1]. Turnerin [31] mukaan erityisesti ne opiskelijat, joilta on vaikeuksia abstraktissa ajattelussa, koke-

vat ohjelmoinnin vain joukkona erilaisia lauseita ja poikkeuksia, eivätkä ymmärrä niistä muodostuvaa ohjelman toiminnallista kokonaisuutta. Näillä opiskelijoilla tuloksena on, että ohjelman kompleksisuus kasvaa eksponentiaalisesti suhteessa ohjelman kokoon.

Aloittelevan ohjelmoijan tulee osata ymmärtää olemassa olevaa ohjelmakoodia. Tämä taito auttaa ohjelmien testauksessa, arvioinnissa ja virheiden etsinnässä. Näitä taitoja olisi syytä opettaa. Erilaisilla automaattisesti harjoitustöitä tarkastavilla välineillä voidaan ohjata oppilasta tähän suuntaan, koska ohjelma antaa oppilaalle välittömän palautteen. Arviointitaitoja voidaan parantaa myös antamalla itse- ja vertaisarviointitehtäviä. Lisäksi voidaan teettää virheenetsintäharjoituksia [1].

Massaohjelmointikurssit ovat haastavia, koska oppilaiden taustat ja osaamistasot vaihtelevat. Oppilaat voivat passivoitua. Ohjelmointikielen valinta ohjelmoinnin peruskurssille on vaikeaa, samoin ohjelmointiparadigman (proseduraalinen vai olioajattelu) valinta. Ohjelmoinnin abstraktit käsitteet ovat vaikeita ymmärtää. Havainnollistavien visualisointityökalujen käyttö on vähäistä ja pintapuolista. [20, s. 8]

5 Ketterä ohjelmistokehitys

Ketterät (Agile) menetelmät ovat erilaisia ohjelmistokehitysmenetelmiä, joissa iteraatiokierrokset ovat hyvin lyhyitä ja sovelluskehitys jatkuvaa. Ketterien ohjelmistokehitysmenetelmien perustana pidetään Ketterän ohjelmistokehityksen manifestia (Agile Manifesto). Tämä manifesti julkaistiin vuonna 2001 Utahissa 17 keveiden ohjelmistokehitysmenetelmien asiantuntijoiden toimesta. Manifestin julkaisijat lupaavat etsiä parempia menetelmiä ohjelmistojen kehittämiseen ja auttaa tässä myös muita [4].

Ketteristä menetelmistä tunnetuin on Kent Beckin XP-menetelmä (Extreme Programming), jonka ominaisuuksia ovat muun muassa pariohjelmointi ja jatkuva testaaminen [11, s. 47]. Tämä menetelmä on kehitetty pienten organisaatioiden ja tiimien pienehköihin projekteihin. XP-ohjelmointiprosessin tarkoituksena on olla helposti mukautuva ja muutoksiin sopeutuva ja sen pyrkimyksenä on tuottaa mahdollisimman laadukasta ohjelmakoodia tehokkaasti. Dokumentaatio ja ohjelmistokehitysprosessin formaalius jäävät vähemmälle. Prosessin perustana ovat kommunikaatio sekä ratkaisuiden yksinkertaisuus. [30, s. 8–13]

”Extreme Programming on ohjelmistokehityksen alue joka perustuu neljään arvoon: yksinkertaisuus, kommunikaatio, palaute ja rohkeus. Se toimii tuomalla koko tiimin yhteen yksinkertaisten käytäntöjen avulla, antamalla tarpeeksi palautetta, jotta tiimi voi nähdä, missä he ovat menossa, ja mukauttaa käytäntönsä heidän ainutlaatuiseen tilanteeseensa.” [3]

XP-ohjelmointi kiteytyy käytäntöihin. Käytännöt pyrkivät kokoamaan ja yhdistelemään hyväksi havaittuja ohjelmoinnin menetelmiä toimivaksi kokonaisuudeksi, jossa toisten käytänteiden vahvuudet pyrkivät paikkaamaan toisten heikkouksia. Pariohjelmointi on oleellinen osa XP-ohjelmointia, sillä kaikki lopulliseen ohjelmistotuotteeseen päätyvä koodi tulisi olla tuotettu pariohjelmointina [3]. Tässä artikkelissa rajoitutaan käsittelemään XP-ohjelmoinnin käytännöistä pariohjelmointia ja sen soveltuvuutta ohjelmoinnin opettamiseen.

6 Pariohjelmointi

Pariohjelmointi on kahden ohjelmoijan välistä jatkuvaa vuorovaikutusta, jossa ohjelmakoodia tuotetaan, analysoidaan, suunnitellaan ja testataan [5, s. 42]. Käytännössä pariohjelmointi toteutetaan niin, että yhdessä työpisteessä työskentelee kaksi ohjelmoijaa, joista toinen (driver) kirjoittaa annetun tehtävän testiä tai ohjelmakoodia, jonka tarkoituksena on läpäistä tehtävälle etukäteen laaditut hyväksymistestit. Driver keskittyy ongelmanratkaisuun ja algoritmin kirjoittamiseen ohjelmakoodiksi. Driver myös selittää kirjoittamaansa ohjelmakoodia työparilleen. [2, 3], [30, s. 42]

Toinen (navigator, copilot), ei ole vapaamatkustaja vaan hän seuraa driverin toimia vierestä, vapauttaa driverin syntaksi- ja logiikkavirheiden tarkistamisesta sekä tekee tarvittaessa ohjelmakehitystä koskevia ehdotuksia. Hän myös tarvittaessa kiinnittää driverin huomion epäolennaisuuksista käsiteltävään asiaan. [2, 3], [30, s. 42]

Beck [5, s. 42] kertoo, että pariohjelmoinnissa ohjelmoijat:

- pitävät toistensa huomion kiinni senhetkisessä tehtävässä,
- pitävät keskenään ideapalavereja järjestelmästä ja sen kehittämisestä,
- selventävät ideoita toisilleen,
- pitävät toisensa velvollisina noudattamaan tiimin käytänteitä ja
- auttavat, kun partneri jää jumiin.

Beck [5, s. 42–43] sanoo, että pariohjelmointi ei kuitenkaan tarkoita sitä, ettei voi ajatella yksinään. Tarvitaan sekä yhteistyötä että työtoveruutta ja yksityisyyttä. Ideoita voidaan käsitellä myös yksinään, ja jatkaa niiden työstämistä koodiksi parin kanssa. Pariohjelmointi on myös rankkaa, eivätkä monet ohjelmoijat jaksa sitä täyttä työpäivää. Tähän viittaa myös XP-ohjelmoinnin perussääntö, 40 tunnin työviikko, jonka mukaan ohjelmoijan tulee myös levätä riittävästi [3] eikä ylitöitä saa tehdä kahtena viikkona peräkkäin. Beckin [5, s. 42–43] mukaan pareja täytyy kierrättää riittävän usein. Parhaisiin tuloksiin on päästy tunnin vaihtosykleissä, vaikeampia ongelmia ratkottaessa jopa puolen tunnin sykleissä.

6.1 Pariohjelmoinnin etuja

Nopeus ja virheettömyys ovat niitä pariohjelmoinnin etuja, joita on havaittu yliopistojen ohjelmoinnin kursseilla järjestetyissä kokeissa. Ohjelmointiaika lyhentyi, virheitä esiintyi vähemmän ja opiskelijat olivat tyytyväisiä pariohjelmointiin työtapana ja olivat valmiita käyttämään pariohjelmointia myös tulevaisuudessa. Pariohjelmoinnilla tuotetuissa ohjelmissa oli koodirivejä 10–30 prosenttia vähemmän huolimatta siitä, että ohjelmien toiminnoissa ei ollut eroja. Harjoitustehtävät olivat kuitenkin vain muutaman sadan rivin ohjelmia, joten näiden kokeiden perusteella ei voida esittää tarkempia arviota pariohjelmoinnin hyödyistä suurempien käytännön sovellusten kehittämisessä. [22, 35, 24]

Hanksin [12] mukaan opiskelijat kokivat pariohjelmoinnin käytön ansiosta oppineensa enemmän ja heidän mielestään luokkailmapiiri oli parempi. Ne opiskelijat, jotka luottivat tekemiinsä ohjelmaratkaisuihin, myös suhtautuivat pariohjelmointiin myönteisemmin. Naiset suhtautuivat pariohjelmointiin miehiä positiivisemmin. Harjoitusten vetäjällä havaittiin myös olevan vaikutusta suhtautumisessa pariohjelmointiin.

Opiskelijat olivat Hanksin ym. [13] tutkimuksessa tyytyväisempiä tekemiinsä harjoitustöihin. Enemmistö opiskelijoi-

ta oli valmis kokeilemaan pariohjelmointia myös tulevaisuudessa [12, 23]. Opiskelijat ovat kokeneet pariohjelmoinnin positiivisena. Yhdessä saman koneen ääressä istuminen kirjoitus- ja ohjausvuoroja vaihtaen helpottaa ongelmien ratkaisua ja opettaa ohjelmointia molemmille. Erilaiset etätyöpöytäsovellukset mahdollistavat pariohjelmoinnin toteuttamisen, vaikka yhteisen koneen äärellä olo ei fyysisesti onnistuisikaan [1].

Pariohjelmoinnissa ja kurssien läpäisyssä havaittiin selvä yhteys. Pariohjelmointimenetelmää käyttäneet opiskelijat saivat todennäköisemmin kurssin suoritettua kuin yksin ohjelmoineet [23]. Pariohjelmoijista yli 92% suoritti lopputentin kun taas yksin ohjelmoineista vain 76% [22]. Kuitenkaan merkittäviä eroja lopputentin arvosanoissa ei havaittu [33].

Steinbergin ja Palmerin [30] mukaan pariohjelmoinnin etuja ovat ohjelmakoodin reaaliaikainen tarkistaminen ja korjaaminen, erilaisten häiriötekijöiden välttäminen parin pitäessä toisensa kiinni käsiteltävässä asiassa, asioiden miettiminen ja hoitaminen yhdessä sekä tietämyksen ja informaation migraatio.

Kotavuopion [20, s. 16–17] tutkimuksessa todetaan, että ohjelmointiuransa alkuvaiheessa (Oulun ammattikorkeakoulun Johdatus ohjelmointiin -kurssi) olleet opiskelijat suhtautuivat pariohjelmointiin myönteisemmin kuin opinnoissaan pidemmälle ehtineet (Kajaanin ammattikorkeakoulun Ohjelmistoalgoritmit -kurssi). Tämä johtui tutkijan mielestä siitä, että ohjelmoijien tasoerot eivät olleet vielä ehtineet kasvaa ja tässä vaiheessa ohjelmoinnin opintoja pariohjelmointi katsottiin mielekkääksi. Jatko-opinnoissa katsottiin olevan eduksi, jos parit jaetaan aikaisempien kurssien arvosanojen perusteella niin, etteivät tasoerot parien sisällä muodostu suuriksi. Kotavuopio [20, s. 16–17] toteaa, että pariohjelmointimenetelmiä ohjelmoinnin opiskelussa käyttäneillä ei lopputentin arvosanoissa havaittu merkittäviä eroja yksin työskennelleiden arvosanoihin verrattuna. Ainoastaan lopputenttiin osallistumisprosentti oli pariohjelmointiryhmässä suurempi kuin yksin työskennelleiden ryhmässä.

Myös reaalimaailmassa ohjelmistokehitysprojektit ovat valmistuneet pariohjelmointia käyttämällä jopa 40–50 prosenttia nopeammin kuin yksin ohjelmoidut projektit [34]. Pariohjelmoinnissa myös ohjelmistojen laatu lisääntyi sekä työtyytyväisyys kasvoi, ohjelmat läpäisivät enemmän testitapauksia kuin yksin ohjelmoidut ohjelmistot. Ohjelmistoprojektin kokonaiskustannukset pienenivät, koska virheet ja sitä kautta ylläpito- ja tuotetukikustannukset vähenivät, ja tätä kautta kustannustehokkuus kasvoi. Myös ohjelmoijien ohjelmointitaito parani [7]. Virheiden määrä on pariohjelmoinnilla saatu vähenemään jopa tuhannenteen osaan ja tehokkuus kasvamaan jopa aikaisempien ohjelmistoprojektien mittaustuloksiin verrattuna 127 prosenttia 10 työntekijän ja 50 000 koodirivin projektissa [15].

6.2 Pariohjelmoinnin haasteita

Pariohjelmoinnissa hankaluuksia ovat tuottaneet aikataulujen yhteensovittaminen [17]. Santa Cruzissa tehdyssä tutkimuksessa [6, s. 25–27] pyydettiin kurssin opiskelijoita ilmoittamaan parin toistuvasta epäluotettavuudesta tai aikataulujen yhteensovittamisongelmista. Opiskelijat kuitenkin saattoivat jatkaa ongelmista huolimatta viikkoja ennen niiden julkituomista. Parien uudelleen organisointi oli näin hankalaa. Ongelmatapauksissakin opiskelijat olivat valmiita palauttamaan ohjelmointityön molempien nimillä varustettuna, vaikka toisen työpanos oli ollut olematon. Tutkimus siis paljasti opiskeli-

joiden epärehellisyyden työnjaon suhteen [6, s. 25–27].

Sandersin tutkimuksessa [29, s. 235–250] vaikeuksia oli myös parien keskinäisessä kommunikaatiossa. Nämä kuitenkin selvitettiin parin sisällä lukukauden aikana. Pariohjelmointiuran alussa ohjelmakoodin kirjoittajasta voi tuntua siltä, että hän on ”tarkkailun alla” [30, s. 37-38]. Sandersin tutkimuksessa [29, s. 235–250] kysyttiin kurssin alussa ja lopussa pariohjelmoinnin soveltuvuudesta kyseiselle kurssille. Alun myönteinen suhtautuminen ja innostus hiipuivat lukukauden loppuun mennessä. Useimpien opiskelijoitten mielestä pariohjelmointi on arvokas käytäntö, mutta että sitä tulisi soveltaa vasta myöhemmillä ohjelmointikursseilla. Myös tässä tutkimuksessa tuli esille se, että parien tulisi olla mahdollisimman samantasoisia osaamiseltaan ja aikatauluongelmat tulisi välttää säännöllisillä laboratoriosessioilla.

Ohjelmointiparien valinnassa tulee ottaa huomioon erilaiset inhimilliset ja kulttuurilliset lähtökohdat parin jäsenten välillä. ”Henkilökohtaisen tilan” tarve pariohjelmoinnissa saattaa kulttuurillisista syistä olla erilainen ja esimerkiksi italialaisen ja tanskalaisen ohjelmoijan tilantarve ja etäisyysvaatimukset ohjelmoijapariin voivat olla hyvinkin erilaisia. Jopa henkilökohtaisen hygienian käsityksen erot voivat aiheuttaa ongelmia. Ohjelmointiparin jäsenten tulee myös olla henkisesti riittävän kypsiä parityöskentelyyn. Yksilöllisiä eroja tulee pystyä kunnioittamaan. Mikäli parin kanssa on epämukavaa työskennellä, tilanne täytyy keskustella auki, sillä muuten pariohjelmointi ei toimi niin kuin pitäisi ja työn tulokset eivät ole parhaat mahdolliset [5, s. 43].

7 Pohdinta

Pariohjelmointi on yksi osa Extreme Programming -ohjelmistokehitystä. Pariohjelmoinnin ideana on, että kaksi ohjelmoijaa työskentelee yhdellä tietokoneella yhden ohjelmointitehtävän parissa. Toinen kirjoittaa ohjelmakoodia keskittyen algoritmin muuntamiseen hyväksymistestin läpäiseväksi ohjelmakoodiksi ja toinen tarkkailee tuotettua koodia syntaksi- ja logiikkavirheiden varalta tehden tarvittaessa huomautuksia ja korjausehdotuksia. Ohjelmoijat vaihtavat näitä rooleja tunnin, jopa puolen tunnin välein.

Ohjelmoinnin opettamisessa ja oppimisessa on monia haasteita. Aloitteleva ohjelmoinnin opiskelija käsittää ohjelmoinnin opiskelun pelkäksi ohjelmointikielen oppimiseksi eikä hän ymmärrä esimerkiksi suunnittelu- ja testausvaiheiden merkitystä. Ohjelmointitaito saattaa näin jäädä ”koodaa ja korjaa” -asteelle. Ohjelmointiympäristöt ja -kielet saattavat olla opetuksen kannalta epätarkoituksenmukaisia. Erilaiset abstraktit käsitteet saattavat jäädä opiskelijalle epäselviksi, mikäli opettaja käyttää huonosti valittua metaforaa, esimerkiksi laatikkoa kuvaamaan muuttujaa. Oppilaan tulisi myös oppia lukemaan toisten kirjoittamaa ohjelmakoodia. Lisäksi on havaittu, että opiskelijat oppivat toisiltaan ja harjoitusten vetäjälle esitetään paremmin muotoiltuja kysymyksiä.

Oppimista on tutkittu paljon ja erilaisia oppimisen teorioita on kehitetty useita. Yhteistoiminnallisen oppimisen malli ja sosiokulttuurinen malli näyttäisivät olevan lähinnä pariohjelmoinnin ideaa. Molemmissa näissä oppiminen tapahtuu sosiaalisen kanssakäymisen ja yhteistyön kautta. Iloinen yhteistyö ja yhteiset päämäärät auttavat tuloksiin niin oppimisessa kuin pariohjelmoinnissakin. Yhteistoiminnallisessa oppimisessa vastuuta ote-

taan ryhmän sisällä myös toisten ryhmän jäsenten oppimisesta ja opettajalle jää näin eräänlaisen ohjaajan rooli.

Pariohjelmoinnissa ja sen käytössä on myös erilaisia haasteita. Sopivien ohjelmointiparien muodostaminen erilaisten kulttuuri-, oppimistyyli- ja muiden taustojen vuoksi voi olla haastavaa. Parin sisäiset osaamisen tasoerot voivat olla joko haaste tai rikkaus. Tästä kirjallisuudessa löytyi ristiriitaisia havaintoja ja asiaa voisi vielä tutkia tarkemmin. Sitoutuminen yhteisiin tavoitteisiin voi pariohjelmoinnissa ja erityisesti ohjelmoinnin opiskelussa olla haaste. Ahkera opiskelija ei välttämättä ilmianna parin vapaamatkustajaa. Roolien vaihtoa kuin myös muita pariohjelmoinnin sääntöjä tulisi noudattaa. Sopivien opetusprojektien ja -tehtävien laadintaan tulee perehtyä, jotta opiskelijat säilyttävät innostuksensa pariohjelmointioppimiseen.

8 Johtopäätökset

Pariohjelmointi näyttäisi tietyin edellytyksin olevan käyttökelpoinen ratkaisu ohjelmoinnin opetukseen, koska se tuottaa parempaa ja virheettömämpää ohjelmakoodia nopeammin kuin yksinohjelmointi. Pariohjelmointi tulee kuitenkin ensin opettaa oppilaille ja sitä tulee noudattaa ohjelmoinnin harjoitustöissä, ettei esimerkiksi synny vapaamatkustaja-ajatuksia.

Kouluympäristössä ja lyhyillä ohjelmoinnin kursseilla voisi olla hyvä, että ohjelmointipareja vaihdeltaisiin eri harjoituksissa. Tällä tavalla oppilaat oppisivat toimimaan oppimistyyleiltään ja henkilökohtaisilta ominaisuuksiltaan erilaisten ihmisten kanssa, mikä olisi kasvattava kokemus. Pariohjelmointi tuo lisäarvoa ohjelmoinnin opetukseen parantamalla laatua, nopeuttamalla sovelluskehitystä ja kaupallisessa mielessä pienentää valmiiden ohjelmistojen kokonaiskustannuksia.

Havaitsimme kirjallisuudessa joitakin ristiriitoja, muun muassa sen, että toiset tutkimukset puhuivat parien heterogeenisuuden, toiset parien homogeenisuuden puolesta. Tässä olisi selvästi jatkotutkimuksen paikka. Muita jatkotutkimuksen aiheita voisi olla ohjelmoijaparien eri oppimistyylien vaikutukset pariohjelmoinnin tuloksiin sekä ryhmätyöhön.

Viitteet

[1] Ala-Mutka, K. Ohjelmoinnin opetuksen ongelmia ja ratkaisuja. Tampereen teknillinen korkeakoulu, Ohjelmistotekniikan laitos, 2004.

[2] Astels, D., Miller, G. & Novak, M. A Practical Guide to Extreme Programming. Prentice Hall PTR. 2002.

[3] Beck, K. Extreme Programming Explained. Addison Wesley Professional. 1999.

[4] Beck, K., Beedle, M., van BenneKum, A. Cockburn, A. Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, R., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. Manifesto for Agile Software Development, 2001. Available: http://agilemanifesto.org [Referenced 14.4.2008].

[5] Beck, K. & Anders, C. Extreme Programming Explained: Embrace Change– Second Edition. Addison-Wesley Longman. Boston. 2004.

[6] Bevan, J., Werner, L., & McDowell, C. Guidelines for the Use of Pair Programming in a Freshman Programming Class. Proc. of the 15th Conference on Software Engineering Education and Training. 2002. 25–27.

[7] Cockburn, A. & Williams, L. The Costs And Benefits of Pair Programming. XP. 2000.

[8] DeClue, T. H. Pair programming and Pair Trading: Effects on Learning and Motivation in a CS2 course. J. Comput. Small Coll. 18(5), 2003. 49–56.

[9] Dybå, T., Arisholm, E., Sjoberg, D., Hannay, J. & Shull, F. In Arisholm E. (Ed.). Are Two Heads Better than One on the Effectiveness of Pair Programming (J. E. Hannay Trans.). IEEE Software. 2007. 12–15.

[10] Engeström, Y. Perustietoa opetuksesta. Valtion painatuskeskus. 1988.

[11] Haikala, I. & Märijärvi, J., Ohjelmistotuotanto. Talentum Media Oyj 10. uudistettu painos. 2004.

[12] Hanks, B. Student Attitudes Toward Pair Programming. ITICSE '06: Proceeding of the 11th Annual SIGSCE Conference on Innovation and Technology in Computer Science Education, Bologna, Italy. 2006. 113–117.

[13] Hanks, B., McDowell, C., Draper, D. & Krnjajic, M. Program Quality with Pair Programming in CS1. ITiCSE '04: Proceedings of the 9th Annual SIGCSE Conference in Innovation and Technology in Computer Sciences Education, Leeds, United Kingdom. 2004. 176–180.

[14] Jeffries, R., Anderson, A. & Hendrickson, C. Extreme Programming Installed. Addison-Wesley. 2000.

[15] Jensen, R. A Pair Programming Experience. Crosstalk. 2003.

[16] Katira, N., Williams, L., Wiebe, E., Miller, C., Balik, S. & Gehringer, E. On Understanding Compatibility of Student Pair Programmers. SIGCSE '04: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education. Norfolk. Virginia. USA. 2004. 7–11.

[17] Kivi, J., Haydon, D., Hayes, D., Schneider, R. & Succi, G. Extreme Programming: A University Team Design Experience. IEEE. 2000.

[18] Kolb, D. Learning Style Inventory Technical Manual. Boston: McBer and Co. 1976.

[19] Kolb, D. Experiental Learning. Experiences as the Source of Learning and Development. New Jersey: Prentice Hall PTR. 1984.

[20] Kotavuopio, M. Pariohjelmoinnin hyödyt ohjelmoinnin opetuksessa. Pro Gradu -tutkielma. Oulun yliopisto, Tietojenkäsittelytieteen laitos. 2008.

[21] Leskinen, P. Matematiikan oppimista tukevia kognitiivisia työkaluja. Helian julkaisusarja C: 17. 2006.

[22] McDowell, C., Werner, L., Bullock, H. & Fernald, J. The Effects of Pair-Programming on Performance in An Introductory Programming Course. ACM SIGCSE Bulletin 34, 1. 2002.

[23] Mendes, E., Al-Fakhir, L. & Luxton-Reilly, A. A Replicated Experiment of Pairprogramming in a 2nd-year Software Development and Design Computer Science Cource. ITICSE '06: Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, Bologna, Italy. 2006. 108–112.

[24] Preston, D. Adapting Pair Programming Pedagogy For Use In Computer Littaracy Cources. 2006.

[25] Rauste-von Wright, M., von Wright, J. & Soini, T. Oppiminen ja koulutus. 9. uudistettu painos. Helsinki; WSOY. 2003.

[26] Robins, A., Rountree, J. & Rountree, N. Learning and Teaching Programming: A Review and Discussion. Computer Science Education, 13(2). 2003.

[27] Sahlberg, P & Sahran, S. (toim.). Yhteistoiminnallisen oppimisen käsikirja. Helsinki, WSOY. 2002.

[28] Sahlberg, P. & Leppilampi, A. Yksinään vai yhteistoimin. Vantaa, Vantaan täydennyskoulutuslaitos. 1994.

[29] Sanders, D. Extreme Programming: The Student View. Computer Science Education, 12-3. 2002. 235–250.

[30] Steinberg, D. & Palmer, D. Extreme Software Engineering: A Hands-on Approach. Pearson Education Inc., Upper Saddle River, New Jersey. 2004.

[31] Turner, J. A. & Zachary, J. L. Javiva: A Tool for Visualizing and Validating Student-Written Java Programs. The Proceedings of the Thirty Second SIGCSE Technical Symposium on Computer Science Education, Charlotte, North Carolina. 2001. 45-49.

[32] Vuorinen, I. Tuhat tapaa opettaa. Suomen Morenoinstituutin julkaisusarja nro 1. 1993.

[33] Werner, L. L., Hanks, B. & McDowell, C. Pair-Programming Helps Female Computer Science Students. ACM Journal on Educational Resources in Computing. 4(1), 2004.

[34] Williams, L., Kessler, R., Cunningham, W. & Jeffries, R. Strengthening the Case for Pair-Programming. IEEE Software. 2000.

[35] Williams, L. & Upchurch R. In Support of Student Pair-programming. ACM SIGCSE Bulletin 33, 1. 2001.