Pariohjelmoinnin taloudelliset hyödyt

Ville Knuuttila

Kandidaatintutkielma
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 6. toukokuuta 2013

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Ville Knuuttila

Työn nimi — Arbetets titel — Title

Pariohjelmoinnin taloudelliset hyödyt

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Kandidaatintutkielma	6. toukokuuta 2013	18

Tiivistelmä — Referat — Abstract

Tutkielmassa tutustutaan pariohjelmoinnin taloudellisiin hyötyihin. Pariohjelmoinnilla tiedetään olevan positiivinen vaikutus ohjelmakoodin laatun, mutta vie keskimääräisesti enemmän henkilötyötunteja, kuin yksin ohjelmoidessa. Onko pariohjelmointi taloudellisesi varteenotettava ohjelmistokehitysmuoto? Selvitämme paraneeko ohjelmakoodin laatu niin paljon että ohjelman ylläpitovaiheessa saadaan koravattua ylimenneet henkilötyötunnit toteutusvaiheesta.

Avainsanat — Nyckelord — Keywords
pariohjelmointi, taloudellisuus

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

1 Johdanto

Pariohjelmointi on ohjelmointimenetelmä, jossa kaksi ohjelmoijaa istuvat saman koneen ääressä ohjelmoimassa [Wil01]. Pariohjelmoinnissa ohjelmoijilla on kaksi eri roolia: kontrolloija ja tarkkailija. Kontrolloija on henkilö, joka kirjoittaa ohjelmakoodia eli käyttää näppäimistöä ja hiirtä. Tarkkailija istuu kontrolloijan vieressä nähden monitorin kokonaan ja etsii virheitä koodista.

Pariohjelmoinnilla on todettu olevan ohjelmakoodin laatuun ja parien ongelmanratkaisukykyyn positiivisia vaikutuksia [WK00]. Ongelmat ratkeavat jopa 60 prosentissa siitä ajasta, mitä yksin ohjelmoivat henkilöt käyttävät saman ongelman ratkaisuun [HDAS09]. Virheiden määrä ohjelmissa myös pienenee jopa neljäsosan [WKCJ00]. Näiden lisäksi pariohjelmoidessa ohjelmoijalla on suurempi kynnys käydä tarkistamassa omaa henkilökohtaista sähköpostiaan tai käydä lukemassa aiheeseen liittymättömiä verkkosivuja [WK03]. Ohjelmoijat kuitenkin arvostavat toistensa aikaa, eivätkä halua tuhlata sitä omilla henkilökohtaisilla asioillaan. Näin ollen myös keskittyminen pysyy paremmin ongelmanratkaisussa tai tehtävän toteuttamisessa.

Taloudellisuus on nyky-yhteiskunnassa merkittävä tekijä. Jos pariohjelmointi voidaan todeta taloudellisesti kannattavaksi, voitaisiin se ottaa laajemmin käyttöön myös yritysmaailmassa. Tällöin voisimme tulevaisuudessa nauttia laadukkaammasta ja virheettömämmästä ohjelmakoodista.

Toisaalta jos pariohjelmointi ei ole taloudellisesti kannattavaa, niin voitaisiin sen käyttöä entisestään vähentää. Pariohjelmointia voitaisiin käyttää vain sillioin kun siitä on hyötyä muuten, esimerkiksi jos ongelma on todella haastava tai yrityksellä on uusi työntekijä, joka pitäisi totuttaa uuteen ohjelmakoodikantaan.

Pariohjelmoijat kuitenkin käyttävät keskimäärin 120-150 % enemmän henkilötyötunteja ongelman tai tehtävän ratkaisuun kuin yksin ohjelmoivat

henkilöt [WU01]. Tämä tarkoittaa sitä, että esimerkiksi kaupallisen ohjelman ohjelmointi tulee maksamaan työnantajalle ohjelman toteutusvaiheessa jopa 1,5 kertaa enemmän, jos ohjelma ohjelmoidaan käyttäen pariohjelmointia. Ohjelmankoodin hyvä laatu helpottaa ohjelman ylläpitoa ja jatkokehitystä [PO95]. Näin ollen ylläpitovaiheeseen tarvittavat tunnit pienenevät. Tämän tutkielman tavoitteena on tarkastella pariohjelmoinnin taloudellista kannattavuutta henkilötyötunneissa mitattuna.

2 Pariohjelmointi

Pariohjelmoinnin määrittelyn mukaan pariohjelmointi on sitä, kun kaksi ohjelmoijaa työstävät samaa tehtävää tai ongelmaa yhdellä tietokoneella [NW01]. Tässä kappaleessa käsittelemme pariohjelmoinnin historiaa, sen prosessia ja parivariaatioita ohjelmoijien kokemuksen perusteella, sekä pariohjelmoinnin hyötyjä ja haittoja.

2.1 Pariohjelmoinnin historia

Pariohjelmointia on harrastettu pidempään kuin sitä on edes kutsuttu pariohjelmoinniksi [WK03]. Varhaisimpia viittauksia pariohjelmointiin löytyy vuosilta 1953-1956 [Wil96]. 1980-luvulla tehdyissä tutkimuksissa havaittiin, että yritysmaailmassa ohjelmoijat käyttävät suurimman osan ajastaan tehden töitä muiden ihmisten kanssa [LD87]. Vain 20 prosenttia työajasta käytetään yksilöohjelmointiin, 50 prosenttia käytetään parin kanssa ja 30 prosenttia kahden tai useamman henkilön kanssa työn tekemiseen.

1990-luvun puolivälissä ohjelmistotuotannossa ruvettiin olemaan yhä enenevissä määrin kiinnostuneita ketteristä ohjelmistotuotantomenetelmistä [Mar03]. Pariohjelmointikin nousi sen seurauksena pinnalle, kun se listatiin yhtenä XP-ohjelmistokehityksen kahdestatoista käytänteestä [Bec00].

2.2 Pariohjelmoinnin prosessi

Pariohjelmointi prosessina on hyvin yksinkertainen. Siinä toinen parista kirjoittaa ohjelmakoodia ja toinen katselmoi sitä jatkuvasti. Parien olisi syytä vaihtaa rooleja riittävän usein [Bec00].

Ohjelmointikokemuksen perusteella parityypit pystytään jakamaan kolmeen eri variaatioon: kokenut-kokenut, kokenut-kokematon ja kokematon-kokematon. Tämän lisäksi pariohjelmoinnille tyyppilisenä piirteenä on roolien, kontrolloija ja tarkkailija, vaihtaminen tietyin väliajoin [WWY+02].

Ohjelmoinnin opetukseen ja koodikantaan tutustumiseen hyödyllisin variaatio on, että kontroillajana toimii kokenut ohjelmoija [CH07]. Tällöin pelkästään kirjoittamalla ohjelmakoodia kontrolloija kykenee opettamaan kokemattomammalle tarkkailijalle hyviä käytänteitä ja koodikannan sisältöä. Erityisen tärkeää on kiinnittää huomiota siihen, että vuorovaikutus pysyy molempiin suuntiin. Tarkkailijan pitää kyetä kertomaan omat mielipiteensä kokeneemalle kontrolloijalle.

Nopeimman ongelmaratkaisukyvyn saavuttaa, jos sekä kontrolloija että tarkkailija ovat molemmat kokeneita koodaajia [Voa01]. Jos ongelma on haastava, ohjelmakoodin laatu paranee entisestään, mutta vaatii henkisesti suuremman ponnistuksen. Todella yksinkertaisiin tehtäviin voi kuitenkin mennä enemmän aikaa kuin yksin ohjelmoidessa.

Uusille ohjelmoijille hyvä tapa opettaa luettavan koodin kirjoittamista on pistää sekä tarkkailijaksi että kontrolloijaksi aloitteleva ohjelmoija [CH07]. Tämä pakottaa kontrolloijan kirjoittamaan ohjelmakoodinsa sellaiseksi, että tarkkailija saa siitä selvää. Huonona puolena on, että ongelmanratkaisukyky ei paljoa nopeudu verrattuna yksinohjelmoivaan aloittelijaan.

Parivariaatio	Hyödyt	Haitat
Kokenut - Kokenut	Nopea ongelmanratkaisu	Yksinkertaiset ongelmat saattavat viedä huomattavasti enemmän aikaa
Kokenut - Kokematon	Opetustarkoitus ja uuteen ohjelmakoodikantaan tutustuttaminen	Vuorovaikutus vaikeaa, ja siihen pitää kiinnittää paljon huomiota
Kokematon - Kokematon	Opettaa kirjoittamaan luettavaa koodia	Ongelman ratkaisu kestää lähes yhtä kauan kuin yksin ohjelmoivalla

2.3 Pariohjelmoinnin hyödyt ja haitat

Pariohjelmoidessa kaksi ihmistä istuvat saman tietokoneen ääressä tekemässä samaa tehtävää, tai toisin sanoen kaksi ihmistä tekevät yhden ihmisen työt. Tämä herättää kysymyksen, mitkä ovat pariohjelmoinnin hyödyt ja haitat todellisuudessa. Pariohjelmoinnilla on pakko olla hyötyjä [PM03], tai sitä ei muuten harjoitettaisi, eikä Kent Beck olisi varmasti nimennyt sitä XP-ohjelmistokehityksen yhdeksi käytänteeksi.

Mitkä ovat sitten pariohjelmoinnin tunnetuimmat haitat. Yleensä ottaen johtotason henkilöt näkevät ohjelmoijat harvana resurssina [CW00] eivätkä halua hukata näitä resursseja keskittämällä useampaa ohjelmoijaa tekemään samaa osaa ohjelmakoodista. Ohjelmointia ollaan myös yleensä opetettu yksinäisen aktiviteettina, josta johtuu monen kokeneen ohjelmoijan haluttomuus ohjelmoida muiden ihmisten kanssa. Osa jopa pitää omaa ohjelmakoodiaan niin persoonallisena, että toinen ihminen vain hidastaisi heitä. Tämän lisäksi yksinkertaisiin tehtäviin saattaa mennä melkein yhtäkauan kuin yhdellä

ohjelmoijallakin menisi.

Ongelmanratkaisunopeus on yksi tunnetuimmista pariohjelmoinnin hyödyistä [CW00]. Mitä kokeneempia ohjelmoijia ja haastavampia ongelmia, sitä suuremman hyödyn pariohjelmoinnista voi saada. Tämän lisäksi pariohjelmointia käyttäen tuotettu ratkaisu on ohjelmakoodin rivimäärältä yleensä huomattavasti pienempi, jopa $\frac{4}{5}$ siitä mitä yksinohjelmoidessa tehtynä. Ohjelmakoodin rivimäärän vähäisyys johtaa myös siihen että laatu on usein parempaa kuin yksin ohjelmoidessa. Näin ollen pariohjelmoinnilla tuotettujen ohjelmien jatkokehitys ja ylläpito on helpompaa.

3 Taloudellisuus ohjelmistotuotannossa

Ohjelmistotuotanto koostuu kahdesta eri vaiheesta: toteutuksesta ja ylläpidosta [SK98]. Ohjelmiston kehitys alkaa aina toteutusviheella, joka koostuu neljästä vaiheesta: suunnittelu, toteutus, testaus ja käyttöönotto. Ylläpitovaihe seuraa aina toteutusta. Kun ohjelma on otettu käyttöön, niin siirrytään ylläpitovaiheeseen. Ylläpito voi olla täysin olematonta, mutta silti käyttöönoton jälkeistä aikaa kutsutaan ylläpitovaiheeksi. Toteutusvaiheeseen voidaan palata ylläpitovaiheesta. Termi, jota tästä käytetään on jatkokehitys. Jatkokehitys ei sinäänsä eroa toteutuksesta mitenkään muuten kuin että ohjelmalle ollaan vähintään kerran suoritettu jo käyttöönotto ja siirrytty ylläpitovaiheeseen.

Ohjelmistotuotannossa suurin osa taloudellisista menoista tulee henkilöstöstä [HM95]. Tietokoneiden ja muiden tarvittavien laitteiden osuus kuluista on huomattavasti pienempi kuin mitä työntekijöiden vaatimat kustannukset. Suomessa henkilöstökustannukset yleensä koostuvat työntekijöille maksettavasta palkasta, vakuutuksista ja eläkemaksuista. Osa kustannuksista tulee myös työntekijöiden viihdyttämisestä, koska on todettu, että tyytyväinen

työntekijä on yritykselle arvokkaampi kuin tyytymätön [ARS08].

Suurimpia kustannuksia voidaan siis mitata laskemalla aikaa, joka ohjelmiston toteutukseen ja ylläpitoon menee. Kuinka nopeasti suunnittelu, toteutus, testaus ja käyttöönotto saadaan toteutettua ja kuinka helppoa ylläpito on. Mitä vähemmän virheitä ohjelman toteutusvaiheessa ohjelmakoodiin jää, sen helpompaa ja vähäisempää ohjelman ylläpito on, eli vie vähemmän aikaa työntekijöiltä.

Luettava ja hyvälaatuinen ohjelmakoodi johtaa siihen, että ohjelman jatkokehittäminen on helpompaa [Joh94]. Näin ollen, jos ohjelmaa jatkokehitetään, tulee halvemmaksi, jos alkuperäisen ohjelman ohjelmakoodi on kirjoitettu laadukkaammaksi [Fag01]. Koodikatselmointi on tapa, jolla koodin laatua pystytään parantamaan ja valvomaan. Katselmoidessa joku toinen ohjelmointitaitoinen henkilö lukee kirjoitetun ohjelmakoodin läpi ja merkitsee ohjelmakoodiin virheet tai huonot käytänteet, jotka varsinainen ohjelmoija sen jälkeen korjaa. Katselmoinnilla voidaan vähentää ohjelman kehityksen vaatimaa aikaa jopa 10-30% [GGF93]

4 Kysymys

Pariohjelmoinnissa toteutusvaiheessa käytetään enemmän henkilötyötunteja kuin yksin ohjelmoidessa [CW00]. Tässä tutkielmassa tarkastelen, paraneeko ohjelmakoodin laatu niin paljon, että ylläpitovaiheessa tarvittavilla henkilötyötunneilla saadaan yhteensä tarvittavat henkilötyötunnit pienemmäksi pariohjelmoidessa kuin yksilöinä ohjelmoidessa. Tarkastelu pohjautuu jo olemassa oleviin tutkimuksiin.

$X_{total} = Yksilöohjelmoinnin\ henkilötyötunnit$

$X_{impl} = Yksilöohjelmoinnin\ toteutusvaiheen\ henkilötyötunnit$

$X_{maint} = Yksilöohjelmoinnin\ ylläpitovaiheen\ henkilötyötunnit$

$Y_{total} = Pariohjelmoinnin\ henkilötyötunnit$

$Y_{impl} = Pariohjelmmoinnin\ toteutusvaiheen\ henkilötyötunnit$

$Y_{maint} = Pariohjelmoinnin\ ylläpitovaiheen\ henkilötyötunnit$

$X_{total} = X_{impl} + X_{maint}$

$Y_{total} = Y_{impl} + Y_{maint}$

$\Rightarrow Y_{total} < X_{total}$

Ohjelmistotuotannon suurimmat kustannukset tulevat henkilöstökuluista [HM95], joten jonkun tietyn ohjelmistotuotantomenetelmän taloudellinen kannattavuus on tietyllä tavalla mahdollista laskea tarkastelemalla sen vaativia henkilöstö resursseja. Näistä henkilötyötunnit ovat helpoiten mitattavia yksiköitä. Näin ollen tarkastelemalla: onko pariohjelmoinnin henkilötyötunnit pienemmät kuin yksilöohjelmoinnin henkilötyötunnit, saadaan kuva siitä onko pariohjelmointi taloudellisesti kannattavaa.

5 Pariohjelmoinnin taloudellisuus

Toteutusvaiheessa pariohjelmointi saattaa tuottaa jopa 120-150% henkilötyötunteja verrattuna yksilöohjelmointiin [WU01]. Jos on aivan täysin varmaa, että ohjelmaa ei olla ylläpitämässä tai jatkokehittämässä, niin pariohjelmointia ei kannata käyttää taloudelliselta näkökulmalta ohjelman ohjelmakoodin kirjoittamiseen. Yleensä tämä ei kuitenkaan ole se tapaus, miten ohjelmia lähdetään toteuttamaan, vaan etenkin yritymaailmassa ohjelmat ovat tarkoitettu käytettäväksi. Näin ollen pariohjelmoinnin taloudellisuuden tarkastelussa oletamme, että ohjelmaa tullaan ylläpitämään ja mahdollisesti myös jatkokehittämään.

5.1 Toteutusvaihe

Toteutusvaiheeseen kuuluu suunnittelu, toteutus, testaus ja käyttöönotto [SK98]. Näistä vaihesta pariohjelmointia voidaan käyttää hyödyksi toteutuksessa ja testauksessa. Suunnittelu ja käyttöönotto toteutetaan yleensä isommissa ryhmissä. Näin ollen niitä ei tulla tarkastelemaan tässä, vaan oletetaan, että ne ovat yhtä aikaa vieviä kuin yksinohjelmoidessa.

Pariohjelmoinnin avulla ongelmat ratkeavat nopeammin kuin yksin ohjelmoidessa [PM03]. Näin ollen toteutuksen kustannukset eivät ole tuplasti sitä, mitä yksinohjelmoidessa, kuten skeptisimmät epäilijät ovat väittäneet [CW00]. Pariohjelmointi kuluttaa pahimmillaankin vain noin 150% siitä henkilötuntimäärästä, jota yksinohjelmoivat käyttävät.

Testausvaiheessa voidaan manuaalinen koodikatselmointikin jättää toteuttamatta, koska sitä ollaan toteutettu koko ohjelmakoodin toteutuksen ajan, ja näin siinäkin säästetään henkilötyötunteja [NW01]. Samalla myös pystytään paremmin takaamaan, että ohjelmakoodin laatu on vaadittavalla tasolla. Katselmointi ja siihen vastaaminen on myös nopeampaa [CW00], koska pariohjelmoidessa katselmointia tapahtuu jatkuvasti, eikä palautetta tarvitse odottaa. Virheet voidaan havaita jopa samalla hetkellä, kun ne kirjoitetaan, ja tällä voidaan säästää aikaa, joka menisi koodin kääntämisen odottamiseen. Pelkästään sillä, että pariohjelmointi pakottaa jatkuvaan koodinkatselmointiin, voidaan säästää 10-30% ohjelman kehityksen vaatimasta ajasta [GGF93].

Pariohjelmoinnilla on todettu olevan myös muita epäsuoria vaikutuksia ohjelmoijiin ja ohjelmakoodin kirjoittamiseen vaadittavaan aikaan [HA05]. Näistä yksi merkittävimmistä on ohjelmoijien viihtyvyyden paraneminen. Tyytyväinen työntekijä on yritykselle paljon arvokkaampi [ARS08] ja saa enemmän aikaan kuin tyytymätön työntekijä. Kun ohjelmoija viihtyy teh-

tävänsä parissa, ohjelmakoodin kirjoittamiseen vaadittava aika pienenee. Eikä ohjelmoijan keskittyminenkään herpaannu aivan niin helposti, kun motivaatiota tehtävän tekemiseen löytyy.

5.2 Ylläpitovaihe ja jatkokehitys

Ylläpito ja jatkokehitys ovat vaiheta, joita toteutetaan ensimmäisen käyttöönoton jälkeen. Niitä on helpompi toteuttaa mitä luettavampaa ja parempilaatuisempaa ohjelmakoodi on [CW00]. Ohjelmakoodiin jääneiden virheiden määrä kasvattaa myös tarvittavaa ylläpidon määrää. Laadukkaampi ohjelmakoodi tarkoittaa siis vähäisempää ylläpitoa.

Ylläpidon suurimmat kustannukset tulevat ohjelmakoodiin jäänneiden virheiden korjaamisesta [Hum94]. Yksittäisen virheen korjauskustannukset saattavat kasvaa jopa yli 8000$. Pariohjelmoinnin kasvattaessa koodinlaatua ja vähentämällä jopa 15% kaikista virheistä [CW00] saadaan syntymään huomattavia säästöjä. Jos virhe huomataan ohjelmakoodin tuottajien tai samassa organisaatiossa työskentelevien kesken, yksittäisen virheen korjaamiseen menee keskimäärin 10 tuntia. Toisaalta, jos virhe huomataan ohjelmaa käyttävän asiakkaan toimesta, sen korjaamiseen käytettävä aika moninkertaistuu jopa 88 tuntiin [Hum94].

Jatkokehitys on vaihe, jossa jo olemassa olevaan ohjelmaan halutaan lisää ominaisuuksia tai parantaa olemassa olevia ominaisuuksia. Pariohjelmoidessa tuotettavat ratkaisut ovat laadultaan parempia ja rivimäärältään pienempiä kuin yksinohjelmoidessa [CW00]. Tämä johtaa siihen, että ohjelmaan on helpompi lisätä uusia ominaisuuksia, koska ohjelmakoodissa valmiina olevia virheitä on vähemmän ja se on luettavampaa.

6 Tulokset

Oletetaan, että tehtävänä on toteuttaa ohjelma jonka ohjelmakoodin rivimäärä on 50000 riviä yksinohjelmoidessa. Tämän lisäksi tehdään oletus, että ohjelmakoodia syntyy noin 50 riviä tunnissa. Näin ollen yksilöohjelmointia hyväksikäyttäen ohjelman kirjoittamiseen menisi 1000 tuntia. Pariohjelmointi saattaa tuottaa jopa 120-150% enemmän henkilötyötunteja, jos otamme tästä karkean keskiarvon eli 135% saadaan pariohjelmoinnilla kulutettua 1350 tuntia ohjelman kirjoittamiseen.

Ylläpitovaiheessa käytetään suurin osa ajasta virheiden korjaamiseen [CW00]. Oletamme, että virhe on huomattu ohjelman kirjoittaneessa organisaatiossa, joten yksittäisen virheen korjaamiseen menevä aika on noin 10 tuntia. Oletamme myös, että ohjelmoijat tekevät jokaista 100 ohjelmakoodiriviä kohden 10 virhettä ja näistä virheistä 70% voidaan karsia käyttämällä hyvää ohjelmistokehitysmenetelmää ja testaamalla ohjelmakoodia [CW00], eli jäljelle jää 1500 virhettä. Ohjelman kaikkien virheiden korjaamiseen menee tällöin 15000 henkilötyötuntia, jos ohjelma on toteutettu käyttäen yksilöohjelmointia. Pariohjelmointia käyttämällä voidaan välttää jopa 15% kaikista virheistä, joita ohjelmakoodiin on jäänyt, eli jäljelle jää 1275 virhettä, jos ohjelmakoodi on kirjoitettu käyttäen pariohjelmointia. Näin ollen pariohjelmoinnilla tuotetun ohjelmakoodin ylläpito tulisi viemään 12750 henkilötyötuntia.

$X_{impl} = 1000$

$X_{maint} = 15000$

$Y_{impl} = 1350$

$Y_{maint} = 12750$

$X_{total} = 1000 + 15000 = 16000$

$Y_{total} = 1350 + 12750 = 14100$

$X_{total} - Y_{total} = 1900$

Näin laskemalla voimme todeta, että näillä oletetuilla luvuilla voimme säästää 1900 henkilötyötuntia, jos jokainen virhe ohjelmasta korjataan. Tuolkseksi voimme todeta $Y_{total} < X_{total}$ pitää paikkansa. Pariohjelmoinnilla toteutetun ohjelman laatu paranee tarpeeksi, että ylläpitoon vaadittavat henkilötyötunnit pienenevät enemmän kuin mitä niiden kasvu on toteutusvaiheessa.

7 Yhteenveto

Pariohjelmoinnilla on todettu olevan ohjelmakoodin laatuun ja parien ongelmanratkaisukykyyn positiivisia vaikutuksia [WK00]. Ongelmat ratkeavat jopa 60 prosentissa siitä ajasta, mitä yksin ohjelmoivat henkilöt käyttävät saman ongelman ratkaisuun [HDAS09]. Virheiden määrä ohjelmissa myös pienenee jopa neljäsosan [WKCJ00]. Näiden lisäksi pariohjelmoidessa ohjelmoijalla on suurempi kynnys käydä tarkistamassa omaa henkilökohtaista sähköpostiaan tai käydä lukemassa aiheeseen liittymättömiä verkkosivuja [WK03]. Ohjelmoijat kuitenkin arvostavat toistensa aikaa, eivätkä halua tuhlata sitä omilla henkilökohtaisilla asioillaan. Näin ollen myös keskittyminen pysyy paremmin ongelmanratkaisussa tai tehtävän toteuttamisessa.

Yksinkertaisten tehtävien ratkaiseminen pariohjelmointia hyväksikäyttäen saattaa silti viedä lähes yhtäpaljon aikaa kuin yksin ohjelmoidessa [HDAS09]. Näin ollen taloudellisesti ei ole kannattavaa käyttää pariohjelmointia tämän tyyppisten ongelmien ratkaisemiseen. Yksinkertaisia tehtäviä ratkaistaessa virheiden määrät ovat hyvin lähellä tosiaan, kirjoittettiin ohjelmakoodi pariohjelmointia hyväksi käyttäen tai ei.

Vähänkään haastavampia ongelmia ratkaistaessa pariohjelmointi käyttää enemmän henkilötyötunteja toteutukseen. Ohjelmakoodin laatu, virheettö-

myys, rivimäärän vähäisempi määrä ja jatkuva katselmointi vähentävät kuitenkin muissa vaiheissa tarvittavia henkilötyötunteja sen verran paljon, että pariohjelmointia ei kannata jättää huomioimatta vaihtoehtona taloudellisesta näkökulmasta.

Lähteet

[ARS08] Airo, Juha Pekka, Rantanen, Jarkko ja Salmela, Timo: *Oma ura, paras ura.* Hämeenlinna: Kariston Kirjapaino Oy, 2008.

[Bec00] Beck, Kent: *Extreme programming explained. 2000.* Addison-Wesley, 2000.

[CH07] Chong, Jan ja Hurlbutt, Tom: *The social dynamics of pair programming.* Teoksessa *Software Engineering, 2007. ICSE 2007. 29th International Conference on*, sivut 354–363. IEEE, 2007.

[CW00] Cockburn, Alistair ja Williams, Laurie: *The costs and benefits of pair programming.* Extreme programming examined, sivut 223–247, 2000.

[Fag01] Fagan, Michael E: *Design and code inspections to reduce errors in program development.* Teoksessa *Pioneers and Their Contributions to Software Engineering*, sivut 301–334. Springer, 2001.

[GGF93] Gilb, Tom, Graham, Dorothy ja Finzi, Susannah: *Software inspection.* Addison-Wesley Longman Publishing Co., Inc., 1993.

[HA05] Hulkko, H. ja Abrahamsson, P.: *A multiple case study on the impact of pair programming on product quality.* Teoksessa *Proceedings - 27th International Conference on Software Engineering, ICSE05*, sivut 495–504, 2005.

[HDAS09] Hannay, J. E., Dybå, T., Arisholm, E. ja Sjøberg, D. I. K.: *The effectiveness of pair programming: A meta-analysis.* Information and Software Technology, 51(7):1110–1122, 2009.

[HM95] Haikala, Ilkka ja Märijärvi, Jukka: *Ohjelmistotuotanto.* Suomen ATK-kustannus, 1995.

[Hum94] Humphrey, Watts S: *Disciplined Software Engineering.".* Lecture Notes, Software Engineering Institute, Camegie Mellon University, 1994.

[Joh94] Johnson, Philip M: *An instrumented approach to improving software quality through formal technical review.* Teoksessa *Proceedings of the 16th international conference on Software engineering*, sivut 113–122. IEEE Computer Society Press, 1994.

[LD87] Lister, Timothy ja DeMarco, Tom: *Peopleware: productive projects and teams*, 1987.

[Mar03] Martin, Robert Cecil: *Agile software development: principles, patterns, and practices.* Prentice Hall PTR, 2003.

[NW01] Nawrocki, Jerzy ja Wojciechowski, Adam: *Experimental evaluation of pair programming.* European Software Control and Metrics (Escom), sivut 99–101, 2001.

[PM03] Padberg, Frank ja Muller, Matthias M: *Analyzing the cost and benefit of pair programming.* Teoksessa *Software Metrics Symposium, 2003. Proceedings. Ninth International*, sivut 166–177. IEEE, 2003.

[PO95] Pearse, Troy ja Oman, Paul: *Maintainability measurements on industrial source code maintenance activities.* Teoksessa *Software*

Maintenance, 1995. Proceedings., International Conference on, sivut 295–303. IEEE, 1995.

[SK98] Sommerville, Ian ja Kotonya, Gerald: *Requirements engineering: processes and techniques*. John Wiley & Sons, Inc., 1998.

[Voa01] Voas, Jeffrey: *Faster, better and cheaper*. Software, IEEE, 18(3):96–97, 2001.

[Wil96] Williams, Laurie: *Pair programming*. Encyclopedia of Software Engineering, 2, 1996.

[Wil01] Williams, L.: *Integrating pair programming into a software development process*. Teoksessa *Software Engineering Education and Training, 2001. Proceedings. 14th Conference on*, sivut 27–36, 2001.

[WK00] Williams, Laurie A. ja Kessler, Robert R.: *All I really need to know about pair programming I learned in kindergarten*. Commun. ACM, 43(5):108–114, toukokuu 2000, ISSN 0001-0782. `http://doi.acm.org/10.1145/332833.332848`.

[WK03] Williams, Laurie ja Kessler, Robert R: *Pair programming illuminated*. Addison-Wesley Professional, 2003.

[WKCJ00] Williams, Laurie, Kessler, Robert R, Cunningham, Ward ja Jeffries, Ron: *Strengthening the case for pair programming*. Software, IEEE, 17(4):19–25, 2000.

[WU01] Williams, Laurie ja Upchurch, Richard L: *In support of student pair-programming*. Teoksessa *ACM SIGCSE Bulletin*, nide 33, sivut 327–331. ACM, 2001.

[WWY+02] Williams, Laurie, Wiebe, Eric, Yang, Kai, Ferzli, Miriam ja Miller, Carol: *In support of pair programming in the introductory computer science course.* Computer Science Education, 12(3):197–212, 2002.