# Vaatimusten ja konfiguraation hallinta avoimessa ohjelmisto-kehityksessä

Anu Ranta

Helsinki 1.10.2008
Avoin ohjelmistokehitys seminaari
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

| Tiedekunta/Osasto – Fakultet/Sektion – Faculty/Section | Laitos – Institution – Department | |
|---|---|---|
| Matemaattis-luonnontieteellinen tdk | Tietojenkäsittelytieteen laitos | |
| Tekijä – Författare – Author | | |
| Anu Ranta | | |
| Työn nimi – Arbetets titel – Title | | |
| Vaatimusten ja konfiguraation hallinta avoimessa ohjelmistokehityksessä | | |
| Oppiaine – Läroämne – Subject | | |
| Tietojenkäsittelytiede | | |
| Työn laji – Arbetets art – Level | Aika – Datum – Month and year | Sivumäärä – Sidoantal – Number of pages |
| Seminaari | 1.10.2008 | 9 sivua |

Tiivistelmä – Referat – Abstract

Vaatimusten hallinnan menetelmät ovat vakiinnuttaneet asemansa perinteisessä ohjelmistotekniikassa. Siihen kuuluva prosessi tunnetaan hyvin ja sitä pidetään projektin onnistumisen kannalta hyvin merkittävässä asemassa. Tässä kirjoitelmassa tutkitaan, esiintyykö avoimen ohjelmistokehityksen puolella vastaavia asemansa vakiinnuttaneita prosesseja vaatimusten hallitsemiseksi. Lisäksi tarkastellaan konfiguraation hallintaa avoimessa ohjelmistokehityksessä.

Avainsanat – Nyckelord – Keywords

Avoin ohjelmistokehitys, vaatimukset, konfiguraatio

Säilytyspaikka – Förvaringställe – Where deposited

Muita tietoja – Övriga uppgifter – Additional information

# Sisältö

# 1 Johdanto

Vaatimusmäärittelyllä on hyvin merkittävä rooli perinteisen ohjelmistotekniikan prosesseissa. Tämän vuoksi herääkin kysymys, onko vaatimusmäärittelyllä yhtä merkittävä asema myös avoimen kehityksen puolella. Jos näin on, eroavatko vaatimusten hallintamuodot merkittävästi toisistaan. Tässä kirjoitelmassa pyritään aluksi löytämään vastaus edellä esitettyyn kysymykseen vertailemalla vaatimusten hallintaa perinteisessä ja avoimessa ohjelmistokehityksessä. Tämän jälkeen luodaan katsaus siihen, kuinka avoimen kehityksen ohjelmistojen kehitystä ja elinkaarta hallinnoidaan. Tarkistellaan sitä, kuinka avoimessa kehityksessä on hoidettu rinnakkaisen työn asettamat haasteet. Lopuksi luodaan vielä katsaus muutosten hallintaan.

# 2 Vaatimusten hallinta

Vaatimusmäärittelyä pidetään yleensä projektin onnistumisen kannalta hyvin merkittävässä asemassa. Tämän vuoksi perinteisessä ohjelmistotekniikassa vaatimusmäärittely prosessina on hyvin pitkälle hiottu ja noudattaa tiettyjä kurinalaisuuksia. Prosessin lähtökohtana oletetaan yleisesti, että kehitettävän tuotteen sidosryhmät tunnetaan suhteellisen hyvin ja ne ovat tietolähteinä tavoitettavissa [McO06]. Vaatimukset kartutetaan lähteitä apuna käyttäen sekä analysoidaan, spesifioidaan ja validoidaan yleensä perusteellisesti ennen varsinaisen kehitystyön aloittamista.

Avoimessa ohjelmistokehityksessä ohjelmiston sidosryhmät eivät useinkaan ole yhtä selkeästi tunnistettavissa [McO06]. Näin jo projektin lähtökohdat vaatimusmäärittelylle ovat yleensä hyvin erilaiset kuin perinteisissä kaupallisissa projekteissa. Erot eivät jää vain sidosryhmien tasolle vaan projektit eroavat lukuisilta ominaisuuksilta toisistaan. Avoimet projektit eivät muun muassa useinkaan työskentele projektin hallinnan rajoissa, jossa projektia ohjaa tarkoin rajattu aikataulu ja budjetti [SJN06]. Tämä on

omiaan vaikuttamaan myös projektin prosessin kulkuun, jossa vaatimukset ovat harvoin perusteellisesti määritetty ennen varsinaisen ohjelmoinnin aloittamista. Vaatimuksia tulee jatkuvasti esiin vielä käyttöönoton jälkeenkin. Oman haasteen avoimen kehityksen projekteille luo usein myös käyttäjien ja kehittäjien sijoittuminen eripuolille maailmaa. Kehitystyössä onkin monesti pystyttävä tasapainottelemaan paikallisten ja laajemman yhteisön tarpeiden ja vaatimusten välillä [McO06].

Perinteiset ohjelmistoprosessit eivät siten sellaisinaan useinkaan esiinny avoimen ohjelmistokehityksen puolella, mutta tietynlaisia prosesseja on sielläkin havaittavissa.

## 2.1 Vaatimusten kartutus

Perinteisesti vaatimusten hallinta alkaa vaatimusten kartutuksella, jossa vaatimuksia kaivetaan esiin tarkoin mietittyjä menetelmiä apuna käyttäen. Työvaiheen pyrkimyksenä on koota riittävät tiedot, jotta tehtäväalue ja siihen sisältyvä ongelma voidaan ymmärtää mahdollisimman hyvin. Tietolähteinä käytetään toteutettavan järjestelmän sidosryhmiä. Kartutusmenetelmät perustuvat hyvin pitkälti suoriin kontakteihin sidosryhmien kanssa. Menetelminä käytetään esimerkiksi suoria haastatteluita, kyselylomakkeita ja käyttäjien tarkkailua. Aikaisemman järjestelmän dokumentit toimivat myös usein vaatimusten tietolähteinä [BrI02,s. 41–50].

Avoimessa ohjelmistokehityksessä ei vastaavanlaista vaatimusten suoranaista esiin kaivelua sellaisenaan tavallisesti esiinny. Vaatimukset tulevat esiin erilaisissa yhteisön jäsenien välisissä keskusteluissa siitä, mitä ohjelman tulisi tai ei tulisi tehdä. Vaatimukset ovat löydettävissä yhteisön Web-sivuilta sähköisessä muodossa erilaisten postituslistojen ja keskustelufoorumeiden yhteydestä [Scac02]. Vaatimuksia lähestytään periaatteessa aivan vastakkaisesta suunnasta. Vaatimuksia ei erikseen kerätä sidosryhmiltä, vaan käyttäjät tai kehittäjät tuovat vaatimuksia itse esille yhteisön Web-sivuilla, jossa pyrkivät vakuuttamaan muut jäsenet siitä, että muutokset olisi syytä toteuttaa järjestelmään.

Avoimissa ohjelmistoprojekteissa vaatimukset tulevatkin usein kehittäjiltä itseltään. Joku huomaa puuttuvan ominaisuuden ja päättää toteuttaa sen. Muita merkittäviä lähteitä ovat muun muassa käyttäjien antama palaute, erilaiset standardit, kaupalliset ohjelmat sekä kehittäjien halu oppia ymmärtämään jokin osa-alue paremmin [BaM02].

Perinteisesti kartutuksen tuloksena syntyy kasa muistiinpanoja vaatimusmäärittelyn myöhempiä vaiheita varten. Avoimessa ohjelmistokehityksessä esiin tulleita vaatimuksia ei yleensä erikseen kirjata ylös vaan ne ovat tarvittaessa löydettävissä yhteisön keskustelupalsta- ja sähköpostiarkistojen joukosta [Scac02].

## 2.2 Vaatimusten analysointi ja spesifiointi

Perinteisesti ohjelmistoprosessi jatkuu kartutettujen vaatimusten analysoinnilla ja spesifioinnilla. Analyysillä varmistetaan, että tehtäväalueen rakenne ja ominaisuudet sekä sidosryhmien tarpeet ymmärretään. Analyysin on tarkoitus jäsentää tehtäväalueen rakenne ja sisältö sekä vaatimusten väliset suhteet. Toisin sanoen kuvata ongelma, johon etsitään ratkaisua, ja sidosryhmien tarpeet. Siinä missä analyysivaiheessa kuvataan todellisuutta, keskitytään spesifiointivaiheessa siihen, miten järjestelmän tulee toimia, jotta se täyttäisi sille asetetut vaatimukset. Spesifiointidokumenttia käytetään muun muassa suunnittelun, toteutuksen ja testauksen perustana [BrI02,s. 51–209].

Avoimessa ohjelmistokehityksessä on hyvin vähän, jos ollenkaan, vastaavanlaista vaatimusten kontrolloitua analyysiä havaittavissa. Vaatimukset ovat usein analysoitavissa projektin sivustoille liitetyistä teknisistä kuvauksista. Vaatimusten ymmärtäminen vaatii tarkkaa teknisen kuvauksen läpikäyntiä ja mahdollisesti muihin, asiaan liittyviin, materiaaleihin tutustumista [Scac02].

Avoimessa ohjelmistokehityksessä esiintyy harvoin täsmällisiä spesifiointi tai mallinnus prosesseja. Web-sivuilla käydyt keskustelut yhdessä ohjelmiston lähdekoodin kanssa muodostavat vaatimusten spesifioinnin [GSP03].

Yhteisön postituslistoilta ja keskustelufoorumeista löytyvät vaatimuksia kuvaavat viestiketjut luovat tarpeen tiivistää ja kuvata tarkemmin niissä esiintyvät vaatimukset toteutettavan järjestelmän toiminnallisiksi ja ei-toiminnallisiksi vaatimuksiksi [Scac02]. Vaatimukset spesifioidaankin erilaisissa näihin liittyvissä keskusteluverkostoissa.

## 2.3 Vaatimusten validointi

Vaatimusten validointi ei perinteisessä ohjelmistokehityksessä esiinny omana vaiheenaan vaan kaikkien vaiheiden osana. Validoinnilla pyritään tyypillisesti löytämään ja korjaamaan vaatimusmäärittelyssä esiintyviä virheitä ja puutteita. Varmistetaan, että kaikki tarvittava on otettu toteutuksen suunnittelussa huomioon, ja ettei vaatimusten välillä ole ristiriitoja. Validoinnin menetelmiä ovat muun muassa dokumenttien katselmoinnit [BrI02,s. 210–213].

Avoimessa ohjelmistokehityksessä vaatimuksia harvoin kirjataan erilliseen spesifiointidokumenttiin, joten aivan vastaavanlaiset menetelmät eivät ole siellä käytössä. Avoimen kehityksen puolella ei usein ole havaittavissa minkäänlaista systemaattista vaatimusten verifioinnin ja validoinnin hallintaa [BaM02]. Se, kuinka vaatimukset kuvataan, validoidaan ja liitetään osaksi järjestelmän muita kuvauksia, on ennemminkin projektikohtaista kuin selkeästi määritetty tavoite.

## 2.4 Vaatimusten dokumentointi

Perinteisessä ohjelmistotekniikassa vaatimusmäärittelyn jokaisessa vaiheessa tuotetaan dokumentti, jota käytetään prosessin seuraavassa vaiheessa vahvasti hyväksi. Avoimessa kehityksessä vaatimukset ovat dokumentoitu eri muodoissa yhteisön Web-sivuille [BeP01]. Vaatimuksia voidaan löytää teknisistä kuvauksista, keskustelufoorumeissa käydyistä keskusteluista, yhteisön vision kuvaavista kertomuksista, usein kysytyistä kysymyksistä tai lähestulkoon mistä tahansa muusta

kuvauksesta. Vaatimukset, niiden analysointi ja spesifiointi, saattavat kaikki hyvinkin löytyä samasta kuvauksesta [Scac02].

# 3 Konfiguraation hallinta

Konfiguraatiolla tarkoitetaan yleisesti monimutkaisten ohjelmistojärjestelmien kehityksen ja elinkaaren hallintaa. Konfiguraation hallinnan avulla pyritään organisoimaan komponentteihin liittyviä muutoksia sekä varmistamaan eheyden säilyminen. Konfiguraatio on kokonaisuus, joka koostuu sen hetkisistä komponenteista. Suurin osa konfiguraationhallinnasta muodostuu dokumentaation ja lähdekoodien versionhistorian hallinnasta [NiJ06].

Versionhallinta on eräs ohjelmiston konfiguraatiohallinnan tärkeimmistä tehtäväalueista. Versio on tietty kokonaisuus järjestelmästä. Versio muuttuu, kun komponentteihin tehdään muutoksia. Versioihin tehtyjä muutoksia hallinnoidaan ja voidaan seurata versiohistorian välityksellä [NiJ06].

## 3.1 Version hallinta

Useimmissa avoimen ohjelmistokehityksen projekteissa käytetään muutosten hallinnan työkaluna CVS (Concurrent Versions System) [Cvs08] ohjelmaa. Ohjelma perustuu avoimeen lähdekoodiin ja noudattaa asiakas-palvelin arkkitehtuuria, ollen siten käytettävissä kaikkialla Internetin välityksellä. Projektin kaikkia versioita säilytetään versionhallintatyökalulla hallinnoitavassa versioarkistossa. Työkalu tarjoaa mekanismeja, joilla voidaan ehkäistä liiallisen tallennustilan käyttöä. Esimerkiksi kustakin komponentin revisiosta tallennetaan ainoastaan muutostieto johonkin toiseen revisioon [AsB02].

Useimmille kehittäjille myönnetään yleensä oikeus lisätä versioita arkistoon. Versioita ei lähes koskaan käytetä version palauttamiseksi aikaisempaan versioon. Sen sijaan niitä

käytetään historiatiedon selailuun. Versioista voidaan seurata, kuinka yksittäinen tiedosto on muuttunut aikojen kuluessa tai verrata eri versioita keskenään. Versioiden avulla saadaan jäljitettyä niihin kunakin ajanhetkenä tehdyt muutokset [AsB02].

## 3.2 Rakentamisen (engl. build) hallinta

Avoimessa ohjelmistokehityksessä yhteisön jäsenet työskentelevät rinnakkain. Tämän vuoksi jokaisella kehittäjällä on oma työalue, joka eristää työskentelyn muiden työskentelystä. Työalueet ovat yhteydessä projektin versioarkistoon, jonka kautta kehittäjät jakavat tekemänsä muutokset muiden jäsenien käytettäväksi [AsB02].

Tehdyn muutoksen jälkeen ohjelmistokehittäjän on varmistettava, että muutetut osat ovat yhteensopivia järjestelmän muiden osien kanssa. Toisin sanoen muutoksen tekijän on todennettava, että ohjelma rakentuu oikein vielä toteutettujen muutosten jälkeenkin. Ohjelman rakentaminen veisi paljon aikaa, jos ohjelman eri osat pitäisi joka kerta hakea rakennuksen yhteydessä fyysisesti eri sijainneista. Tämän vuoksi paikallisella työalueella on kaikki ohjelman rakentamisen kannalta tarvittavat tiedostot. Lisäksi tiedostojen löytyminen omalta paikalliselta työalueelta mahdollistaa työskentelyn off-line tilassa. Käyttäjän ei tarvitse olla jatkuvassa yhteydessä versioarkistoon [AsB02].

## 3.3 Konfiguraatio kokoelma

Yleensä vain ohjelmiston viimeisintä versiota ylläpidetään. Tämän vuoksi muiden konfiguraatioiden ei periaatteessa tarvitse olla saatavilla. Jos kahta erillistä versiota ylläpidetään samanaikaisesti, ajetaan niitä tavallisesti erillisinä projekteina [AsB02]. Tällainen tilanne saattaa esiintyä silloin, kun toinen versio on puhdas kehitys versio ja toinen on eriytetty versioksi, johon uusien ominaisuuksien toteuttaminen ei enää ole sallittua. Ainoastaan havaitut virheet korjataan.

## 3.4 Työtilan hallinta

Avoimessa ohjelmistokehityksessä paljon käytetty version hallinnan työkalu CVS tukee omilla ominaisuuksillaan projektien rinnakkaista tapaa työskennellä. Se mahdollistaa ohjelmistokehittäjälle oman erillisen työtilan luomisen, synkronoinnin versioarkiston kanssa sekä toimenpiteet muutoksien lisäämiseksi versioarkistoon. Työkalun johdosta kehittäjän ei tarvitse erikseen huolehtia tiedostojen siirrosta Internetin ylitse. CVS hoitaa tämän kehittäjän puolesta [AsB02].

## 3.5 Rinnakkaisuuden hallinta

Projektin jäsenten rinnakkaista työskentelyä on hallittava jotenkin, jotta ristiriidoilta vältyttäisiin. Tarvitaan toimintamalli, jolla hallitaan komponenttien samanaikaista kehitystä [NiJ06]. CSV työkalua hyödyntävät avoimet projektit käyttävät yhteistyöstrategiana niin sanottua optimistista rinnakkaisuuden hallintaa, jossa komponenttien rinnakkainen kehitys sallitaan. Versionhallinnan työkalulla on omat menetelmänsä tunnistaa, jos yksittäiseen tiedostoon on tehty rinnakkaisia muutoksia. Tällaisessa tapauksessa se pakottaa viimeisimpänä muutoksensa versioarkistoon liittävän ohjelmistokehittäjän tarkistamaan mahdolliset konfliktit. Konflikteja tapahtuu yleensä harvoin, vaikka kehitys on nopeatahtista ja kehittäjiä lukuisia. Jos ristiriitoja kuitenkin esiintyy, kommunikoivat tekijät keskenään ratkaistakseen syntyneen ongelman. Projektin sähköpostilistoilla ja uutisryhmissä lisätään yleensä tietoisuutta siitä, mitä ohjelmistokehityksessä on meneillään ja pyritään näin vähentämään mahdollisia konflikteja [AsB02].

## 3.6 Muutoksen hallinta

Perinteisissä ohjelmistoprojekteissa muutoksia hallinnoidaan muutospyynnöillä. Ehdotetut muutokset arvioidaan ja hyväksytyt muutosehdotukset annetaan ohjelmistokehittäjien toteutettaviksi. Avoimessa ohjelmistokehityksessä ei juuri esiinny

samanlaista selkeää muutosten hallintaa. Periaatteessa kuka tahansa voi ehdottaa muutosta ohjelmistoon. Käytännössä muutos on usein jo toteutettu ennen kuin muutosehdotusta edes tehdään. Jonkinlaista muutosehdotusten priorisointia saattaa esiintyä [McO06], mutta muutoksia ei yleensä voida suoraan ohjata kehittäjille tehtäväksi. Usein jokainen kehittäjä tekee sitä, minkä tuntee itse mielekkääksi [AsB02].

### 3.7 Julkaisujen hallinta

Harva avoimen ohjelmistokehityksen projekti julkaisee uusia versiota tuotteestaan perinteisen ohjelmistotekniikan keinoin, jossa tarvittavat ohjelmistokomponentit paketoidaan toimitettavaksi ohjelmistopaketiksi. Projektit eivät usein myöskään käytä ennalta sovittuja version julkistuspäiviä vaan julkaisujen ajoitus on yleensä hyvin sattumanvaraista. Lisäksi projektit julkaisevat niin sanottuja sisäisiä julkaisuja, joiden pyrkimyksenä on jäädyttää kyseiseen versioon tehtävät muutokset. Tällöin uusien yhtenäisyyttä rikkovien muutoksien toteuttaminen ei siihen versioon ole enää suotavaa, joskaan ei täysin kiellettyjä [AsB02].

## 4 Yhteenveto

Avoimen kehityksen vaatimusten hallinta eroaa melkoisen paljon perinteisen ohjelmistotekniikan välineistä hallita ohjelmistoon tai sen tuottamiseen kohdistuvia vaatimuksia. Avoimen kehityksen puolella ei ole samallalailla vakiintuneita prosesseja kuin perinteisen ohjelmistotekniikan puolella. Voisi siten helposti ajatella, että avoimessa kehityksessä vaatimusmäärittelyllä, tai sen prosesseilla, ei ole yhtä merkittävää asemaa kuin perinteisessä ohjelmistotekniikassa. Ehkä johtuen avoimen kehityksen luonteesta, jossa vaatimukset tulevat pääasiassa kehittäjiltä itseltään. Tämä ei kuitenkaan ole välttämättä huono asia, sillä avoimen kehityksen yhteisöillä on yleensä omanlainen, yhteisön keskuudessa, hyvin sisäistetty tapa toimia, joka tuottaa tulosta.

Konfiguraation hallinnassa perinteisen ja avoimen kehityksen välillä ei ole juurikaan havaittavissa eroja. Avoimessa kehityksessä on perinteisiä vastaavat menetelmät käytössä rinnakkaisen työn ja version hallinnan tukemiseksi. Ainoa merkittävä ero on muutoksien hallinnassa. Tällä alueella avoimessa kehityksessä tuntuisi olevan kehitysvaraa. Kehitystä varmasti tapahtuukin tulevaisuudessa avoimen kehityksen nostaessa jatkuvasti suosiotaan.

# Lähteet


AsB02 U. Asklund, L. Bendix, A study of Configuration Management in Open Source Software Projects. *Software, IEE Proceedings*, 149, 1, 2002, 40-46

BaM02 Bart Massey, Where Do Open Source Requirements Come From (And What Should We Do About It)? The 2nd Workshop on Open Source Software Engineering, *ICSE*, 2002.

BeP01 Erik Berglund, Michael Pristley, Open-Source Documentation: In Search of User-Driven, Just-in-Time Writing. In Proc. 19th annual international conference on Computer documentation. *ACM Special Intrest Group for Design of Communication*, Santa Fe, New Mexico, 2001, 132-141.

BrI02 Ian K. Bray, An introduction to requirements engineering. Addison-Wesley, 2002.

Cvs08 CVS –versionhallintatyökalun kotisivu, http://ximbiot.com/cvs/wiki/. 2008

GSP03 Gasser, L., Scacchi, W, Penne, B., Sandusky, R., Understanding Continuous Design in OSS *Projects. Proc. 16th. Int. Conf. Software Systems Engineering and their Applications*, Paris, 2003.

McO06 Owen G. McGrath, Balancing Act: Community and Local Requirements In an Open Source Development Process. *Proceedings of the 34th annual ACM SIGUCCS conference on User services*, 2006, 240–244.

NiJ06 Niemelä, Jouni, Ohjelmistojen konfiguraationhallinta. *Tietotekniikan luentomoniste*. Jyväskylän Yliopisto, Tietotekniikan laitos, Jyväskylä, 2006.

Scac02 Scacchi, W., Understanding the Requirements for Developing Open Source Software Systems. *Software, IEE Proceedings*, 149, 1, 2002, 24–39

SJN06 Scacchi, W., Jensen, C., Noll, J., Elliot, M., Multi-Modal Modeling, Analysis and Validation of Open Source Software Development Processes. *Intern J. Internet Technology and Web Engineering*, 1(3), 2006, 49–63.