Miten avoimen lähdekoodin kehitys ja ketterät menetelmät kohtaavat?

Jarl-Erik Malmström

Referaatti
Helsingin Yliopisto
Tietojenkäsittelytieteen laitos

Helsinki, 4. helmikuuta 2013

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

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

Tekijä — Författare — Author
Jarl-Erik Malmström

Työn nimi — Arbetets titel — Title

Miten avoimen lähdekoodin kehitys ja ketterät menetelmät kohtaavat?

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Referaatti	4. helmikuuta 2013	7

Tiivistelmä — Referat — Abstract

Tiivistelmä.

Avainsanat — Nyckelord — Keywords
agile, ketterä, open source, avoin

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

1 Johdanto

Avoimen lähdekoodin (open source software) kehittäjäyhteisö on tyypillisesti maantieteellisesti hajaantunut. Yhteisön jäsenten etäisyys vaikeuttaa jäsenten välistä kommunikaatiota, verrattuna perinteiseen työyhteisöön, missä jäsenet työskentelevät yleensä samassa tilassa.

Ketterät ohjelmistotuotantomenetelmät (agile methods) eivät vaikuta sopivilta hierarkiseen ja hajaantuneeseen organisaatiorakenteeseen, mutta avoimen lähdekoodin projektit toteuttavat usein ketterien menetelmien periaatteita. Projektit ovat valmiita muutoksille, vastaavat jatkuvaan palautteeseen toimittamalla uusia toiminnallisuuksia, arvostavat yhteistyötä ja kohtaavat haasteita.

Linux käyttöjärjestelmän levinneisyys ja ohjelmistoalan muutos kohti ketteriä menetelmiä ovat osoittaneet, että avoimilla projekteilla ja ketterillä menetelmillä molemmilla on etunsa. kumpikaan ei ole kuitenkaan tarjonnut yksiselitteistä ratkaisua ohjelmistotuotannon ongelmiin. Referoidussa artikkelissa[1] Hugo Corbucci ja Alfredo Goldman pyrkimys yhdistämään molempien menetelmien edut.

2 Avoimen lähdekoodin ketterät menetelmät

Hugo Corbucci ja Alfredo Goldman esittävät, että avoimen lähdekoodin projekteissa toteutetaan ketteryyttä ja monien avoimen kehityksen ominaisuuksia voidaan suoraan liittää ketterään julistukseen (agile manifesto).

He pitävät ketteränä sellaisia ohjelmistosuunnittelun metodeja, mitkä seuraavat ketterän julistuksen periaatteita: yksilöt ja vuorovaikutus yli prosessien ja työkalujen, toimiva ohjelmisto yli kattavan dokumentaation, asiakasyhteistyö yli sopimusneuvottelun ja muutoksiin sopeutuminen yli suunnitelman seuraamisen.

2.1 Yksilöt ja vuorovaikutus

Kirjoittajat sanovat avoimen lähdekoodin projektien olevan lähes poikkeuksetta ketteriä. He kirjoittavat, että useissa tutkimuksissa on havaittu avoimen lähdekoodin ohjelmistokehityksessä käytettävän kohtuullisesti työkaluja yhteisön kommunikaation ylläpitämiseksi. Versionhallinta, projektin kotisivu ja postituslistat olivat pääasialliset yhteydenpitovälineet käyttäjien ja kehittäjien keskuudessa. Yhteisön yksilöt ja sen vuorovaikutus kiinnostavan asian ympärillä ovat koko avoimen kehityksen ydin. Yhteistoiminta liittyy useimmiten lähdekoodin ympärille ja dokumentaation laadintaan. Yhteistoiminta vie prosesseja eteenpäin ja muokkaa yhä paremmin yhteisön tarpeisiin sopiviksi.

2.2 Toimiva ohjelmisto

Hieman yli puolet avoimen lähdekoodin projekteista päivittävät ja tarkistavat dokumentaatiota usein ja kolmannes ylläpitää dokumentteja kertoakseen kuinka ohjelmiston osat toimivat ja kuinka se on organisoitu. Kirjoittajat näkevät dokumentoinnin tukevan projekteja eikä olevan projektien pääasiallinen tarkoitus.

2.3 Asiakasyhteistyö

Asiakassopimukset ovat harvoin ongelma avoimissa projekteissa. Asiakas palkkaa kehittäjän tai organisaation kehittämään toiminnallisuuden. Liiketoimintamalli ei takaa yhteistyötä asiakkaan kanssa, mutta yhteistoiminta on kuitenkin koko avoimen kehityksen perusta. Asiakas on yhteistyössä toiveidensa mukaan.

2.4 Muutoksiin sopeutuminen

Avoimen lähdekoodin projekteilla on yleensä virstanpylväitä (milestone) ja julkaisuja (releses). Monilla projekteilla on vain lyhyen ajan suunnitelmia. Jos pitkän ajan suunnitelmia on, ne ovat tavoitteita ilman painetta niiden toteuttamiseen. Hugo Corbucci ja Alfredo Goldman kirjoittavat sopeutumisen olevan välttämätöntä kilpailuhenkisessä ympäristössä. Projektien on vastattava käyttäjien muuttuviin tarpeisiin, jos yhteisö haluaa sen selviytyvän kilpailusta.

3 Ketterä ohjelmistokehitys avoimeksi

3.1 Tallettajan rooli

Monilla avoimen lähdekoodin projekteilla on pieni määrä henkilöitä, joilla on oikeus lisätä koodia runkohaaraan (trunk branch). Runkohaara muodostaa uuden julkaistavan ohjelmiston. Yhteisö luottaa tallettajien (committer) koodiarviointiin. Ketterässä ohjelmistokehityksessä päinvastoin kaikilla kehittäjillä on oikeus lisätä koodia runkohaaraan.

Hugo Corbucci ja Alfredo Goldman esittävät valvontaa tuotantokoodin yksinkertaisuuden takaamiseksi. Useimmilla kettterillä menetelmillä on kehittäjätiimeissä johtaja tai valmentaja, joka on yleensä muita jäseniä kokeneempi. Valvontatehtävä voisi olla tiimin johtajalla, ellei se aiheuttaisi viivästystä tuotannossa tai häiritsisi muita hänen tehtäviään. Toinen vaihtoehto olisi vuorotella valvontavastuuta osalla kehittäjiä.

3.2 Tulokset julki

Kirjoittajien mielestä suljetun lähdekoodin ohjelmistot voisivat hyötyä julkisista vianseurantajärjestelmistä (bug tracking) ja testituloksista. Ohjelmistojen tuottajien olisi hyväksyttävä siten koodia julkaistavaksi. Julkiset työkalut kannustaisivat käyttäjiä osallistumaan tuotantoprosessiin.

Ketterässä kehityksessä vianseuranta ja testit ovat tärkeitä työkaluja. Ketterät menetelmät eivät esitä, että asiakkaan tulisi käyttää kehitystyökaluja. Välineet ovat usein myös vaikeakäyttöisiä muille kuin kehittäjille.

Useimmissa menetelmissä asiakasta pidetään osana kehitystiimiä. Käytöspohjainen suunnittelu (Behavior Driven Development) on tuonut testien osalta ohjelmistokehitykseen selkeyttä ei-teknisen henkilöstön kannalta.

Avoimen kehityksen julkisuus ei koske vain testejä ja vianseurantaa. Kehittäjien ja ulkopuolisten ihmisten keskustelut kirjautuvat sähköpostilistojen arkistoihin. Keskustelua näiden arkistojen ulkopuolella pyritään välttämään, jotta kaikilla olisi mahdollisuus kommentoida ja antaa uusia ideoita.

Hugo Corbucci ja Alfredo Goldman kirjoittavat, että seurattavuuden puuttuminen on ketterien menetelmien heikkous. Useimmat niistä olettavat suunnitelman (design) kehittyvän ajan kuluessa ja tarpeen mukaan. Kehitys tapahtuu pyyhittävillä tauluilla, ja vaikka ne tallennettaisiin, kehityssuuntaan johtanut keskutelu ei arkistoidu. Keskustelu on tehokas tapa kommunikoida mutta myös katoavaista. Keskustelijoiden on vaikea löytää enää täsmällistä informaatiota keskustelun jälkeen.

4 Ketterien menetelmien edut avoimelle kehitykselle

Työkalut voisivat helpottaa ketterien menetelmien käyttöönottoa avoimissa projekteissa ja edistää kehitysprosessia. Hajautetun pariohjelmoinnin työkalut ovat olleet kohtuullisesti tutkimuksen ja kehityksen kohteena. Muiden käytänteiden tukemiseen tarkoitettuja välineitä on ilmestynyt vain vähän. Kirjoittajat esittivät seuraavissa kappaleissa esiintyviä käytänteitä ja työkaluja avoimen kehityksen edistämiseen ketterien menetelmien kautta.

4.1 Tietoa sisältävä työtila

Ketterän kehitystiimin tulisi työskennellä tilassa missä on työhön tarvittava informaatio. Informaatiota tulisi pitää yllä tietty henkilö tai useampi. Tiedon kerääminen on yleensä aikaa vievää, ja siksi vain harvat avoimen lähdekoodin yhteisöt päivittävät metriikkaa ja dataa Internet-sivuillaan. Internet-pohjainen työkalu tai liitännäinen edistäisi informaation hallintaa, lisäämistä ja esittämistä avoimessa projektissa. Se tulisi olla helposti yhdistettävissä koodivarastoon ja projektin Internet-sivuille.

4.2 Tarinat

XP ehdottaa (eXtreme programming) suunnitteluvaiheessa vaatimusten keräämistä käyttötapauksista (user story) korteille. Tämän käytännön tarkoituksena on helpottaa projektin seuraavan vaiheen löytämistä ja vaatimusten tärkeysjärjestyksen muuttamista ajan kuluessa.

Hugo Corbucci ja Alfredo Goldman kirjoittavat, että avoimissa projekteissa kehityksen vaatimusmäärittely perustuu tavallisesti vianhallintajärjestelmään. Puuttuva toiminnallisuus raportoidaan vikana (bug) mikä pitäisi korjata. Keskustelu ja korjausehdotukset liittyvät ainoastaan tuohon virheeseen tai vikaan. Ongelmana on, että vaatimusten muuttaminen ja julkaisusuunnitelman tekeminen on aikaa vievää sekä perustuvat dokumentoimattomiin oletuksiin. Lisäksi kokonaiskuvaa vaatimuksista on vaikea saada.

tärkeimpien toiminnallisuuksien löytäminen ja niiden järjestyksen muuttaminen, yhteisön palautteen perusteella, ovat toimivan ohjelmiston kehittämiselle. Tämän saavuttamiseksi kirjoittajat ehdottavat työkalua vikojen esittämiseksi siirreltävinä elementteinä julkaisusuunnitelmassa. Yhteisön tiedon hyödyntämiseksi työkalun pitäisi priorisoida viat ja sisältää käyttäjien luoman sisällön Wikipedian tapaan.

4.3 Yleiskatsaus

Yleiskatsauksen (retrospective) ideana on tuoda kehitystiimi yhteen säännöllisesti keskustelemaan projektin suunnasta. Kirjoittajat ehdottavat avoimelle kehitykselle Internet-pohjaista työkalua kehityksen aikajanan ja koodivaraston yhdistämiseksi. Tiimin voisi lisätä aikajanalle huomautuksia. Tiimin johtaja voisi luoda raportin jäsenille tietoa sisältävään työtilaan.

4.4 Tapaamiset

Scrum menetelmässä tiimi kokoontuu ja jokainen tiimin jäsen kertoo mitä he ovat tehneet ja mitä aikovat seuraavaksi tehdä. Avoimen kehityksen kannalta ongelmana on hajaantuneisuus. Useilla projekteilla IRC-kanavat ovat osittainen ratkaisu keskittää keskustelu kehityksen aikana. Se ei takaa että kaikki tietäisivät mitä toiset ovat tekemässä. Kirjoittajat ehdottavat, että IRC-kanavien keskustelut pitäisi kirjata ja viimeisimmät viestit pitäisi näyttää uusille tulijoille.

5 Yhteenveto

Hugo Corbucci ja Alfredo Goldman esittivät ketterien menetelmien ja avoimen lähdekoodin kehityksen edistämisestä niiden käytänteitä yhdistämällä.

Useat projektit ovat omaksuneet tekniikoita ketteristä menetelmistä ollakseen vastaanottavaisempia käyttäjiä kohtaan. Kokonaisvaltaisempi menetelmäkuvaus, mikä huomioisi avoimen kehityksen osa-alueet lisäisi menetelmien käyttöönottoa.