

Pariohjelmoinnin taloudelliset hyödyt

Ville Knuuttila

Kandidaatintutkielma

HELSINGIN YLIOPISTO

Tietojenkäsittelytieteen laitos

Helsinki, 5. toukokuuta 2013

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	5. toukokuuta 2013	12	
Tiivistelmä — Referat — Abstract			
<p>Tutkielmassa tutustutaan pariohjelmoinnin taloudellisiin hyötyihin. Pariohjelmoinnilla tiedetään olevan positiivinen vaikutus ohjelmakoodin laatuun, 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.</p>			
Avainsanat — Nyckelord — Keywords			
pariohjelmointi, taloudellisuus			
Säilytyspaikka — Förvaringsställe — Where deposited			
Muita tietoja — Övriga uppgifter — Additional information			

Sisältö

1	Johdanto	3
2	Pariohjelmointi	4
2.1	Pariohjelmoinnin historia	4
2.2	Pariohjelmoinnin prosessi	5
2.3	Pariohjelmoinnin hyödyt ja haitat	6
3	Taloudellisuus ohjelmistotuotannossa	7
4	Kysymys	8
5	Case	9
6	Tulokset	9
	Lähteet	10

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 tuhlaa sitä omilla henkilökohtaisilla asioillaan. Näin ollen myös keskittymisen 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 silloin kun siitä on hyötyä muuten, esim. jos ongelma on todella haastava, tai yrityksellä on uusi työntekijä joka pitäisi totuttaa uuten 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 tyypillisenä 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 molempi suuntaisena. Tarkkailijan pitää kyetä kertomaan omat mielipiteensä kokeneemalle kontrolloijalle.

Nopeimman ongelmaratkaisukyvyyn 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 kauheasti 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 ohjelmoivala

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 toteutusvaiheella, joka koostuu neljästä vaiheesta: suunnittelu, toteutus, testaus ja käyttöönotto. Ylläpitovaihe seuraa aina toteutusta. Kun ohjelma on otettu käyttöön, niin siirytään 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 sinänsä eroa mitenkään muuten toteutuksesta, 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.

Luettavampi ja parempilaatuisempi ohjelmakoodi johtaa siihen, että ohjelman jatkokehittäminen on helpompaa [Joh94]. Näin ollen, jos ohjelmaa jatkokehitetään niin 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.

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.

$$X_{total} = Yksilöohjelmoinninhenkilötyötunnit$$

$$X_{impl} = Yksilöohjelmoinnintoteutusvaiheenhenkilötyötunnit$$

$$X_{maint} = Yksilöohjelmoinninylläpitovaiheenhenkilötyötunnit$$

$$Y_{total} = \text{Pariohjelmoinnin henkilötötunnit}$$

$$Y_{impl} = \text{Pariohjelmoinnin toteutusvaiheen henkilötötunnit}$$

$$Y_{maint} = \text{Pariohjelmoinnin ylläpitovaiheen henkilötötunnit}$$

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

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

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

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

5 Case

Tutkimus	Impl mh	Main mh	Koodikanta	context	tavat
[HA05]	x %	y %	50k	organization wide	pro pro vaikea ongelma
[PM03]	1,4 %	0,7 %	0	-	keskimäärin

6 Tulokset

Oliko tot2 < tot1

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.
- [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.
- [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.