TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikka
Ohjelmistotekniikka

Tutkintotyö

Sami Männistö

Extreme Programming (XP) -metodologian soveltaminen ohjelmistojen tuotantoon

Työn ohjaaja Lehtori Tony Torp, TAMK

Työn teettäjä Digia Oyj, valvojana Heikki Lintinen

Tampere 2008

TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikka
Ohjelmistotekniikka

Männistö Sami

Extreme Programming (XP) -metodologian soveltaminen ohjelmistojen tuotantoon
Tutkintotyö
41 sivua
Työn-ohjaaja
Lehtori, Tony Torp, TAMK
Työn teettäjä
Digia Oyj, valvojana Heikki Lintinen
Syyskuu 2008
Hakusanat
ketterät menetelmät, pariohjelmointi, extreme programming, XP

Tiivistelmä

Extreme programming (XP) on niin sanottu ketterä ohjelmistokehitysmetodologia. Se sopii erityisesti riskialttiille projekteille, joissa vaatimukset saattavat muuttua. XP on kevyt metodologia, joka yhdistää olemassa olevien ohjelmistokehitysprosessien käytäntöjä. Tällöin kehittäjien ei tarvitse laatiakattavia dokumentaatioita, vaan voivat keskittyä olennaiseen: laadukkaan koodin kirjoittamiseen.

Tutkintotyön tarkoituksena on tutustua XP -metodologiaan, jonka suosio teollisuudessa on kasvanut viime vuosina. Työssä tutkitaan tämän metodologian eri käytäntöjä, tutkimuksissa saatuja tuloksia ja niiden soveltamista ohjelmistojen tuotannossa. Tässä työssä järjestetään käytännön kokeilu, joka painottuu XP:n pariohjelmointimenetelmään ja sen ominaisuuksien hyödyntämiseen ohjelmistojen tuotannossa.

Työn tavoitteena on antaa mahdollisimman kattava selvitys XP -metodologiasta ja sen menetelmistä. Työssä pyritään laajentamaan kirjoittajan näkemystä ja osaamista ketterien menetelmien saralla.

Tutkintotyö sopii hyvin kaikille ketteristä menetelmistä kiinnostuneille tahoille, ja varsinkin XP -metodologiaan suuntautuneille ohjelmistokehittäjille.

TAMPERE UNIVERSITY OF APPLIED SCIENCES
Computer science
Software engineer

Männistö Sami	Appliance of Extreme Programming (XP) -method to software production
Thesis	41 pages
Thesis-Supervisor	Senior Lecturer Tony Torp, Tampere Polytechnic University
Thesis Instructor	Digia Plc, Supervisor Heikki Lintinen
September 2008	
Keywords	Agile software development, pair programming, extreme programming, XP

Abstract

Extreme programming (XP) is so called agile software development method. It's good especially to project with high risk and where demands can change constantly. XP is light weight methodology which combines existing practices from software development projects. Developers are released from unnecessary work etc. large documentations and can focus on important things: writing good quality code.

Purpose of this thesis is to get know XP-methodology, whose popularity in industry has been increased in last year's. Thesis will investigate this methodology's different practices and results from research and how to adapt them in software development. Hands-on test will be in this thesis, which will focus on XP's pair programming practice and how to use its features in software development.

Goal of work is to give as extensive explanation of XP-methodology and its practices as possible. This work will pursue to increase writers point of view and know-how in the area of agile software development.

This thesis is suitable for everyone who is interested in agile methods, and especially those who are interested in XP methodology in software development

Alkusanat

Tämä tutkintotyö on tehty DIGIA Oyj:n Telekommunikaatio-yksikössä tammikuun 2008 ja syyskuun 2008 välisenä aikana. Työn ohjaajana on toiminut tiimipäällikkö Heikki Lintinen DIGIA Oyj:stä ja valvojana lehtori Tony Torp Tampereen ammattikorkeakoulusta.

Kiitokset esimiehelleni, työtovereilleni ja DIGIAlle tuesta, insinöörityön aiheesta ja resursseista, joita sain käyttööni, kun tein tutkintotyötäni.

Tampereella 25.9.2008

_____

Sami Männistö

TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikka
Ohjelmistotekniikka

SISÄLLYSLUETTELO

Lyhenteet ja termit

Extreme Programming	(XP) Ketterä ohjelmistokehitysmenetelmä.
Refaktorointi	Prosessi, jossa tietokoneohjelman lähdekoodia muutetaan siten, että sen toiminnallisuus säilyy.
Iteratiivisyys	Prosessi etenee vaiheittain eteenpäin. Yhden vaiheen tuotoksena on inkrementti
Inkrementti	Kooste viimeisimmän ja sitä edeltävien vaiheiden tuotoksista
QT	Trolltech yhtiön valmistama tietokoeohjelmien kehitykseen tarkoitettu graafinen käyttöliittymäkirjasto.
MAEMO	Linux-pohjainen kehitysalusta kämmenmikroille.
RSS	joukko verkkosyötemuotoja, joita käytetään usein päivittyvän digitaalisen sisällön julkaisemiseen.

1 Johdanto

Monissa ohjelmistoprojekteissa perinteiset ohjelmistonkehitysprosessimallit on havaittu liian jäykiksi ja kykenemättömiksi vastaamaan asiakkaan vaatimuksia. Tällaisen mallin mukaan etenevän projektin alussa asiakkaan kanssa on tehty valmiit suunnitelmat siitä, millainen kehitettävän sovelluksen tulisi olla. Toteutusvaihe voi kestää muutamasta kuukausista jopa vuosiin, ja sitten kun asiakkaalle esitellään valmista tuotosta, se ei välttämättä vastaa asiakkaan toiveita. Voi olla että joku näkökulma on jäänyt projektin alussa kokonaan huomioimatta tai se on ymmärretty väärin, jolloin alusta alkaen on alettu kehittämään vääränlaista järjestelmää. Ratkaisuksi tällaisiin ongelmiin on kehitetty joustavampia prosessimalleja.

Ketterät menetelmät pyrkivät tuomaan kaivattua joustavuutta ohjelmistokehitykseen. Tavoitteena on tuottaa asiakkaalle toimivia versioita kehitettävästä ohjelmistosta säännöllisin väliajoin, hallita muuttuvia vaatimuksia myös myöhäisessä projektin vaiheessa sekä olla yhteistyössä asiakkaan kanssa.

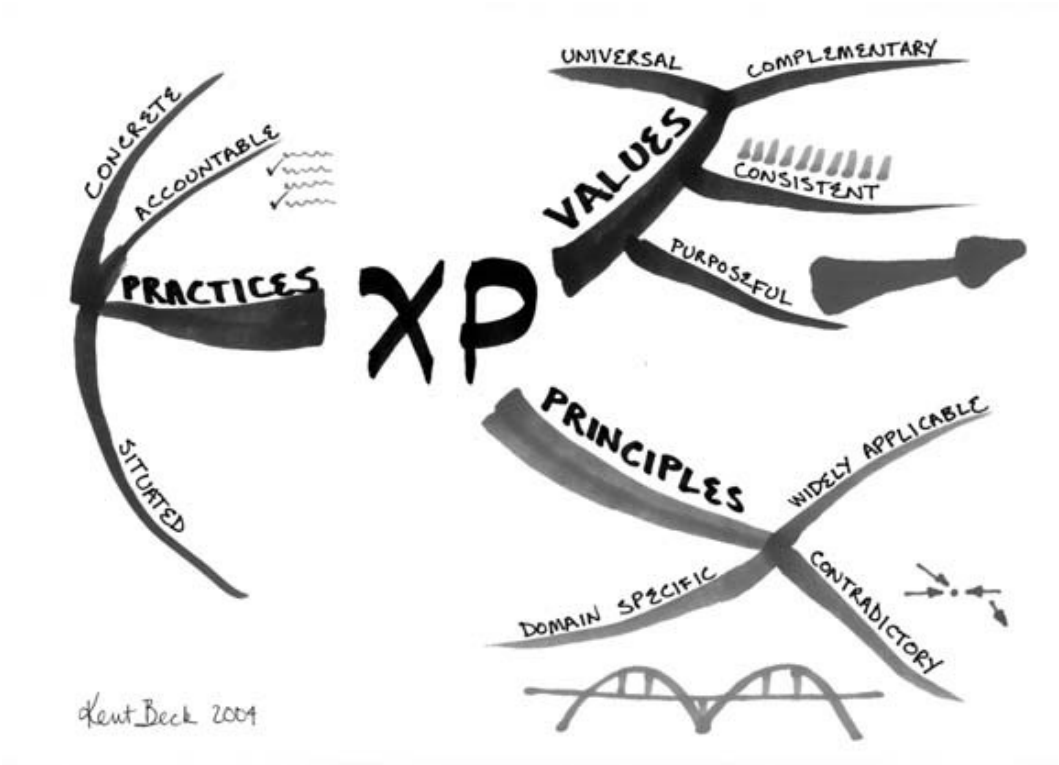
Ketterät menetelmät, kuten tässä tutkintotyössä käsitelty XP, ovat kasvattaneet teollisuudessa suosiotaan viime vuosina. Kasvava määrä ohjelmistojen kehittäjätiimejä käyttää työelämässä menestyksellisesti näitä menetelmiä.

Työssä esitellään XP ja tutkitaan pariohjelmoinnin soveltumista ohjelmistojen kehitykseen. Tavoitteena on lisätä omaa tietämystä ketteristä menetelmistä.

2 Mikä on XP?

Extreme programming on harkittu ja kurinalainen ohjelmistokehityksen näkökulma, joka pohjautuu seuraaviin arvoihin: yksinkertaisuus, kommunikointi, palaute, rohkeus ja kunnioitus.Se on luopumista vuosien aikana opituista vanhoista kaavoista, jotka ovat nyt ohjelmistokehityksen tiellä. XP on menestyvä, koska se korostaa asiakastyytyväisyyttä. Metodologia on suunniteltu toimittamaan ohjelma asiakkaalle, kun asiakas sitä tarvitsee.

XP-Metodologia korostaa ryhmätyöskentelyä. Johtajat, asiakkaat ja kehittäjät ovat kaikki osa työryhmää, jotka ovat sitoutuneet luovuttamaan laadukasta ohjelmistoa. XP-ohjelmoijat kommunikoivat asiakkaiden ja toisten ohjelmoijien kanssa. Ohjelmoijat pitävät heidän suunnitelmansa yksinkertaisina. He saavat palautetta testaamalla ohjelmaa ensimmäisestä päivästä alkaen ja toimittavat ohjelmiston asiakkaalle niin pian kuin mahdollista. Tämä menetelmä sopii erinomaisesti pieniin alle 20 henkilön projekteihin. Kuvassa 2.1 yleiskuva arvojen, käytäntöjen ja tapojen vuorovaikutuksesta XP-metodologiassa.



KUVA 2.1 XP:n perusidea /1/

Mikä tekee XP:stä ”extremen”?

Perinteisessä ohjelmistokehityksessä käytössä olevet käytännöt ja periaatteet viedään XP:ssä äärimmäiselle tasolle:

- Jos koodin katselmointi on hyvää, sitä katselmoidaan jatkuvasti
- Jos suunnittelu on hyvä, se on jokaisen päivittäistä työtä
- Jos yksinkertaisuus on hyvä, teemme aina yksinkertaisimman toimivan ratkaisun
- Jos rakenne on tärkeä, jokainen työskentelee määritellen ja jalostaen rakennetta jatkuvasti
- Jos testaus on hyvä, teemme sitä jatkuvasti
- Jos lyhyet iteraatiot on hyvä asia, pyrimme saavuttamaan todella lyhyet iteraatiot sekunteja, minuutteja ja tunteja ei viikkoja, kuukausia tai vuosia.

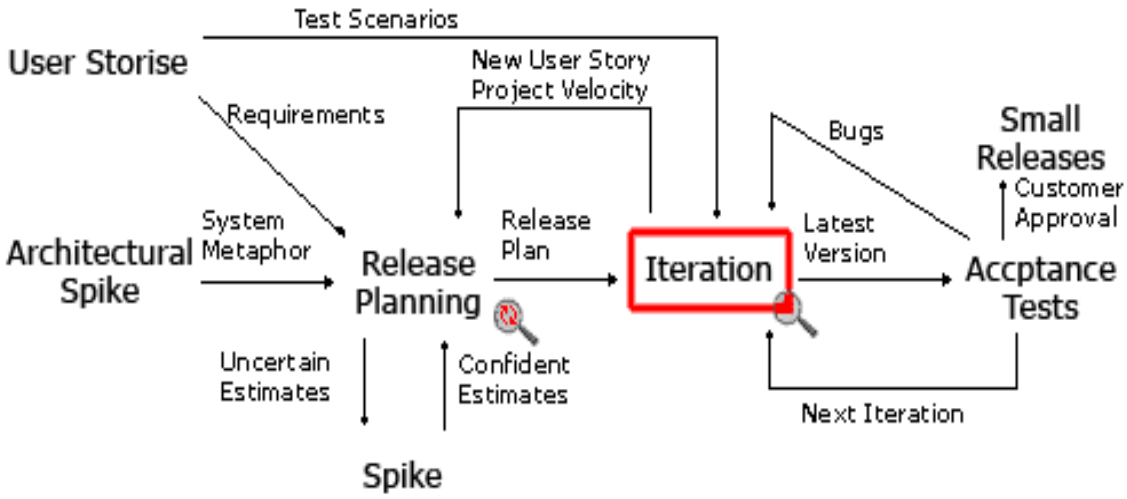
XP antaa kaksi sarjaa lupauksia.

2.1.1 Ohjelmoijalle

Ohjelmoijat saavat työskennellä joka päivä sen parissa, millä on todella merkitystä. Pelottavia tilanteitä ei tarvitse kohdata yksin. He saavat tehdä kaiken mahdollisen saadakseen toimivan kokonaisuuden. Ohjelmoijat saavat päättää vain niistä asioista jotka he osaavat parhaiten.

2.1.2 Asiakkaalle ja johtajalle

Asiakkaat ja johtajat saavat parhaan mahdollisen arvon jokaiselle ohjelmointiviikolle. Joka viikko he voivat nähdä konkreettista edistymistä heitä kiinnostavista kohteista. He voivat muuttaa kurssia kesken projektin kehityksen ilman kohtuutonta maksua. Kuva 2.2 kuvaa XP-projektin kulkua.



Kuva 2.2 XP:n projektin kulku /2/

3 XP:n arvot ja periaatteet

3.1 Arvot

Jokainen ohjelmistotuotannon parissa työskentelevä tietää, mikä on tärkeää. Yksi henkilö voi ajatella, että perinpohjainen suunnitteleminen on tärkeää ennen ohjelmointia. Toinen voi ajatella, että todella merkitsevää on oma henkilökohtainen vapaus, jolle ei ole mitään estoja. Näiden sijaan tulisi keskittyä XP:n arvoihin, jotka ovat ensisijainen apukeino, kun ei ole varmaa, tilanne selvitetään.
Nämä arvot ovat seuraavat:

- yksinkertaisuus
- kommunikointi
- palaute
- rohkeus
- kunnioitus.

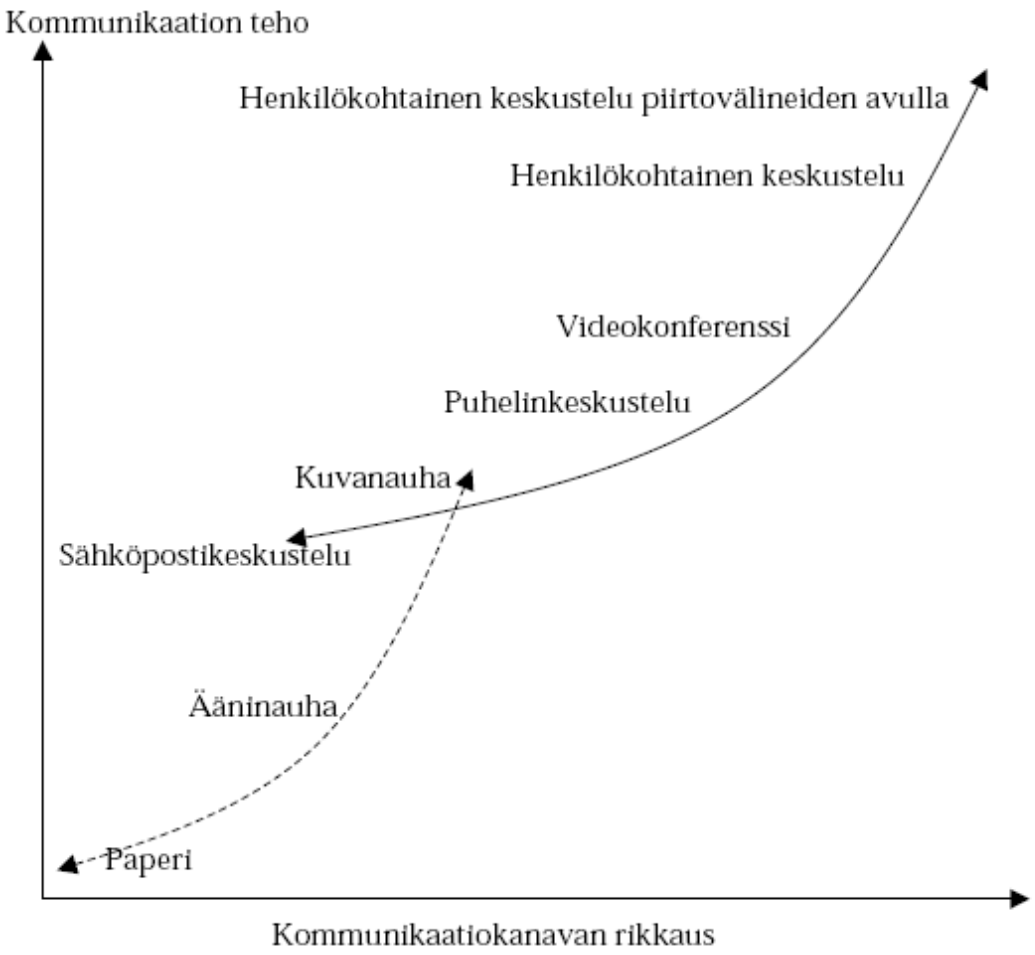
3.1.1 Yksinkertaisuus

Aluksi etsitään yksinkertaisin vaihtoehto työn toteuttamiseen ja parannellaan sitä myöhemmin, jos tarve vaatii. Tämä arvo perustuu siihen, että ohjelmiston tarpeet ja moni muu asia muuttuu usein ja ne ovat niin epätarkkoja, että on harvoin kannattavaa lisätä ominaisuutta, joka voisi olla hyödyllinen jossain tulevaisuuden vaiheessa. Tätä kuvaa niin sanottu YAGNI-periaate: ellet tarvitse tiettyä asiaa nyt, et luultavasti tule tarvitsemaan sitä myöhemminkään.

"XP is making a bet. It is betting that it is better to do a simple thing today and pay a little more tomorrow to change it if it needs it, than to do a more complicated thing today that may never be used anyway." /4/

3.1.2 Kommunikointi

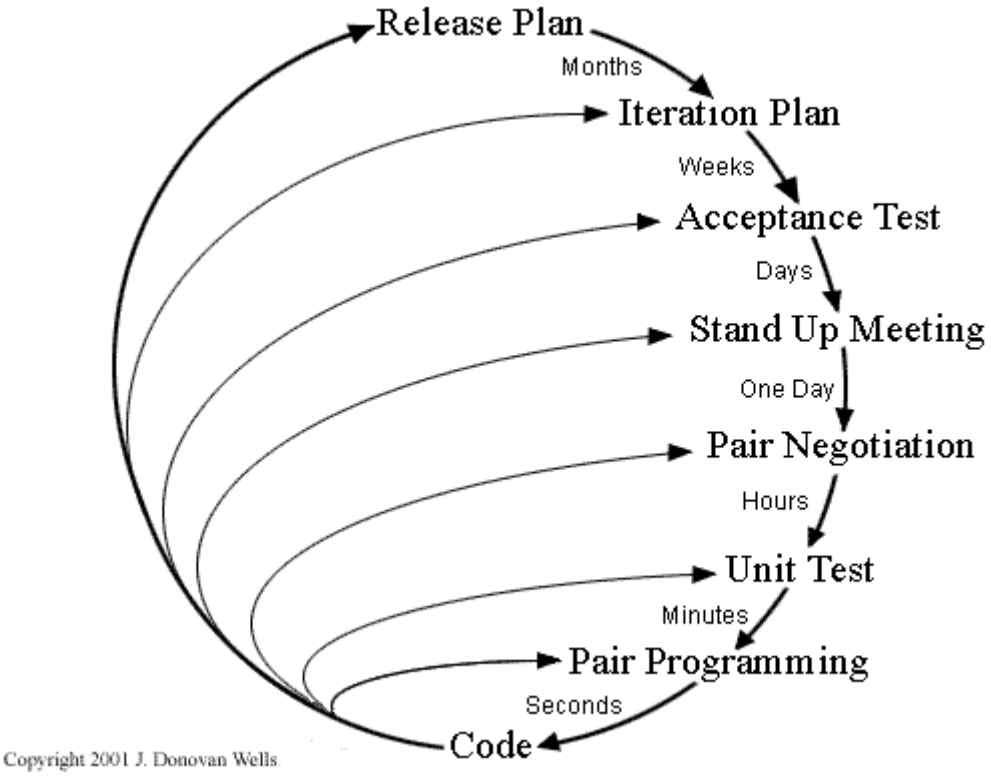
Kommunikoinnin tärkeys on nimenomaan siinä, että työryhmät ajattelevat kaiken tiedon jakamisen olevan erittäin tärkeää. XP:n työryhmät kuitenkin ajattelevat, että ylikommunikointi on parempi vaihtoehto kuin kommunikoinnin puute. XP-työryhmät jakavat tietoa monella eri tasolla: Pariohjelmointia nopeaan tiedon vaihtoon, sekä päivittäisiä tapaamisia koko työryhmän tietojen päivittämiseen. Kuva 3.1 kuvaa sitä, kuinka tehokasta kommunikaatio on eri välineitä käytettäessä. Kuvaajassa esitetään katkoviivalla erilaisia dokumentaatiovälineitä ja yhtenäisellä viivalla erilaisia välineitä ohjelmiston mallintamiseen. Paperidokumentaatio nähdään heikoimmaksi dokumentaatiovälineeksi, vaikka sitä ohjelmistoprojekteissa on perinteisesti tehty runsaasti.



Kuva 3.1 Erilaisten kommunikaatiomallien tehokkuus /3/

3.1.3 Palaute

Palautteen arvo lisää uskoa siihen, että ohjelmistojen tilaajien tarpeet muuttuvat aina tai ei ole ymmärretty hyvin projektin alussa. Sen takia ainoa tapa luoda ohjelmistoa, jota asiakas todella tarvitsee, on jatkuvasti muuttaa kehitystä asiakkaan palautteen perusteella. Palaute tulee monessa eri aikaskaalassa. Ensimmäinen palaute toimii minuutti- ja päiväskaalassa. Ohjelmoija kirjoittaa testin ja ajaa sen. Ohjelmoija saa testin palautteen minuuteissa käsiteltäväkseen. Palautetta tulee myös työkavereilta, kokouksista ja esimiehiltä päivittäin. Viikko- ja kuukausiskaalassa palaute tulee asiakkaalta ja testaajalta, jotka testaavat kaikki ohjelmat. Kuva 3.2 eri käytäntöjen nopeus palautteen antamisessa.



KUVA 3.2 Palaute eri ajanjaksoissa. /5/

"The business people say "I had no idea that was so expensive. Just do this one third of it. That will do fine for now." /4/

3.1.4 Rohkeus

Jonkin asian tekeminen seurauksista välittämättä ei ole ryhmätyöskentelyä. Rohkaise ryhmätyöskentelyyn etsimällä muista arvoista apua tilanteisiin, joissa olet peloissasi. Vaikeat päätökset on syytä tehdä heti. Myös epäonnistuneen koodin voi hylätä, sillä on parempi alkaa tehdä alusta kuin yrittää parantaa asiaa joka ei toimi.

3.1.5 Kunnioitus

Kunnioitus on XP:ssä uusi arvo, joka tuli esiin Extreme programming explained second edition -kirjassa. Jokaista tiimin jäsentä tulee kunnioittaa. Olen tärkeä ja niin olet sinäkin.

Periaatteet

Pariaatteet, jotka muodostavat XP:n perustan pohjautuvat yllä mainittuihin arvoihin. Periaatteiden on tarkoitus olla konkreettisempia kuin arvojen ja paremmin ymmärrettäviä käytännön tilanteessa.

Perusperiaatteet ovat:

- nopea palaute
- pyrkimys yksinkertaisuuteen
- inkrementaalinen muutos
- muutosten hyväksyminen
- laadukas työ.

Toissijaiset periaatteet:

- pienet alkuperäiset sijoitukset
- konkreettiset kokeet
- Vastuun ottaminen
- oppimaan opettaminen
- avoin, rehellinen kommunikointi
- paikalliset sovellukset
- rehellinen mittaaminen.

3.1.6 Nopea palaute

Palaute on parhaimmillaan, kun se tehdään heti. Toiminnan ja palautteen välisellä ajalla on olennainen merkitys oppimiselle. Ohjelmistotuotanto tehdään pienissä iteraatioissa mikä: mahdollistaa asiakkaan palautteen nopeasti.

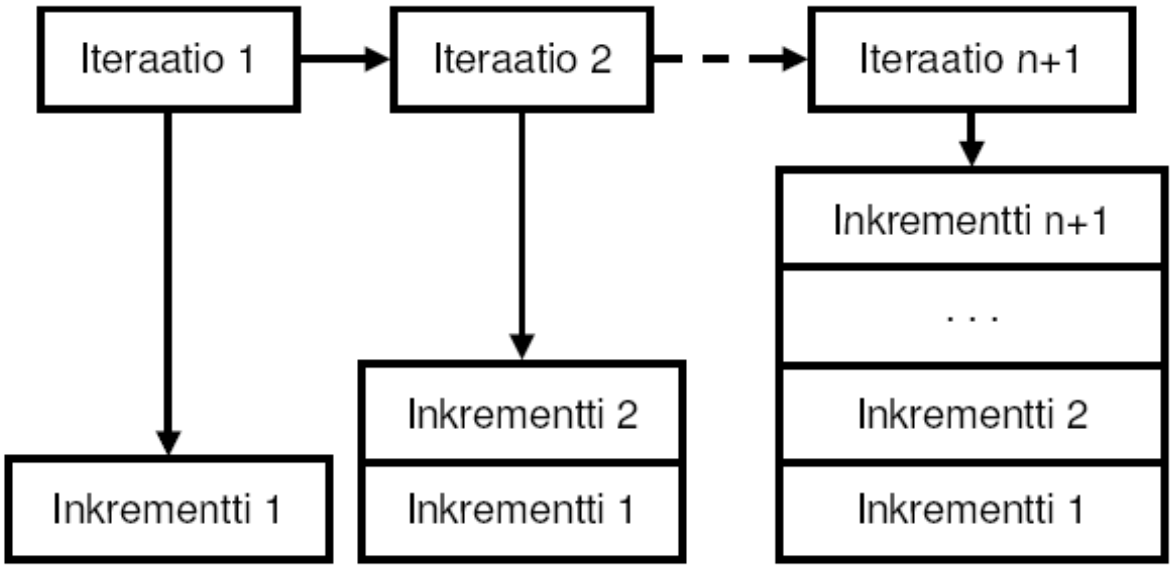
3.1.7 Pyrkimys yksinkertaisuuteen

Kohdataan jokainen ongelma kuin se olisi erittäin yksinkertainen. Tehdään pieniä ja yksinkertaisia ratkaisuvaihtoehtoja ja parannetaan niitä tulevaisuudessa, jos on tarvetta. Pienten muutosten seuraaminen ja korjaaminen on huomattavasti yksinkertaisempaa kuin suurten muutosten hallinta.

3.1.8 Inkrementaalinen muutos

Tehdään pieniä yksittäisiä muutoksia ja jokaisen muutoksen jälkeen tarkistetaan, onko se aikaisempaa parempi. Näin menetellen löydetään yleensä vanhaa ratkaisua paremman mallin, mutta ei välttämättä parasta vaihtoehtoa.

XP:n prosessi on iteratiivinen ja inkrementaalinen kuva 5.2. Iteratiivisyys tarkoittaa, että prosessi etenee vaiheittain eteenpäin. Yhden vaiheen tuotoksena on inkrementti, joka koostuu viimeisimmän ja sitä edeltävien vaiheiden tuotoksista.



KUVA 5.2 XP on iteratiivinen ja inkrementaalinen

3.1.9 Muutosten hyväksyminen

Otetaan muutokset vastaan ja mukaudutaan muutosten tarpeisiin. Nopeiden iteraatioiden projekteissa muutoksia tulee asiakkaalta pienissä mittakaavoissa päivittäin ja suuremmissa, jopa viikoittain.

3.1.10Laadukas työ

Muutokset toteutetaan huolellisesti ja kokonaan. Edistyksen pitää jatkuvasti näkyä myös toimivissa tuotteissa.

4 XP:n säännöt

XP:n käytännöt eivät vielä määrittele XP:tä. Ne vertaavat ohjelmistokehitystä jalkapallon pelaamiseen, ja samalla tavalla kuin jalkapallo on määritelty sen sääntöjen avulla, myös XP tulisi määritellä sen sääntöjen avulla.

Suunnittelu

- Käyttäjäkertomukset on kirjoitettu
- julkaisu suunnittelu luo aikataulut
- Useita pieniä julkaisuja
- Projektin tehtyä työmäärää mitataan
- Projekti on jaettu iteraatioihin
- Iteraation suunnittelu aloittaa jokaisen iteraation
- Henkilöt vaihtavat usein tehtäviä
- Joka aamu pikakokous seisten
- Korjaa XP kun se hajoaa.

Suunnitelma

- Yksinkertaisuus
- Yhteinen kielikuva
- Tee pika ratkaisuja riskien vähentämiseksi
- Ei toiminnallisuuksia aikaisin
- Refaktoroi aina kun mahdollista.

Koodaus

- Asiakas on aina tavoitettavissa
- Koodi kirjoitetaan ennalta määrättyjen standardien perusteella
- Koodaa testi ensin
- Kaikki koodi on pariohjelmoitua
- Vain yksi pari päivittää koodia
- Päivitä usein
- Yhteinen koodin omistaminen
- Jätä optimointi viimeiseksi
- Ei ylitöitä.

Testaus

- Kaikella koodilla pitää olla testi
- Kaiken koodin pitää läpäistä testi ennen kuin se julkaistaan
- Kun virhe löytyy tehdään sille testi
- Hyväksymistestaus ajetaan usein ja tulokset julkistetaan.

5 XP:n käytännöt

XP:llä on kaksitoista käytäntöä, jotka on jaettu neljään ryhmään: hieno palaute, jatkuva prosessi, jaettu ymmärtäminen ja ohjelmoijan hyvinvointi.

Hieno palaute

5.1.1 Pariohjelmointi

Pariohjelmointi tarkoittaa, että kaksi henkilöä ohjelmoi yhdellä tietokoneella. Toinen hallitsee työaseman ja toinen istuu vieressä ja pääasiassa ajattelee ohjelmointia. Hänellä on laaja käsitys projektista, mutta hän vain katselmoi toisen tuottamaa koodia. Henkilöt keskustelevat jatkuvasti tuottamastaan koodista, ratkaisuista ja ongelmista. He myös vaihtavat rooleja säännöllisesti. Parin kanssa Kommunikoinnin tulisi olla helppoa ja vapaata.

5.1.2 Suunnittelupeli

Pääsuunnitteluprosessia XP:ssä kutsutaan nimellä ”Planning game”. Peli toteutuu kerran iteraatiossa. Se on jaettu kahteen osaan releasioinnin suunnittelu ja iteraation suunnittelu. Releasioinnin suunnittelu koostuu -tutkinta, sitoumus- ja Ohjausvaiheista. Tutkintavaiheessa tarkastellaan asiakkaan vaatimuksia ja arvioidaan niiden perusteella työn kesto. Sitoumusvaiheessa: Arvioidaan hintaa, hyötyä ja ajan käyttöä. Ohjaus: voidaan muuttaa ja priorisoida tehtäviä. Iteraation suunnittelu koostuu -Tutkinta, sitoumus- ja Ohjausvaiheista. Tutkintavaiheessa: tehdään tehtävät ja arvioidaan niihin kuluva aika. Sitoumus: Valitaan tehtävät, joista otetaan vastuu. Ohjausvaiheessa: otetaan tehtävä, valitaan pari (jos pariohjelmointia), suunnitellaan tehtävän toteutusta, kirjoitetaan testi, kirjoitetaan koodi, ajetaan testi ja korjataan koodia.

5.1.3 Testipainoitteinen kehitys

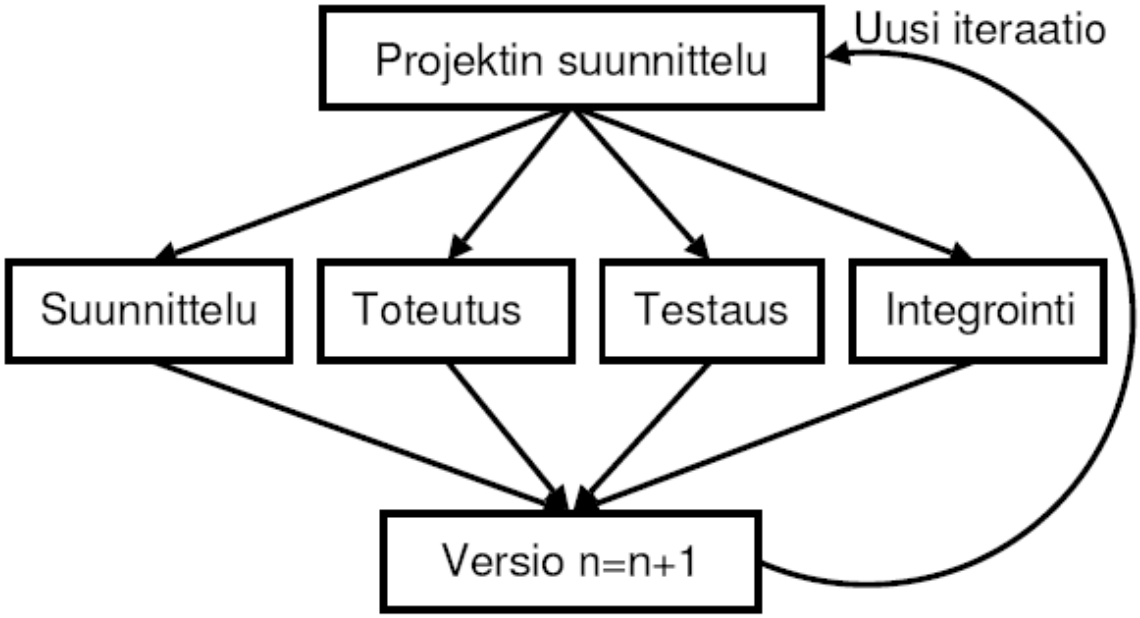
Jos testaus ajetaan käsityönä virheitä saattaa jäädä huomaamatta, aikaa menee hukkaan ja työ on tylsää, jolloin se jää tekemättä. XP:ssä testi kirjoitetaan jo ennen varsinaista koodia. Tämä tapa auttaa ohjelmoijaa ennakoimaan, miten hänen koodinsa voisi epäonnistua. Työryhmät tuottavat 100-prosenttisesti testattua koodia. Joka kerta, kun ohjelmoija julkaisee koodia, se testataan. Jokaisen testin täytyy mennä läpi. Tämä tarkoittaa, että ohjelmoija saa välitöntä palautetta ohjelman toiminnasta.

5.1.4 Koko tiimi

Kaikki projektin projektiin liittyvä väki toimii yhtenä joukkona. Asiakkaan tai asiakkaan edustajan pitäisi olla jatkuvasti saatavilla kysymyksiä varten. Kaikki työskentelevät samassa huoneessa (avoimessa työtilassa).

Jatkuva prosessi

XP:n prosessi alkaa yksinkertaisesta rakenteesta, joka täyttää alun vaatimukset ja kehittyy tasaisesti haluttuun joustavuuteen poistaen tarpeettoman monimutkaisuuden. Kuva 5.1 Pyrkimyksenä on tuottaa yksinkertaisin mahdollinen ratkaisu, joka täyttää aina sen hetkiset vaatimukset.



KUVA 5.1 XP:n prosessi

5.1.1 Käyttäjäkertomukset on kirjoitettu

Käyttäjäkertomukset korvaavat vaatimusten määrittelydokumentin. Ne muistuttavat käyttötapauksia, mutta eivät ole yhtä tarkkoja. Tarkat tiedot saadaan suoraan asiakkaalta. Käyttäjäkertomukset keskittyvät käyttäjän tarpeisiin ja hyötyihin. Käyttäjäkertomukset toimivat pohjana aikataulutukselle ja hyväksymistesteille.

5.1.2 Jatkuva integraatio

Jokaisen ohjelmistokehittäjän tulisi työskennellä ohjelman viimeisimmän version kanssa, koska eri tiimin henkilöillä voi olla tallennettu omalle koneelle vanha versio, johon on tullut uusia muutoksia. Kehittäjien tulisi päivittää koodinsa tietovarastoon muutaman tunnin välein.

5.1.3 Suunnittelun parantaminen

Jatkuva refaktoorointi pitää koodin rakenteen mahdollisimman yksinkertaisena ja lisää koodin ymmärrettävyyttä. Lisäksi jatkuva refaktorointi poistaa toiston ohjelmakoodista.

5.1.4 Pienet toimitukset

Useimmiten jokainen toimitus sisältää pienen palan toimitettavasta ohjelmasta. Pienet toimitukset joka iteraatiossa auttavat asiakasta uskomaan projektin etenemiseen.

Jaettu ymmärtäminen

5.1.5 Koodausmenetelmät

Koodausmenetelmät ovat hyväksyttyjä sääntöjä, joita koko projekti käyttää. Niissä on määritelty koodiin tyyli ja rakenne.

5.1.6 Yhteinen koodin omistaminen

Jokainen on vastuussa tuotetusta koodista. Jokaisella on myös oikeus muuttaa koodia. Jos koodin omistavat yksilöt, vaaditut ominaisuudet yleensä laitetaan väärin paikkoihin, kun yksi ohjelmoija huomaa tarvitsevansa ominaisuuden jossain koodissa, jota hän ei omista. Tavallinen virhe on, että kun opistaja on liian kiireinen, ohjelmoija lisää tarvittavan ominaisuuden koodiinsa, mihin se ei kuulu. Tämä johtaa rumaan, hankalasti ylläpidettävään koodiin, joka on täynnä kopioita.

5.1.7 Yksinkertainen rakenne

Ohjelmoijan tulisi ottaa ”yksinkertaisin on paras” -näkökulma ohjelmistokehitykseen. Ohjelmakoodia kirjoittaessaan tulisi kirjoittajan miettiä, voiko tämän tehdä vieläkin yksinkertaisemmin.

5.1.8 Yhteinen kielikuva

Yhteinen kielikuva tarkoittaa sitä, että käytetään samanlaista nimeämismenetelmää luokille ja metodeille koodauksessa. Koodin tulee näyttää siltä, kuin sen olisi ohjelmoinut yksi ja sama henkilö kokonaan.

Ohjelmoijan hyvinvointi

5.1.9 Sopiva tahti

Ohjelmistokehittäjä ei saisi tehdä enempää kuin 40 tuntia viikossa. Kun viikolla on ylitöitä, seuraavalla viikolla niitä ei saisi olla.

6 XP:n tavoite

XP:n lähtökohtana on tuottaa nopeasti näyttävää koodia. XP vaatii mm. sen, että kehitystiimillä on jatkuvasti mahdollisuus käyttää asiakasta ja tämän tietoja hyväkseen. Käytännössä asiakkaan pitäisi olla kehitystiimin vieressä istumassa, mikä ei yleensä ole mahdollista taloudellisista saati maantieteellisistäkään syistä.

Lisäksi kehitystiimiltä vaaditaan melko tasaista täyspäiväistä läsnäoloa sekä erityisen paljon yhteen hiileen puhaltamista, eikä tiimin koko saa olla kovinkaan suuri.. XP:n hyviä puolia on testien laatiminen ennen muun koodin kirjoittamista sekä pariohjelmointi. Jälkimmäisen perusteleminen johdolle voi olla erittäinkin vaikeaa, mutta omien kokemusten perusteella tulokset ovat erinomaisia:

- vähemmän virheitä koodissa
- helpommin ylläpidettävää koodia
- laaditun koodinpätkän omistus vähintään kahdella kehittäjällä perinteisemmän yhden sijaan
- nuoremman ohjelmoijan nopeampi taitojen karttuminen.

Pariohjelmoinnin huonoja puolia on henkilökemian löytäminen, jota ilman hommasta ei tule mitään, ja se, että kysymysten ja palautteen on oltava välitöntä. Jos esim. nuorempi ohjelmoija ei viitsi kysellä tai huomauttaa virheistä tarpeeksi ajoissa, ei pariohjelmoinnilla ole saavutettu käytännössä firman tai tuotteen kannalta mitään.

Lisäksi XP tähtää nimenomaan siihen, että asiakas saa mahdollisimman aikaisessa vaiheessa käyttöönsä ohjelman, jolla voi tehdä tuottavaa työtä. Kuten jo aiemmin todettiin, tuon seikan puitteissa XP sopii huomattavasti paremmin uuden ohjelmiston tuottamiseen kuin vanhan ylläpitoon. Kyllä XP:ssäkin tuotetaan määrittelyjä, mutta huomattavasti kevyemmässä muodossa kuin muissa perinteisemmissä malleissa.

7 Miksi käyttää XP:tä

XP on yksi harvoista metodologioista, jotka ajavat tuottamaan korkealaatuista koodia, joka vastaa asiakkaan toiveita. XP luottaa vahvasti asiakaskommunikointiin koko kehitys prosessin ajan. Tämä metodologia kokoaa joukon yksilöitä yhtenäiseksi ja tehokkaaksi ryhmäksi.

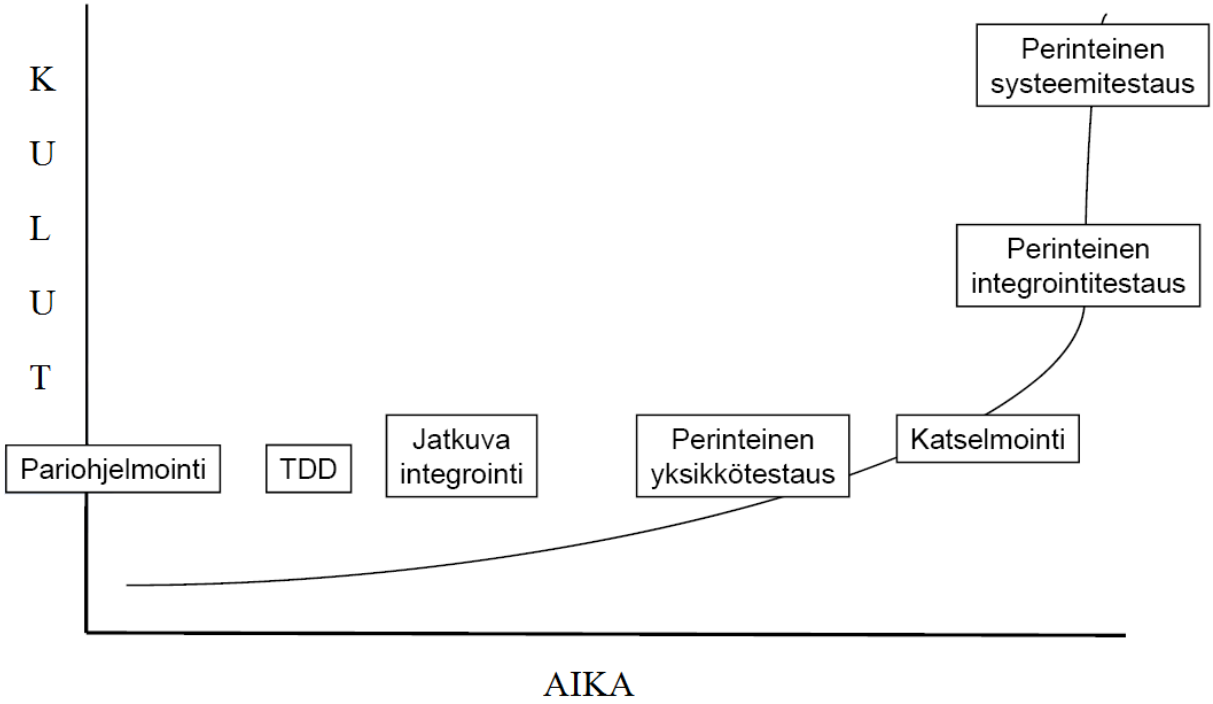
Projektin alusta-asti pidetyt päivittäiset 5 - 10 minuutin seisten tapahtuvat tapaamiset pitävät kaikki kehittäjät tietoisena projektin tavoitteesta. Nämä tapaamiset tarkoittavat myös, että jokainen kehittäjäpari tietää päivän tehtävät ja missä vaiheessa projekti on. Tällä tavalla suunnitteleminen innostaa ryhmiä tulemaan tehokkaammiksi ja suoriutumaan korkeammalla tasolla. Kun ryhmä ymmärtää, mihin se on menossa ja kuinka sinne päästään, on todennäköisempää, että tavoitteet saavutetaan.

XP on myös yksi parhaista metodologioista laadun varmistamiseen. Tämä pääasiassa, koska testit ovat elinkaaren etupäässä. Testikäsikirjoitus on kirjoitettu ennen kuin moduulit on ohjelmoitu ja hyväksymistestauksen vaaditaan olevan virheetön ennen koodin hyväksymistä. XP:n vahva vaikutus testaamiseen tarjoaa alustan, joka mahdollistaa kehittäjien rakentaa sarjan testejä, joita voidaan käyttää tulevaisuudessa. Koska järjestelmä on suunniteltu käyttämään testejä tällä tavoin, uudet kehittäjät voivat luottavaisina muuttaa koodia moduulissa ilman rikkoutumisen pelkoa.

Viimeinen hyöty XP:ssä on asiakastyytyväisyys. Tämä metodologia mahdollistaa toimivan ohjelman nopeammin asiakkaalle kuin muilla tekniikoilla. Kun tarjotaan kehittyvää järjestelmää, asiakkaalla on mahdollisuus muuttaa tärkeysjärjestystä ja toimintaa. Tämä mahdollistaa ohjelmiston kehityksen siinä vaiheessa, missä se tarjoaa suurimman hyödyn kaupankäynnin kannalta. Toisilla metodologioilla menee kauemmin toteuttamiseen ja siksi on suurempi riski hukata liiketoiminta mahdollisuuksia.

8 Pariohjelmointi

Pariohjelmointi on ohjelmoinnin tyyli, jossa kaksi ohjelmoijaa työskentelee yhdessä saman tietokoneen ääressä, jatkuvasti kehitellen suunnitelmaa, algoritmejä tai ohjelmakoodia. Suurin osa ohjelmoijista on kuitenkin jo pitkän aikaa työskennellyt yksin ja yleensä vastustavat siirtymistä pariohjelmointiin. Kuva 8.1 kuvaa koodausvirheiden havaitsemista eri tekniikoiden avulla.



KUVA 8.1 Eri menetelmien kustannukset virheiden havaitsemisissa.

Jaa kaikki

Pariohjelmoinnissa kaksi ohjelmoijaa on määrätty yhdessä kehittämään yhtä artefaktia (suunnitelma, algoritmi tai ohjelmakoodi). Kaksi ohjelmoijaa on kuin yhtenäinen organismi, joka työskentelee kuin yksi ajatus, vastuussa jokaisesta artefaktin osasta. Yksi henkilö kirjoittaa tai lukee, toisen jatkuvasti tarkistaessa työtä. Molemmat ovat yhtä osallisina prosessissa. Ei ole hyväksyttävää sanoa asioita kuten, "Sinä teit virheen suunnitelmaasi." tai "Tämä virhe oli sinun syytäsi." vaan, "Me teimme virheen suunnitelmassa." tai, vielä paremmin, "Saimme juuri testin läpi ilman virheitä!". Molemmat omistavat kaiken.

Pelaa reilusti

Pariohjelmoinnissa toinen henkilö "ajaa” (hallitsee näppäimistöä) sillä välin kun toinen tarkistaa työtä. Jopa silloin kun toinen ohjelmoija on huomattavasti kokeneempi kuin toinen, on tärkeää vuorotella ajovuoroja. Henkilö joka ei ole ajovuorossa ei ole passiivinen tarkastaja, vaan on aina aktiivinen. "Vain toisen ohjelmoimisen katseleminen on suunnilleen yhtä kiinnostavaa, kuin katsoa ruohon kuolevan aavikolla." Tarkastajan päärooli on suorittaa jatkuvaa analyysiä, suunnittelua ja tarkistamista.

Keskity olennaiseen

Pidä huoli että pari on keskittynyt ja oikeassa asiassa. Epäilemättä, pariohjelmoinnin hyöty on että kumpikaan epätodennäköisesti "haaskaa aikaa" lukemalla sähköpostia, selailemalla internetiä - koska pari odottaa jatkuvaa panosta ja palautetta. Lisäksi, kumpikin olettaa toisen seuraavan määrättyjä ohjelmointikäytäntöjä.

Pistä tavarat takaisin paikalleen

Mieli on monimutkainen asia. Jos ajattelet jotain tarpeeksi, aivot ajattelevat että se on totta. Jos kerrot itsellesi jotain negatiivista, esimerkiksi "Minä olen huono ohjelmoija." pian aivosi uskovat sinua. Tarkastellut pariohjelmoijat näyttävät, että on todella vaikeata työskennellä sellaisen henkilön kanssa, joilla on huono itsetunto ohjelmointitaidoistaan. Tällaisten ohjelmoijien tulisi katsoa pariohjelmointia menetelmänä kehittää taitojaan jatkuvasti katsellen ja palautetta saaden pariltaan. Yhdessä pari voi ratkaista ongelmia, joita he eivät saisi ratkaistua yksin ja voivat auttaa parantamaan toistensa taitoja. Kukaan meistä, ei väliä kuinka lahjakas, ei ole täydentämätön ja toisten palautteen yläpuolella.

Siivoa jälkesi

Pariohjelmointi siteeraa että on uskomatonta, kuinka moni selvä mutta huomaamaton virhe tulee löydetyksi toisen ohjelmoijan katsellessa sinun olkapääsi ylitse. Lisäksi, nämä virheet voidaan poistaa ilman luonnollista vihamielisyytä joka saattaisi kehittyä normaalissa tarkastuskokouksessa.

Älä ota asioita liian vakavasti

Pariohjelmointi tutkimuksen perusteella, liiallinen itsetunto paljastaa itsensä kahdella tapaa, molemmat vahingoittaen yhteistä suhdetta. Ensimmäisenä, On "Kuten minä sanon tai ei ollenkaan" asenne, joka voi estää ohjelmoijaa ajattelemasta muita ideoita. Toisena, liiallinen itsetunto voi johtaa ohjelmoijan puolustelemiseen saadessaan palautetta. Käänteisesti, henkilö joka aina samaa mieltä parinsa kanssa, jottei loisi liikaa jännitettä, myös minimoivat yhdessä työskentelyn hyödyt. On olemassa hieno tasapaino liian vahvan on ja liian heikon itsetunnon välillä. Tehokas pariohjelmoija hioo tämän tasapainon perehdytysvaiheessa. Ward Cunnigham, eräs XP:n löytäjistä ja kokenut pariohjelmoija, raportoi perehtymisvaiheen kestävän pari tuntia tai päivää riippuen yksilöistä, työn kuvasta ja heidän kokemuksestaan pariohjelmoinnista.

Työympäristö

Ohjelmoijien täytyy saada istua kylki kyljessä ja ohjelmoida, samaan aikaan katsellen näyttöä ja jakaen näppäimistön ja hiiren. Tehokas kommunikointi, omassa työparissa ja toisten työparien kanssa on etusijalla. Ilman vaivaa, ohjelmoijien tulee nähdä toisensa, kysyä toisiltaan kysymyksiä ja tehdä päätökset artefaktia koskevista asioista. Ohjelmoijat hyötyvät vahingossa kuulemistaan keskusteluista, joilla voi olla tärkeä osa omaa työtä.

Poista skeptisyys kun aloitat

Moni ohjelmoija menee heidän ensimmäiseen pariohjelmointi työtehtävään skeptisenä yhteistyön hyödyistä ohjelmoinnissa, odottaen että ei hyödy tai nauti kokemuksesta. Pariohjelmointi suhde voidaan aloittaa sillä, että toinen ohjelmoija kysyy toista istumaan ja antamaan hänelle apua - ja jatketaan siitä. Kun suhde saadaan kuntoon toinen voisi sanoa, "Se meni hyvin. Minulla on nyt vähän aikaa. Onko sinulla ongelmia joissa voin auttaa?" Kokemus on näyttänyt että on vain yksi ohjelmoija, positiivinen ja kokenut pariohjelmoinnissa, voi johtaa parin tehokkaaseen yhteistyöhön.

Tehokas pari on joukko ihmisiä niin tiukasti sidottuna, että kokonaisuus on suurempi kuin osien summa. Tällaisen parin tuotos on suurempi kuin se että, he työskentelisivät yksinään. Kun pari on sitoutunut niin tiukasti yhteen, onnistumisen mahdollisuus nousee dramaattisesti. Parista tulee lähes pysäyttämätön.

Neuvo käynnissä oleville ja tuleville pariohjelmoijille: Poista skeptisyys, kehitä toive onnistumisesta ja tervehti pariasi sanomalla "Huuda minulle!". Tämä on ennenkuulumaton mahdollisuus emille kahdelle loistaa yhtenä.

Huuhdo

Väistämättä, pariohjelmoijat joutuvat työskentelemään yksinään. XP:n kokemuksen perusteella suurin osa virheistä voidaan jäljittää takaisin aikaan jolloin ohjelmoijat työskentelivät yksinään. Päätös huuhtoa tai tarkastaa yksin tehty työ voi tehdä pariohjelmoijat, tai päätöstä voidaan kannustaa. Kuitenkin, on tärkeää että kaikki itsenäisesti tehty ohjelmointi tarkastetaan.

Säännölliset tauot

Koska pariohjelmointi pitää molemmat jatkuvasti keskittyneenä ja oikeassa asiassa, se voi olla todella voimakasta ja henkisesti raskasta. Säännöllisesti pidettävät tauot ovat tärkeä osa kestävyyden ylläpitämiseksi seuraavaa tuotteliasta pariohjelmointi erää varten. Tauon aikana, on parasta unohtaa meneillä oleva työtehtävä ja palata sen kimppuun tuoreilla ajatuksilla kun aloitetaan. Toivottuja aktiviteetteja: Tarkista sähköposti, tee puhelinsoitto, selaa internetiä tai pidä kahvitauko.

Tauot pariohjelmoinnista

Ei ole pakollista työskennellä erikseen joka iltapäivä. Mutta, 50 % tarkkailluista ohjelmoijista, on hyväksyttävää työskennellä 10 - 50 % ajasta yksin. Moni pitää raskaista, kovaa ajattelemista tarvitsevista ongelmista ja loogisesta ajattelusta yksin. Moni on samaa mieltä että, yksinkertainen, hyvin määritelty koodi on tehokkaammin tehty yksittäiseltä ohjelmoijalta ja sen jälkeen tarkistettu parilla.

Ei kilpailua

Pariohjelmoinnissa kahdesta ohjelmoijasta tulee yksi. Näiden kahden välillä ei saa olla kilpailua; molempien tulee työskennellä yhteisen tarkoituksen hyväksi, aivan kuten artefakti olisi kehitetty yhden hyvän mielen ansiosta. Syy ongelmista tai virheitä ei saisi koskaan kohdistua pariin. Parin tulee voida luottaa toisen arvioon ja toistensa uskollisuuteen ryhmässä.

9 Pariohjelmointi kokeilu

Testin tarkoituksena on testata pariohjelmoinnin hyötyä ohjelmistojen tuotannossa. Testiryhmänä toimi ryhmä Qt ohjelmointi kielen parissa työskenteleviä ohjelmoijia, joiden kehityskohteena oli toteuttaa uusia ominaisuuksia QT pohjaiseen sormella käytettävään internet selaimeen MAEMO ympäristössä. Tehtävänä oli toteuttaa seuraavanlaisia ominaisuuksia mm. Sivuhistoria, kirjainmerkit, RSS tuki ja tuki evästeille. Ryhmään kuului projektipäällikkö sekä 5 ohjelmoijaa. Pariohjelmointi metodologian testaamiseen oli varattu kahden viikon mittainen aika, josta pariohjelmointiin käytettäisiin kaikki projektille varattu aika 900 työtuntia. Pariohjelmointikokeilu aloitettiin kokouksella, jossa kerrottiin hieman XP:stä sekä itse pariohjelmointimetodologiasta ja sen tavoitteista. Kokouksessa suunniteltiin myös tulevan kahden viikon työtehtävät, jotka olivat ohjelmointipainotteisia johtuen tulevasta kokeilusta. Samassa tilaisuudessa määrättiin alustavat työparit pariohjelmointiin. Kokeilun aikana vaihdeltiin työpareja säännöllisesti erilaisten pariohjelmointi kokemusten aikaansaamiseksi.

Ensimmäisenä pariohjelmointipäivänä pariohjelmointi ei tuottanut juurikaan lisäarvoa, sillä kyseessä oli vain toisen henkilöistä aiemmin työstämä koodi, johon ei tehty valtavan suuria muutoksia. Lisäksi ensimmäinen päivä sisälsi erilaisten ratkaisujen tehokkuuden kokeilemista erillisellä laitteella. Tähän ei pariohjelmoiti kovin hyvin soveltunut. Ohjelmointiosio selkeästi nopeutui uutta tehtäessä ja virhettä etsittäessä, mutta monesti muutokset olivat pieniä, missä pariohjelmoinnista ei ollut paljoa hyötyä.

Ohjelmistokoodiin tutustumisen jälkeen huomattiin suuri suunnitteluvirhe, joka löydettiin pariohjelmoinnin ansiosta. Jo pelkästään tämän virheen löytymisestä saatu hyöty ohjelmiston kehitykselle oli huomattava.

Heti aluksi huomattiin, että samalla tasolla olevan henkilön kanssa kommunikointi on huomattavasti helpompaa. On yleisesti tiedossa, että on helpompaa kommunikoida omalla tasolla olevan henkilön kanssa, kuin esimies tasolla tai vanhemmalle / kokeneemmalle työntekijälle. Pienet pariohjelmointiprojektit ovat helpompia toteuttaa samantasoisilla työpareilla, jotka ovat jo ennestään toisille tuttuja, jolloin saadaan henkilökemiat ja koodaustavat loksahtamaan paikoilleen.

Ensimmäisen viikon aikana jonkin verran hyötyä saatiin siitä, että moduulin tehokkuutta optimoitiin mobiililaitetta varten ja tähän saatiin monia ideoita yhdessä miettimällä. Itse pariohjelmointi toimi yllättävän hyvin, kun ottaa huomioon, että tähäs asti kaikki olivat työskennelleet yksin.

Toinen kokeilu viikko sisälsi vähemmän tukimista ja optimointia. Painopiste oli ohjelmakoodin tuottamisessa. Työparit pääsivät kehittämään uusia ominaisuuksia järjestelmään täydellä teholla. Pariohjelmointi menetelmä paljasti parhaat puolensa suurissa ja monimutkaisissa ohjelmistokoodeissa, joita on hankala kehittää ja ylläpitää perinteisillä menetelmillä.

10 Kysely

Lähetin jokaiselle kokeilussa mukana olevalle henkilölle kyselyn pariohjelmoinnista molempien kokeilu viikkojen lopuksi (liite 1).

Kysely oli pariohjelmointi painotteinen ja sisälsi seuraavanlaisia kysymyksiä:

- pariohjelmointiin käytetty aika päivässä
- pariohjelmoinnin tehokkuus asteikolla 1-5
- pariohjelmoinnin mielekkyys asteikolla 1-5
- pariohjelmoinnin tehokkuus toteutusvaiheessa 1-5
- kokemasi hyödyt pariohjelmoinnista?
- kokemasi haitat pariohjelmmoinista?

Palautteen paperilla sain neljältä henkilöltä (liite 2). Projektipäällikön ja kokeneen ohjelmistokehittäjän kanssa palaute saatiin lopussa pidettävällä kokouksella, jossa keskusteltiin pariohjelmointi kokeilusta ja sen onnistumisesta.

11 Tulokset

Ensimmäisen työpäivän jälkeen saatu palaute oli suurimmalta osalta negatiivista. Tämän aiheutti siirtyminen normaalista ohjelmoinnista pariohjelmoinnin pariin. Kommentteja olivat, työt eivät onnistu kun, joku hengittää niskaan jatkuvasti ja huomauttelee, miten pitäisi ongelma korjata. Tämä tulos oli odotettavissa, koska kaikki olivat tähän mennessä ohjelmoineet omassa rauhassa ilman, että kukaan tulee neuvomaan, miten asia pitäisi tehdä.

Ensimmäisen viikon jälkeen negatiiviset palautteet olivat kääntyneet päinvastaisiksi. Nyt suurin osa palautteesta oli erittäin positiivista. Suurimpia positiivisia asioita olivat:

- selvä ei työhön liittyvien asioiden väheneminen (sähköpostin lukeminen ja internetin selaaminen)
- ohjelmakoodin selvä parantuminen (rivien väheneminen, ohjelmakoodin virheiden väheneminen ja ongelmien ratkaisu).

Näiden lisäksi positiivisena koettiin kokenut / kokematon parin kokemattoman osapuolen selvä kompetenssin lisääntyminen ohjelmoinnista. Huonona puolena olivat henkilökemiat, palautteen antaminen työparille, sekä työparin osien liian nopea vaihtuminen, mikä katkaisi hyvin vauhtiin päässen ohjelmoinnin.

Tulokset toisen viikon jälkeen olivat samantapaiset kuin ensimmäisen viikon jälkeen. Positiivisessa mielessä muutoksia tuli kommunikointiin ja palautteen antamiseen, jotka parantuivat huomattavasti. Huonona puolena koettiin testin lyhyt kesto, koska juuri kun päästiin hyvään vauhtiin ja pariohjelmointi alkoi sujua, palattiin normaaliin yhden hengen ohjelmointiin.

Palaute, joka saatiin pariohjelmointi kokeilun aikana:

Positiiviset:

- Pikkuvirheet havaitaan usein välittömästi
- Yhdessä miettiminen tuottaa enemmän ideoita ja älykkäitä ratkaisuja
- Suunnittelu nopeutui myös lisääntyneenä luottona valittuun ratkaisuun, kun pystyi keskustelemaan valitusta ratkaisusta
- Koodiin aiemmin perehtymätön saa hyvän perehdytyksen.

Negatiiviset:

- Työtunteja kertyy paljon, tosin yllämainitut hyödyt saattavat lieventää tätä haittapuolta myöhemmässä vaiheessa (vähemmän korjattavia virheitä yms).
- Saattaa mennä joskus turhaksi jutusteluksi.
- Vaikea aloittaa, jos kyse on koodista, jota vain toinen henkilöistä on aiemmin työstänyt.

Kompetenssin kasvattaminen pariohjelmoinnin avulla on hyväksi havaittu tapa. Jatkuva keskustelu ja tutkiminen kasvattavat molempien osapuolien kompetenssia kyseiseltä alueelta.

Kokonaisuudessa testi oli erittäin onnistunut ja antoi paljon lisätietoa pariohjelmoinnin hyödyistä ja haitoista ohjelmistokehityksen näkökannalta.

12 Yhteenveto

Insinöörityön tavoitteena oli tutkia extreme programming metodin pariohjelmointi menetelmää ja sen soveltuvuutta ohjelmistojen tuotantoon. Tavoitteena oli antaa lukijalle yleiskuvaus XP-metodista ja laajempi kuvaus pariohjelmointi menetelmästä, sen periaatteista ja soveltuvuudesta ohjelmistojen-kehitykseen.

Mielestäni työ onnistui tavoitteessaan hyvin. Pariohjelmointi kokeilu saatiin testattua sopivan kokoisella porukalla ja aikaakin oli varattuna riittävästi kokeilulle. Itse kokeilusta saatu palaute ja pariohjelmoinnin toimiminen ohjelmistojen kehityksessä oli positiivista. Positiivisiksi asioiksi koettiin:

- Selvä ei työhön liittyvien asioiden väheneminen (sähköpostin lukeneminen ja internetin selaaminen)
- ohjelmakoodin selvä parantuminen (rivien väheneminen, ohjelmakoodin virheiden väheneminen ja ongelmien ratkaisu).

Negatiivisina puolina priohjelmoinnissa oli:

- Työtunteja kertyy paljon, tosin yllämainitut hyödyt saattavat lieventää tätä haittapuolta myöhemmässä vaiheessa (vähemmän korjattavia virheitä yms).
- Saattaa mennä joskus turhaksi jutusteluksi.
- Vaikea aloittaa, jos kyse on koodista, jota vain toinen henkilöistä on aiemmin työstänyt.

Toivottavasti tämä tutkintotyö auttaa ketteristä menetelmistä kiinnostuneita henkilöitä huomaamaan XP:n ja etenkin pariohjelmoinnin hyvät puolet ohjelmistokehityksessä.

Pariohjelmoinnin käyttöä olisi voinut vielä tutkia paritestauksessa ja opetuksessa, mutta työ täytyi rajata tarkkaan, jotta työn pituus ja aihealue saatiin järkeviin mittasuhteisiin.

Työn toinen tarkoitus oli syventää tietämystäni XP:n teoriasta ja periaatteista. Myös tässä tavoitteessa onnistuin kuten pitikin. Prosessin aikana tuli esille monia asioita, joita ei aiemmissa opinnoissa tai työtehtävissä ole tullut vastaan.

Lähteet

Painetut lähteet

1. Beck, Kent, Extreme programming explained embrace change, second edition. Addison-Wesley Professional, 2004. 244 s.

Sähköiset lähteet

2. XP programming [www-sivu]. [viitattu 10.9.2008] Saatavissa: http://www.extremeprogramming.org/map/project.html
3. XP programming [www-sivu]. [viitattu 5.9.2008] Saatavissa: http://www.agilemodeling.com/essays/agileDocumentationBestPractices.htm
4. Koodausvirheiden kustannukset [www-sivu]. [viitattu 16.6.2008] Saatavissa: http://www.softwarereality.com/lifecycle/xp/four_values.jsp
5. Palaute XP:ssä [www-sivu]. [viitattu 19.9.2008] Saatavissa: http://www.extremeprogramming.org/map/loops.html

Tietotekniikka, Ohjelmistotekniikka
Sami Männistö

LIITE 1 Pariohjelmointi kyselylomake

Nimi:

Kuinka paljon käytit päivässä aikaa pariohjelmointiin?

Arvioi pariohjelmoinnin tekokkuutta asteikolla 1-5 (1=ei hyötyä, .. ,5=erittäin hyödyllinen)

Arvosana pariohjemoinnin mielekkyydestä asteikolla 1-5

Arvio pariohjelmoinnin tehokkuudesta ohjelmistoa suunnitellessa asteikolla 1-5

Arvio pariohjelmoinnin tehokkuudesta toteutusvaiheessa asteikolla 1-5

Käyttäisitkö pariohjelmointia vapaaehtoisesti?

Kokemasi hyödyt pariohjelmoinnista

Kokemasi haitat pariohjelmoinnista

Vapaata palautetta pariohjelmoinnista

Tietotekniikka, Ohjelmistotekniikka
Sami Männistö

LIITE 2 Pariohjelmointi kyselyn tulokset

Kuinka paljon käytit päivässä aikaa pariohjelmointiin?
7 tuntia
Arvioi pariohjelmoinnin tekokkuutta asteikolla 1-5 (1=ei hyötyä, .. ,5=erittäin hyödyllinen)
3
Arvosana pariohjemoinnin mielekkyydestä asteikolla 1-5
3
Arvio pariohjelmoinnin tehokkuudesta ohjelmistoa suunnitellessa asteikolla 1-5
2
Arvio pariohjelmoinnin tehokkuudesta toteutusvaiheessa asteikolla 1-5
4
Käyttäisitkö pariohjelmointia vapaaehtoisesti?
Kyllä

Kokemasi hyödyt pariohjelmoinnista:

- **Ei rönsyilyä (postien lukemista, netin selaamista jne)**
- **Virheiden löytäminen**
- **Koodin "puhdistuminen" kun kaveri nalkuttaa vieressä koodin laadusta: kaikenkarvaiset häkit vähenevät**
- **Ongelmia ei kierretä, ne ratkaistaan**
- **Koodin laatu parempaa (ehkä), häkit ja virheet vähenee**
- **Ajankäyttö tehokkaampaa**
- **Tiedonetsintä ja ongelmien ratkaisu tehokkaampaa kun kaksi ihmistä miettii**

Kokemasi haitat pariohjelmoinnista:

- **Rivejä tulee vähemmän**
- **Kaveri hermostuu tuijotuksesta**
- **Palautteen antaminen vaikeaa**
- **Hermot menee, kun tekisi itse mieli koodata**
- **Kommunikaatio välillä vaikeaa**

- **Koodaus-"flown" rikkominen liian lyhyillä vaihdoilla :)**
- **Toisen koodauksen katsominen välillä väsyttävää, kun käsillä on jonkin peruskoodin**

Vapaata palautetta pariohjelmoinnista:
Vaatii aikaa ja asennoitumista, että tästä olisi jotain hyötyä

Tietotekniikka, Ohjelmistotekniikka
Sami Männistö

Kuinka paljon käytit päivässä aikaa pariohjelmointiin?
5,5 tuntia
Arvioi pariohjelmoinnin tekokkuutta asteikolla 1-5 (1=ei hyötyä, ..
,5=erittäin hyödyllinen)
2
Arvosana pariohjemoinnin mielekkyydestä asteikolla 1-5
2
Arvio pariohjelmoinnin tehokkuudesta ohjelmistoa suunnitellessa
asteikolla 1-5
4
Arvio pariohjelmoinnin tehokkuudesta toteutusvaiheessa asteikolla 1-5
2
Käyttäisitkö pariohjelmointia vapaaehtoisesti?
en

Kokemasi hyödyt pariohjelmoinnista:

Kirjoitusvirheet ja ajatusvirheet tippuivat selkeästi, kun oli vierellä kaveri puuttumassa niihin. Tämä nopeutti ohjelman kirjoittamista selkeästi. Suunnittelu nopeutui myös lisääntyneenä luottona valittuun ratkaisuun, kun pystyi keskustelemaan valitusta ratkaisusta. Pariohjelmointi toimi myös hyvänä tapana tutustuttaa ohjelmaan, koska parini ei ollut toteutusta aikaisemmin nähnyt.

Kokemasi haitat pariohjelmoinnista:

Ohjelman tekemisen ollessa erilaisten kokeilujen tekemistä ja runsasta kääntämistä, varsinkin erilliselle laitteelle kääntämistä, menee suuri aika hukkaan odotellessa ja ihmetellessä.

Vapaata palautetta pariohjelmoinnista:

Tämä palaute syntyi ensimmäisen päivän kokeilusta 13.6. Ohjelmointi sisälsi erilaisten ratkaisujen tehokkuuden kokeilemista erillisellä laitteella. Tähän ei pariohjelmoiti kovin hyvin soveltunut. Ohjelmointi osio selkeästi nopeutui uutta tehtäessä ja virhettä etsittäessä, mutta monesti muutokset olivat pieniä, missä pariohjelmoinnista ei ollut paljoa hyötyä. Parini oli myös hankala lähteä itse ohjelmoimaan, koska tuli mukaan tuntemattomaan toteutukseen.

Tietotekniikka, Ohjelmistotekniikka
Sami Männistö

Kuinka paljon käytit päivässä aikaa pariohjelmointiin?
7 tuntia
Arvioi pariohjelmoinnin tekokkuutta asteikolla 1-5 (1=ei hyötyä, ..
,5=erittäin hyödyllinen)
2
Arvosana pariohjemoinnin mielekkyydestä asteikolla 1-5
3
Arvio pariohjelmoinnin tehokkuudesta ohjelmistoa suunnitellessa
asteikolla 1-5
4
Arvio pariohjelmoinnin tehokkuudesta toteutusvaiheessa asteikolla 1-5
2
Käyttäisitkö pariohjelmointia vapaaehtoisesti?
Voisin käyttää, mikäli tilanne on sopiva, esim. vastaavanlaisissa tilanteissa olisi pariohjelmoinnilla saatu aikaisemmin hyviä tuloksia.

Kokemasi hyödyt pariohjelmoinnista:

- **Pikkuvirheet havaitaan usein välittömästi**
- **Yhdessä miettiminen saattaa tuottaa enemmän ideoita ja älykkäitä ratkaisuja**

Kokemasi haitat pariohjelmoinnista:

- **Työtunteja kertyy paljon, tosin yllämainitut hyödyt saattavat lieventää tätä haittapuolta myöhemmässä vaiheessa (vähemmän korjattavia virheitä yms).**
- **Saattaa mennä joskus turhaksi jutusteluksi.**
- **Vaikea aloittaa, jos kyse on koodista, jota vain toinen henkilöistä on aiemmin työstänyt. Toisaalta hyvänä puolena tässä on se, että koodiin aiemmin perehtymätön saa mahdollisesti hyvän perehdytyksen. Tämä vie kuitenkin molempien aikaa.**

Vapaata palautetta pariohjelmoinnista:

Ensimmäisenä pariohjelmointipäivänä pariohjelmointi ei tuottanut juurikaan lisäarvoa, sillä kyseessä oli vain toisen henkilöistä aiemmin työstämä koodi, johon ei tehty valtavan suuria muutoksia. Ensimmäisen viikon aikana jonkin verran hyötyä saatiin siitä, että moduulin tehokkuutta optimoitiin mobiililaitetta varten ja tähän saatiin monia ideoita yhdessä miettimällä.

Tietotekniikka, Ohjelmistotekniikka
Sami Männistö

Kuinka paljon käytit päivässä aikaa pariohjelmointiin?
6 tuntia
Arvioi pariohjelmoinnin tekokkuutta asteikolla 1-5 (1=ei hyötyä, .. ,5=erittäin hyödyllinen)
3
Arvosana pariohjemoinnin mielekkyydestä asteikolla 1-5
2
Arvio pariohjelmoinnin tehokkuudesta ohjelmistoa suunnitellessa asteikolla 1-5
3
Arvio pariohjelmoinnin tehokkuudesta toteutusvaiheessa asteikolla 1-5
3
Käyttäisitkö pariohjelmointia vapaaehtoisesti?
En

Kokemasi hyödyt pariohjelmoinnista:

- **Virheet löytyivät paremmin.**
- **Löytää bugit ja epäloogisuudet paremmin**

Kokemasi haitat pariohjelmoinnista:

Hommiin menee kahden ihmisen työtunnit yhden sijaan, koska koodausnopeus ei ainakaan kasva.

Vapaata palautetta pariohjelmoinnista:

Koodausnopeus pikemminkin hidastuu pariohjelmoitaessa, koska tulee paineita kun toinen vahtii selän takana, eikä tällöin saa miettiä omassa rauhassa eikä saa typottaa.