Esimerkkiotsikko

Kenny Heinonen

Aine
Helsingin Yliopisto
Tietojenkäsittelytieteen laitos

Helsinki, 6. helmikuuta 2013

Sisältö

1 Johdanto

2 Ryhmätyöskentelyn merkitys ketterissä menetelmissä

Ohjelmistoalalla tarvitaan monenlaisia teknisiä taitoja, jotta etenkin suuremmissa projekteissa saadaan toteutettua kaikki kehityksen vaiheet kattavasti. Nämä ohjelmistokehityksen vaiheet voidaan jakaa karkeasti vaatimusmäärittelyyn, suunnitteluun, toteutukseen, testaukseen ja ylläpitoon [2]. Sen lisäksi, että eri vaiheet vaativat eri taitoja, myös yksittäinen vaihe kysyy laajaa osaamista. Tämän seurauksena tulee tarve koota joukko osaavia ihmisiä toteuttamaan yhteistyössä kaikki kehityksen vaiheet. Onkin hyvin tavanomaista, että ohjelmistoprojektit toteutetaan ryhmätyönä. Useat prosessimallit jopa sanelevat miten ryhmätyöskentely tapahtuu, jotta kehitys sujuisi luontevammin ja tuotteliaammin.

Ryhmätyöskentelyä voi tehdä monin tavoin. On esimerkiksi tilanteita, joissa ryhmät sijaitsevat samoissa oloissa, mutta työskentelevät silti täysin erillään toisistaan. Tämän seurauksena kommunikointi kärsii ja voi olla epäsäännöllistä. Ketterät menetelmät sen sijaan suosivat, että yksittäisen ryhmän jäsenillä on yhteinen työtila ja kommunikointi on mahdollisimman kasvokkaista. Asioista neuvottelu on tällöin nopeaa, suoraviivaista ja väärinkäsitysten mahdollisuus pienenee. Seuraavissa osioissa tarkastellaan muutamia ketteriä prosessimalleja nähdäksemme miten ryhmätyöskentely ja kommunikointi otetaan niissä huomioon.

2.1 Extreme Programming

2.1.1 Pariohjelmointi

Extreme Programming (XP) on eräs ketterä menetelmä, jonka käytänteisiin kuuluu *pariohjelmointi* ("pair programming"). Käytännössä pariohjelmoinnissa kaksi henkilöä ohjelmoivat pareittain, jakaen saman tietokoneen. Henkilöt eivät kuitenkaan ohjelmoi samaan aikaan, vaan heille on nimetty kaksi roolia: toinen on *ajaja* ("the driver"), ja toinen *navigoija* ("the navigator"). Ajajan tehtävänä on yksinkertaisesti kirjoittaa koodia. Navigoijan tehtävänä on analysoida jatkuvasti kirjoitettua koodia ja kertoa ajajalle mitä tehtäviä heidän tulee milloinkin toteuttaa. Näin ajaja voi keskittyä pelkästään ohjelmointiin. Sovituin väliajoin henkilöt voivat vaihtaa rooleja. Pariohjelmointi voidaan nähdä ryhmätyöskentelynä - kaksi ihmistä suorittavat yhteistä tehtävää saavuttaakseen saman päämäärän. [3]

Pariohjelmointi korostaa yhteistyötä ja kommunikointia. Tästä on monia etuja, jotka tekevät hyvää sekä yksilölle, ryhmälle että projektille. Tarkastel-

laan näitä etuja seuraavaksi.

- **Koodin laatu**

  Kun kaksi henkilöä pariohjelmoivat ja ratkaisevat samaa ongelmaa, lopputulos on usein tehokkaampi verrattuna siihen, että yksi henkilö tekisi kaiken. Parit pystyvät helposti keskustelemaan keskenään siitä mitä heidän tulisi seuraavaksi tehdä ja he voivat jakaa ideoita saadakseen koodista laadukkaamman tai ratkaistakseen jonkin ongelman. Kuten sanottua, ajaja ohjelmoi ja navigoija katselmoi koodia. Näin navigoija pystyy tekemään tärkeitä huomioita ja pohtimaan kuinka koodista saataisiin laadukkaampi, kun taas ajaja voi keskittyä kirjoittamiseen.

- **Keskittymiskyky ja itsekuri**

  Pariohjelmoinnin on todettu parantavan keskittymiskykyä sillä toisen henkilön läsnäolo estää herkemmin yksilöä laiskottelemasta tai rikkomaan XP:n käytänteitä, kuten TDD:tä (Test Driven Development).

- **Oppiminen**

2.2 Scrum

2.3 Lean/FDD?

3 Persoonallisuustyypit kehityksen eri vaiheissa

3.1 Vaatimusmäärittely

3.2 Suunnittelu

3.3 Toteutus

3.4 Testaus

3.5 Ylläpito

4 Ryhmätyötaitojen parantaminen

5 Lähteet

[1] Begel, Andrew ja Nachiappan Nagappan: *Pair programming: what's in it for me?* Teoksessa *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, ESEM '08, sivut 120–128, New York, NY, USA, 2008. ACM, ISBN 978-1-59593-971-5. `http://doi.acm.org/10.1145/1414004.1414026`.

[2] Capretz, Luiz Fernando ja Faheem Ahmed: *Making Sense of Software Development and Personality Types*. IT Professional, 12(1):6–13, tammikuu 2010, ISSN 1520-9202. http://dx.doi.org/10.1109/MITP.2010.33.

[3] Shore, James ja Shane Warden: *The art of agile development*, luku 5. O'Reilly, first painos, 2007, ISBN 9780596527679.