TAMPEREEN TEKNILLINEN KORKEAKOULU
Tietotekniikan osasto

RAMI LEHTI

OPEN SOURCE -OHJELMISTOKEHITYS

Diplomityö

Aihe hyväksytty osastoneuvoston kokouksessa
10.3.1999
Tarkastajat: Professori Ilkka Haikala (TTKK)
Professori Hannu-Matti Järvinen (TTKK)

ALKULAUSE

Haluan kiittää kaikkia Open Source -ohjelmistokehitystä tekeviä ihmisiä ympäri maailmaa. Ilman heitä meillä ei olisi useita erinomaisia ohjelmia eikä GNU/Linux käyttöjärjestelmää. Lisäksi haluan kiittää seuraavia tämän diplomityön tekemisessä avustaneita ja ohjanneita henkilöitä: Anu-Marjut Lehti, Ilkka Haikala, Hannu-Matti Järvinen ja Kari Lehti. Haluaisin myös kiittää kaikkia AIDE-projektiin osallistuneita, erityisesti Pablo Virolaista.

Tampereella 30. syyskuuta 1999

Rami Lehti
Opiskelijankatu 4 E 247
33720 Tampere

Sisältö

TAMPEREEN TEKNILLINEN KORKEAKOULU

Tietotekniikan osasto

Ohjelmistotekniikka

LEHTI, RAMI: Open Source -ohjelmistokehitys

Diplomityö, 55 s.

Tarkastaja: prof. Ilkka Haikala

Lokakuu 1999

Avainsanat: Open Source, Ohjelmistokehitys

Open Source -ohjelmistot ovat kehittyneet harrastelijoiden pikkuohjelmista suuriksi ja vakavasti otettaviksi käyttöjärjestelmäympäristöiksi. Kuitenkaan ne eivät ole menettäneet aikojen saatossa tiettyä ”hakkerimaista“ mainettaan. Toisaalta jotkut yritykset ovat viime aikoina ottaneet Open Source -ohjelmistokehityksen osaksi yrityskulttuuriaan.

Vaikka Open Source -ohjelmistokehitystä on tehty jo usean vuoden ajan, ei siitä ole tarjolla kovinkaan paljon kirjallista materiaalia. Tietoa on kuitenkin saatavissa digitaalisessa muodossa runsaasti eri puolilta Internet-verkkoa.

Kaiken markkinahumun keskellä on saattanut syntyä käsitys, että Open Source -ohjelmistokehitys on jokin taikakeino, jolla voi herättää kuolleen ohjelmistoprojektin eloon. Näin ei kuitenkaan ole. Open Source -ohjelmistokehitykselle asetetaan odotuksia, jotka saattavat olla perusteettoman korkeita.

Diplomityön tarkoituksena on kerätä yksiin kansiin hajallaan olevaa tietoa, tutkia mitä etuja Open Source -ohjelmistokehityksellä saavutetaan ja selvittää yrityksien mahdollisuuksia käyttää Open Source -ohjelmistokehitystä ja Open Source -ohjelmistoja.

TAMPERE UNIVERSITY OF TECHNOLOGY

Department of Computer Science

Software Engineering

LEHTI, RAMI: Open Source Software Development

Master of Science Thesis, 55 pages

Examiner: prof. Ilkka Haikala

October 1999

Keywords: Open Source, Software Development

Open Source software has developed from small programs originally written by single hackers into significant operating system environments. Through the years, they have not lost their "hacker-glamour", however. On the other hand, some enterprises have recently adopted Open Source software development as a part of their business culture.

Even though Open Source software development has been practiced for several years already, the literature on the subject is limited. Nonetheless, there is a decent amount of information to be found on the Internet.

All the marketing buzz may have provoked the false impression that Open Source software development is a sort of "magic powder" which can revive a dead software project. Open Source software development may thus sometimes have unreasonably high expectations attached to it.

The aim of this thesis is to bring together scattered information, to study the advantages of Open Source software develpoment and to examine the ways in which companies can benefit from Open Source software development and Open Source software.

LYHENTEET

AIDE	Advanced Intrusion Detection Environment; Open Source -eheystarkistinohjelmisto.
BSD	Berkeley Standard Distribution; Ensimmäinen Unix-käyttöjärjestelmälevitys, joka lisensoitiin alkuperäisellä BSD-lisenssillä.
CVS	Concurrent Versioning System; Versionhallintaohjelmisto, joka mahdollistaa yhtäaikaisen kehityksen eri puolilla maailmaa.
FSF	Free Sofware Foundation; GNU-projektin suojelijajärjestö.
GCC	GNU C-Compiler; GNU-projektin tuottama monikielikääntäjä.
GGI	General Graphics Interface; GGI-projektin luoma grafiikkaohjelmointirajapinta.
GIMP	General Image Manipulation Program; GPL-lisensoitu kuvankäsittelyohjelma.
GNOME	GNU Object Model Environment; GPL-lisensoitu työpöytäsovelluskokoelma.
GNU	GNU is Not Unix; GNU-projektin tarkoituksena on luoda täysin vapaa käyttöjärjestelmäympäristö.
GPL	GNU General Public License; GNU-projektin käyttämä lisenssi.
GTK	GNU ToolKit; GPL-lisensoitu ikkunointityökaluohjelmointikirjasto.

HTML	HyperText Markup Language; Hyperteksti dokumenttien kuvaamiseen käytetty ohjelmointikieli.
HTTP	HyperText Transfer Protocol; Protokolla, joka on suunniteltu hypertekstin välittämiseen.
HTTPD	HyperText Transfer Protocol Daemon; HTML sivuja HTTP:llä jakava palvelinohjelma.
IRC	Internet Relay Chat; Tekstipohjainen keskustelu ohjelmisto.
LGPL	Lesser General Public License; Toinen GNU-projektin lisensseistä. Alunperin suunniteltu ohjelmistokirjastoja varten.
NPL	Netscape Public License; Netscapen käyttämä Open Source -lisenssi.
MPL	Mozilla Public License; Toinen Netscapen kehittämä Open Source -lisenssi. Hyvin NPL:n kaltainen.

1 JOHDANTO

Open Source -ohjelmistot yleisesti tarkoittavat ohjelmistoja, joiden lähdekoodi on saatavilla ilmaiseksi ja siihen saa tehdä muutoksia. Tämä on varsin vallankumouksellinen ajatus nykyisessä ohjelmistoteollisuudessa. Yrityksien lisenssit ovat erittäin tarkkoja siitä, mitä ohjelmiston binäärimuodolla tehdään. Lähdekoodia useimmista ohjelmistoista ei pääse edes katsomaan ilman, että allekirjoittaa salassapitosopimuksen.

Lähdekoodin levitys ohjelmistojen mukana sai alkunsa yliopistoista ja erilaisista tutkimusohjelmista 1970-luvulla. Lähdekoodin ilmainen levitys juontaa juurensa samasta ympäristöstä. 1970-luvun lopussa ja 1980-luvun alussa yliopistoissa ohjelmoijille maksettiin ohjelmoimisesta eikä heidän tuottamastaan lähdekoodista, joten he saattoivat levittää sitä suhteellisen vapaasti. Toisaalta lähdekoodi oli usein osa tutkimusprojektia. Lähdekoodin katsottiin olevan osa näiden tutkimusprojektien tuloksista, joten sekin julkaistiin samalla kuin muut tulokset.

MIT:ssä (Massachusetts Institute of Technology) oli erittäin aktiivinen hakkerijoukko, jolla oli tapana jakaa lähdekoodia vapaasti keskenään. GNU-projektin perustaja Richard Stallman oli osa tätä joukkoa. Tämä joukko ja monet muut hakkeriyhteisöt ovat edistäneet huomattavasti lähdekoodin vapaata levitystä.

Hakkerilla tarkoitetaan tässä diplomityössä ohjelmoinnista pitävää henkilöä, joka käyttää taitojaan nokkelasti. Tämä on Eric Stallmanin määritelmä. Vastaava määritelmä löytyy myös kirjasta New Hacker's Dictionary [1]. Sen määritelmän mukaan hakkeri on henkilö, joka pitää ohjelmoitavan järjestel-

män yksityiskohtien tutkimisesta ja sen mahdollisuuksien laajentamisesta, päinvastoin kuin tavalliset käyttäjät, jotka haluavat tietää vain sen, mikä on ehdottomasti välttämätöntä tietää.

Internet-verkon räjähdysmäisen kasvun tuloksena maantieteelliset välimatkat eivät enää ole merkitseviä lähdekoodin leviämisen esteenä. Pienissäkin projekteissa voi olla mukana ihmisiä monelta mantereelta.

Open Source -ohjelmistot ovat viime aikoina nousseet yhä tärkeämpään asemaan tietotekniikassa syrjäyttäen shareware-ohjelmistoja. Tätä kehitystä on huomattavasti kiihdyttänyt Internet-verkon raju laajeneminen. Nämä projektit ovat kuitenkin edelleen pääasiassa harrastuspohjalla. Tästä huolimatta niiden tuotokset ovat usein vähintään yhtä hyviä ellei parempia kuin niiden kaupalliset vastineet.

Tämä tapa tehdä ohjelmistoja on suhteellisen uusi ja varsin vähän tunnettu suuren yleisön keskuudessa. GNU/Linux-käyttöjärjestelmä on kuitenkin tuonut Open Source -termin ja Open Source -ohjelmistot yleensä suuremman yleisön tietoisuuteen. Tietääkseni mitään tarkempaa tutkimusta Open Source -projekteista ei kuitenkaan ole tehty.

Open Source -mallia voidaan soveltaa sekä ”todellisissa“ Open Source -projekteissa että useanlaisissa suurten ja keskisuurten ohjelmistotalojen “normaaleissa” projekteissa. Sen käytöllä voidaan tehostaa koodin uudelleenkäyttöä ja avoimuutta.

Open Source -ohjelmistokehitystä ei ole vakavasti tutkittu, koska sitä on usein pidetty vain harrastelijoiden puuhailuna. Open Source -menetelmä on kuitenkin voimakas ohjelmistokehitysmalli, jota pitäisi tutkia tarkemmin.

Diplomityön tarkoituksena on luoda yleiskatsaus Open Source -ohjelmistokehitykseen ja siihen vaikuttaviin asioihin. Open Source -ohjelmistokehitystä on tehty jo monen vuoden ajan, mutta miltei kaikki siitä saatava tieto on hajallaan ympäri verkkoa. Tähän diplomityöhön on kerätty tuota tietoa.

Luvussa 2 tarkastellaan erilaisia Open Source -määritelmiä ja sitä miten ne ovat vaikuttaneet Open Source -ohjelmistojen levinneisyyteen. Internetin eri keskusteluryhmissä on käyty välillä kiivastakin keskustelua siitä, mitä tarkoitetaan Open Source -ohjelmistoilla ja pitäisikö niitä edes kutsua Open Source -ohjelmistoiksi.

Luvussa 3 käsitellään Open Source -ohjelmistojen lisenssejä. Nämä lisenssit poikkeavat huomattavasti yleisesti käytössä olevista kaupallisista lisensseistä. Ne vaativat omat erityispiirteensä täyttääkseen luvussa 2 esitetyt vaatimukset. Open Source -lisenssit ovat erittäin tärkeitä, koska ohjelmistojen lähdekoodi on kaikkien saatavilla.

Luvussa 4 keskitytään Open Source -ohjelmistokehityksen erikoispiirteisiin. Open Source -projektit vaativat erityisiä menetelmiä niiden johtamisessa ja erityisiä työkaluja projektin koordinoimiseen. Tämä luku pyrkii esittämään erilaisia ratkaisumalleja ja työkaluja, joita yleisesti käytetään Open Source -projekteissa.

Luvussa 5 tarkastellaan joitakin todellisia projekteja. Ne ovat eri kokoisia normaaleja Open Source -projekteja. Yksi niistä on minun aloittamani testiprojekti. Tällä projektilla on tarkoitus testata Open Source -ohjelmistokehitysmallia.

Luvussa 6 esitetään erilaisia tapoja, miten Open Source -ohjelmistoke-

hitystä voidaan soveltaa yritysmaailmassa. Open Source -ohjelmistot ovat perinteisesti olleet harrastuspohjaisia tuotoksia, mutta viime aikoina Open Source -mallia on otettu käyttöön myös kaupallista ohjelmistokehitystä tekevissä yrityksissä. Tässä luvussa kerrotaan myös Open Source -ajattelun vaikutuksista normaaliin yritystoimintaan.

Luvussa 7 tarkastellaan Open Source -ohjelmistokehityksestä saatavia etuja ja haittoja. Tässä luvussa katsotaan Open Source -ohjelmistoja myös asiakkaan näkökulmasta. Tässä asiakas tarkoittaa ohjelmiston käyttäjää.

2 MÄÄRITELMÄT

2.1 Open Source Definition

Yritykset eivät olleet innostuneita vapaisiin ohjelmistoihin 1990-luvun alkupuolella pääasiassa Richard Stallmanin ohjelmistokehittäjien vapauksien painotuksen vuoksi. Niinpä pieni ryhmä ihmisiä päätti muuttaa tämän käsityksen keksimällä uuden termin, joka laajemmin kuvaa vapaita ohjelmistoja. Tähän ryhmään kuuluivat Todd Anderson, Chris Peterson, John Hall, Larry Augustin, Sam Ockman ja Eric Raymond. He perustivat Open Source Initiative -järjestön, jonka tarkoitus on edistää Open Source -ohjelmistojen käyttöä ja kehitystä. Open Source Initiative järjestön toisena tehtävänä on antaa sertifikaatteja ohjelmistolisensseille. Tällaisia sertifioituja lisenssejä ovat mm. GPL, BSD-lisenssi ja MPL.

Termi "Open Source" on Chris Petersonin keksintöä. Termi syntyi edellämainittujen ihmisten pitämässä aivoriihessä. Vakiintunutta suomennosta kyseisellä termillä ei ole, eikä sille ole löydetty hyvää suomenkielistä vastinetta. Tässä diplomityössä käytetään alkuperäistä englanninkielistä muotoa.

Bruce Perens kehitti Open Source -määritelmän [2] Debian GNU/Linux käyttöjärjestelmälevitykseen hyväksyttävien ohjelmistojen lisenssirajoitusten (Debian Free Software Guidelines) pohjalta. Debian Free Software Guidelines dokumentti syntyi kuukauden kestäneen sähköpostikeskustelun aikana, jossa oli mukana monia Debian distribuution kehittäjiä. Bruce Perens jalosti Debian Free Software Guidelines dokumentin Open Source -määritelmäksi Eric Raymondin pyynnöstä. Ainoat Debian Free Software Guidelines doku-

menttiin tehdyt muutokset olivat Debianiin liittyvien viittausten poisto.

Eric Raymond on erittäin aktiivisesti toiminut Open Source -määritelmän kannattajana ja puolestapuhujana. Häntä käytetään usein viitteenä, kun alan lehdissä puhutaan Open Source -ohjelmista. Hän on kohdistanut toimintansa erityisesti yrityksiin. Häntä on usein pidetty katalyyttinä Open Source -ohjelmistojen leviämiselle. Eric Raymondin toiminta on ollut tärkeää, mutta Open Source -liikettä ei voida kuitenkaan pitää hänen ansionaan. Hän on saanut monet perinteistä hakkerikulttuuria kannattavat henkilöt vihaisiksi markkinahenkisillä puheillaan.

Määritelmä asettaa ohjelmistolisensseille tietyt rajat, jotta niitä voidaan kutsua Open Source -ohjelmistoiksi. Seuraavaksi Bruce Perensin Open Source -määritelmän epävirallinen suomennos.

1. Vapaa uudelleenlevitys.

   Ohjelmaa pitää pystyä levittämään edelleen yksin tai muiden ohjelmien kanssa. Minkäänlaisia rojalteja ei saa vaatia.

2. Lähdekoodi.

   Ohjelmapaketin tulee sisältää lähdekoodi ja sen tulee sallia lähdekoodin ja binäärisen muodon uudelleenlevitys.

3. Polveutuvat tuotteet.

   Muutoksien tekeminen ohjelmaan ja polveutuvat tuotteet tulee sallia ja niitä tulee voida levittää saman lisenssin alaisena.

4. Alkuperäisen kirjoittajan oikeudet.

Alkuperäinen kirjoittaja saa estää muutetun lähdekoodin levittämisen. Kuitenkin tulee sallia erillisten muutospakettien levittäminen. Muutospaketeilla tarkoitetaan ohjelmaa käännösaikana muuttavia paketteja. Lisenssi saa vaatia, että polveutuvat tuotteet on nimetty eri tavalla tai että niiden versiointi on suoritettu eri tavalla kuin alkuperäisen ohjelmiston versiointi.

5. Ei syrjintää tiettyjä henkilöitä tai ryhmää kohtaan.

   Ohjelmiston lisenssi ei saa rajoittaa ohjelmiston käyttöä tai kehitystä tietyn henkilön tai ryhmän osalta.

6. Ei syrjintää tiettyjä aloja kohtaan.

   Lisenssi ei saa myöskään syrjiä mitään tiettyä alaa. Esimerkiksi lisenssi ei saa kieltää ohjelmiston käyttämistä liiketoiminnassa tai geenitutkimuksessa.

7. Lisenssin levitys.

   Ohjelmaan liitettyjä oikeuksia tulee voida soveltaa kaikkiin henkilöihin, joille ohjelmisto levitetään, ilman erillistä lisenssiä.

8. Lisenssi ei saa olla liitetty mihinkään tuotteeseen.

   Lisenssiin liitetyt oikeudet eivät saa riippua siitä, onko ohjelmisto liitetty johonkin laajempaan kokonaisuuteen vai ei. Jos ohjelma otetaan erilleen tällaisesta laajemmasta kokonaisuudesta, tulee kaikilla osapuolilla olla samat oikeudet kuin alkuperäisen levityksen yhteydessä.

9. Lisenssi ei saa rajoittaa muita ohjelmistoja.

   Lisenssi ei saa asettaa rajoituksia muille ohjelmille, joita levitetään ohjelmiston kanssa. Esimerkiksi lisenssi ei saa vaatia, että kaikki ohjelmat,

jotka levitetään samalla medialla, ovat Open Source -ohjelmistoja.

10. Yhdenmukaiset lisenssit ja sertifiointi.

    Open Source -tuotemerkkiä saavat käyttää vain sertifioidut lisenssit tai ohjelmistot, jotka ovat yleisomaisuutta.

Nämä ehdot täyttäviä ohjelmistolisenssejä voidaan sanoa Open Source -lisensseiksi. Nämä ehdot rajaavat miltei kaikki nykyiset kaupalliset ohjelmistolisenssit pois heti ensimmäisellä kohdallaan tai viimeistään toisella. Nämä ovat varsin kiusallisia vaatimuksia yrityksille, jotka haluavat tehdä Open Source -ohjelmistokehitystä, mutta joiden yritystoiminta nojaa siihen olettamukseen, että lähdekoodi on suljettu. Asiakkaan näkökulmasta nämä ehdot ovat erittäin mieluiset, koska ne sallivat ohjelmiston muuttamisen omiin tarpeisiin ja ne edellyttävät, että ohjelmisto on saatavilla ilmaiseksi.

Kaikki perinteiset vapaat ohjelmistolisenssit täyttävät tämän määritelmän vaatimukset suoraan. Ne on haluttu päästää läpi suoraan ilman muutoksia. Määritelmään on otettu useita perinteisissä lisensseissä olleita perusvaatimuksia, kuten vapaa uudelleenlevitys ja polveutuvien tuotteiden salliminen.

2.2 Vapaat ohjelmistot

Ennen Open Source -termin keksimistä, oli olemassa pieni, mutta aktiivinen ryhmä ohjelmoijia, jotka kehittivät ohjelmistoja vapaaseen levitykseen. He eivät pitäneet itsestään kovinkaan suurta melua, toisin kuin Eric Raymond. Ryhmittymä keskittyi pääasiallisesti GPL-lisenssillä tehtyihin ohjel-

miin ja GNU-projektiin. He kutsuvat näitä ohjelmia vapaiksi ohjelmistoiksi (Free Software). GNU-projektin kanssa on perustettu Free Software Foundation (FSF). Sen tarkoituksena on edistää GNU-projektia ja yleisesti vapaiden ohjelmistojen käyttöä.

GNU-projektin perustaja Richard Stallman pyrki luomaan uuden hakkeriympäristön, jossa ohjelmoinnista pitävät henkilöt voisivat jakaa lähdekoodia vapaasti ilman pelkoa siitä, että myöhemmin kaupalliset yritykset käyttäisivät hyödykseen tuota koodia kaupallisissa tuotteissaan. Tähän tarpeeseen syntyi GPL-lisenssi, josta kerrotaan tarkemmin seuraavassa luvussa. Lisenssin tarkoituksena oli pitää ohjelmistot vapaana ja kaikkien käytettävissä. Tämä onkin ollut hyvin suosittu ajattelutapa useiden hakkereiden keskuudessa.

GNU-projektin tavoitteena on luoda vapaista ohjelmistoista kokonainen Unix-käyttöjärjestelmäympäristö. Tämä projekti osoittautui valtavaksi haasteeksi käytössä olevilla resursseilla. Tuohon aikaan ei ollut vielä olemassa ensimmäistäkään vapaata versiota Unixin perustyökaluista. Koska kehitys oli aluksi hidasta, eikä Internet ollut vielä kovinkaan laajalle levinnyt, ei GNU-projektia otettu vakavasti. Kun Linus Torvalds myöhemmin kehitti Linux käyttöjärjestelmäytimensä, oli tuon projektin tuottamille monille palasille alusta, jolla niitä voitiin käyttää. Yhtäkkiä oli käytettävissä täysin vapaa käyttöjärjestelmä kaikkine oheisohjelmistoineen. Tämä aiheutti räjähdysmäisen kasvun GPL-lisensoitujen ohjelmistojen ja yleisesti Open Source -ohjelmistojen kehityksessä. Oli olemassa konkreettista näyttöä siitä, että lähdekoodin vapaalla levityksellä voitiin tuottaa ohjelmistoja.

GNU-projekti oli tuottanut miltei kaikki käyttöjärjestelmässä tarvitta-

vat ohjelmat, kun Linux-käyttöjärjestelmäydin julkistettiin. GNU-projektilta puuttui tuossa vaiheessa täydellisestä GNU-järjestelmästä ainoastaan toimiva käyttöjärjestelmäydin. GNU-projektin modulaarinen käyttöjärjestelmäydin HURD oli tuossa vaiheessa vasta alkutekijöissään.

FSF:n mukaan vapaat ohjelmistot määritellään ohjelmistoiksi, joiden lisensointi antaa seuraavat vapaudet. Tämä määritelmä on suora käännös FSF:n seittisivuilta [3].

- Vapaus ajaa ohjelmaa millä tahansa tarkoituksella (vapaus 0).
- Vapaus tutkia kuinka ohjelma toimii ja muuttaa sen toimintaa omiin tarkoituksiin paremmin sopivaksi (vapaus 1).
- Vapaus uudelleenlevittää kopioita, jotta voi auttaa lähimmäisiään (vapaus 2).
- Vapaus parantaa ohjelmaa ja julkistaa muutokset, jotta yhteisö hyötyisi julkistuksesta (vapaus 3).

Tämä määritelmä on pääkohdiltaan samantapainen kuin Open Source -määritelmä. Se ei kuitenkaan ole koskaan saanut samanlaista kannatusta yritysmaailmassa. FSF:n tavoitteena ei ollutkaan markkinoida tätä määritelmää yrityksille, vaan heidän kaltaisilleen hakkereille. Tässä FSF onkin onnistunut varsin hyvin. GPL ja LGPL ovat suosituimmat lisenssit Open Source -projekteissa.

Koska Open Source -määritelmä on löyhempi kuin GPL-lisenssi, ovat monet sitä mieltä, että Eric Raymondin toiminta on monin tavoin vesittänyt GNU-projektin tavoitteet. Tämä on aiheuttanut eripuraa GNU-projektin

kannattajien ja Open Source -ajattelun kannattajien välillä. On selvää, että ilman Bruce Perensin määritelmää ja Eric Raymondin jatkuvaa työtä Open Source -ohjelmistojen puolestapuhujana, Open Source -ohjelmistot eivät olisi niin suuressa suosiossa kaupallisten ohjelmistotalojen keskuudessa.

3 LISENSSIT

3.1 Lisenssien tarkoitus

Lisenssien tarkoitus ohjelmistotekniikassa on rajata tai suojata ohjelmiston käyttöä ja levitystä. Kun puhutaan Open Source -ohjelmistoista, on nämä periaatteet käännetty päälaelleen eli lisenssit on nimenomaan suunniteltu ohjelmiston levityksen edistämiseksi. Koska ohjelman koko lähdekoodi on kaikkien saatavilla, ovat lisenssit tärkeässä asemassa kehittäjän oikeuksien turvaamisessa.

Open Source -ohjelmistot sekoitetaan usein julkisohjelmistoihin, joiden käyttöä ei ole millään tavalla rajattu. Vaikka Open Source -ohjelmistojen lähdekoodi on saatavilla, ovat ne silti tekijänoikeussuojan alaisia. Julkisohjelmissa on puolestaan luovuttu kaikista tekijänoikeuksista. Julkisohjelmista on tosin harvemmin lähdekoodi saatavilla.

On olemassa hyvinkin erilaisia näkemyksiä siitä, mitä halutaan suojata ja kuinka tarkasti. Eric Raymondin Open Source -määritelmä antaa kohtuullisen vapaat kädet määritellä lisenssi. Näiden lisenssien tiukkuus ja tarkoitukset vaihtelevat tiukimmista NPL:stä ja GPL:stä yleisomaisuuteen. Kuitenkin vain erittäin harvoin Open Source -ohjelmistot ovat täysin julkisia (ns. public domain). Seuraavissa kohdissa esitellään tärkeimpiä ja yleisimmin käytettyjä Open Source -lisenssejä.

Open Source -lisenssien yleisenä ongelmana on ohjelmistokehityksen haarautuminen. Lisenssit sallivat muutoksien tekemisen ohjelmistoon ja näiden muutoksien levittämisen. Näin ollen kuka tahansa voi lähteä kehittämään

Open Source -ohjelmistoa haluamaansa suuntaan, aiheuttaen haarautuman projektiin. Haarautumat vievät projektissa olevien ihmisten huomion muualle kehitystyöstä ja haarautuminen saattaa viedä mukanaan osan kehittäjistä. Kaiken kaikkiaan työteho projektissa laskee ja tuotoksien laatu heikkenee.

Ohjelmistokehityksen haarautuminen ei kuitenkaan ole ollut ongelmana kuin vain muutamissa projekteissa. Tällaisia projekteja ovat esimerkiksi BSD-pohjaiset vapaat Unix-kloonit. Yleensä projektissa työskentelevät henkilöt ovat päässeet jonkinlaiseen yhteisymmärrykseen siitä, mitä tehdään.

3.2 BSD-tyyliset lisenssit

BSD-tyyliset lisenssit ovat vanhimpia Open Source -lisenssejä. BSD-lisenssit saivat alkunsa Berkeley Standard Distribution Unixin lisenssistä [4]. Tämän tyylisiä lisenssejä ovat perinteisesti käyttäneet yritykset julkisissa projekteissa. Esimerkkinä tällaisesta projektista mainittakoon X-ikkunointijärjestelmäprojekti. Sitä kehittävä X Consortium on valinnut BSD-pohjaisen lisenssin sillä tarkoituksella, että olisi olemassa yksi yhteinen koodi, johon sitten erilliset X Consortiumin jäsenyritykset tekevät omat yksityiset laajennuksensa ja julkistavat ne sitten omina tuotteinaan. Näin on haluttu taata eri implementaatioiden yhteensopivuus.

Perusominaisuuksiltaan BSD-lisenssi on suhteellisen yksinkertainen. BSD-lisenssi sallii lähdekoodin uudelleenkäytön, kunhan BSD-lisenssi ja sen vaatima tekijänoikeusilmoitus löytyy dokumentaatiosta. Tämä on kuitenkin pidemmän päälle hankalaa, erityisesti, jos tuodaan yhteen useiden eri tahojen tekemiä BSD-lisenssin alaisia ohjelmia. Tällöin pitäisi kaikessa dokumen-

taatiossa olla kaikkien mahdollisten tekijänoikeuksien haltijoiden oma versio BSD-tyylisestä lisenssistä ja ilmoitus heidän tekjänoikeuksistaan. Tämä saattaa joissakin tapauksissa olla useita sivuja. Ongelma on varsin todellinen käyttöjärjestelmälevittäjille, joiden tuotteessa on koottu yhteen useita eri Open Source -ohjelmia.

Mitään yksittäistä BSD-lisenssiä ei ole, vaan jokainen projekti käyttää omaa versiotaan tästä lisenssistä. Usein ainoa lisenssistä muutetteva asia on kuitenkin vain tekijänoikeuden omistajaan liittyvät kohdat. Jossain tapauksissa on haluttu tuoda tekijänoikeuden osuus vieläkin selkeämmin esille ja vaaditaan, että myös binäärinen muoto ohjelmasta tuottaa tekijänoikeusilmoituksen. Tämä ehto on myös alkuperäisessä Berkeleyn BSD-lisenssissä. Myöhemmin kirjoitetussa X Consortiumin käyttämästä lisenssistä tuo vaatimus on poistettu ja tarkennettu mitä lähdekoodille saa tehdä. Tuo tarkennus ei kuitenkaan rajoita lähdekoodin käyttöä.

Joissakin BSD-tyylisissä lisensseissä vaaditaan, että myös kaikissa tuotetta mainostavissa teksteissä täytyy olla edellä mainittu tekijänoikeusilmoitus. Tämä vaatimus oli alkuperäisessä Berkeleyn BSD-lisenssissä, mutta heinäkuussa 1999 tämä vaatimus poistettiin lisenssistä. Jotkin lisenssit taas puolestaan eivät salli tekijänoikeuden omistajien nimien tai heidän edustamiensa instanssien nimien käyttöä mainostustarkoituksissa.

3.3 GNU-projektin lisenssit

Richard Stallman oli sitä mieltä, että BSD-tyyliset lisenssit eivät ole tarpeeksi tiukkoja lähdekoodin uudelleenkäytön suhteen. Koska BSD-lisenssit

sallivat lähdekoodin käytön, voi mikä tahansa yritys ottaa menestyneen ohjelmiston omakseen, eikä heidän tarvitse tuoda tekemiään muutoksia kenenkään muun tietoon. Tämä epäkohta oli Richard Stallmanilla päämotivaattori alkuperäisen GPL-lisenssin (General Public License) kirjoittamiseen [5].

GPL-lisenssin tarkoituksena on, että ohjelma ja kaikki siihen tehtävät muutokset ja jatkokehitelmät ovat julkisia ja että ne ovat tämän lisenssin alaisia. GPL-lisenssi estää GPL-lisensoidun ohjelmiston tai sen osan liittämisen ei-vapaaseen ohjelmistoon tai päinvastoin. Ei-vapaa ohjelmisto määritellään kohtuullisen tiukasti GPL-lisenssissä. Richard Stallmanin mukaan myös binääriversioiden linkittäminen GPL-lisensoituun ohjelmistoon tai päinvastoin on kiellettyä. GPL täyttää GNU-projektin vapaiden ohjelmistojen määritelmän ja sen lisäksi se estää ohjelmiston tekemisen ei-vapaaksi.

GPL-lisenssistä on myös olemassa hiukan löysempi versio nimeltä LGPL (Lesser General Public License) [6]. Tämä lisenssi sallii ohjelman käyttämisen kaupallisten ohjelmien yhteydessä. Alunperin tämä lisenssi oli tarkoitettu yleiskäyttöisten ohjelmointikirjastojen lisenssiksi, koska GPL-lisenssin alaisia ohjelmointikirjastoja ei voida käyttää kaupallisilla lisensseillä varustetuissa ohjelmissa. Nykyisin sitä käytetään myös tavallisten ohjelmien lisensointiin, kun halutaan, että ohjelmaa voidaan laajentaa eri tavoilla lisensoiduilla moduuleilla.

Molemmat lisenssit täyttävät Open Source -määritelmän vaatimukset selkeästi. Ne ovat suosituimpia lisenssejä harrastelijoiden keskuudessa. Yritykset ovat olleet hyvin varovaisia GPL-lisenssin käyttöönotossa, koska se vaatii kaikkien lisenssin alaiseen ohjelmistoon liitettyjen ohjelmien lisensoimista GPL:llä.

3.4 NPL ja MPL

NPL (Netscape Public License) on Netscape Communications Inc:n Eric Raymondin kanssa yhteistyössä luoma lisenssi, jonka tarkoituksena on tehdä mahdolliseksi alunperin suljetun ohjelmiston julkistaminen Open Source -projektina [7]. MPL (Mozilla Public License) on saman yrityksen kehittämä NPL:n kaltainen lisenssi

NPL on ensimmäinen kaupallinen lisenssi, joka toteuttaa Open Source -määritelmän vaatimukset. Alunperin NPL:n tarkoituksena oli toimia Netscapen julkistaman seittiselaimen lisenssinä yhdessä MPL:n kanssa. Myöhemmin Netscape yhdisti NPL lisenssin ja MPL:n. NPL ja MPL olivat aiemminkin periaatteellisesti hyvin lähellä toisiaan. Koska lisenssitekstin otsikkona on Mozilla Public License, käytetään tässä MPL-nimitystä.

MPL:n yleiset Open Source -määritelmän vaatimat osuudet ovat samanlaisia kuin muissakin Open Source -lisensseissä. MPL eroaa muista lisensseistä muutoksia käsitteleviltä kohdiltaan. MPL jakaa ohjelmiston ja siihen tehtävät muutokset kahteen eri tyyppiin. Nämä kaksi osaa ovat alkuperäinen koodi (Covered Code) ja jatkokehitysversio (Larger Work).

Alkuperäinen koodi ja siihen tehtävät muutokset vaaditaan julkaistaviksi MPL:n alaisena. Alkuperäisellä koodilla tarkoitetaan julkistetun ohjelmiston lähdekoodia. Tässä suhteessa MPL toimii samoilla tavoin kuin GPL-lisenssi. Julkaisun yhteydessä täytyy erikseen kertoa mitkä osat koodista ovat alkuperäistä koodia. Tämä estää sekaannuksien syntymistä, kun tehdään jatkokehitysversioita. Ohjelmiston ei siis tarvitse olla kokonaan MPL-lisenssissä puhuttua niin sanottua alkuperäistä koodia, vaan ohjelmistossa voi olla myös

alusta alkaen jatkokehitysversioksi katsottavia osia.

Jatkokehitysversio ja siihen tehtävät muutokset voidaan levittää millä tahansa lisenssillä. Kuitenkin alkuperäinen koodi, joka on osa jatkokehitysversiota, täytyy edelleen olla MPL:n alaista. Mikäli alkuperäiseen koodiin on tehty muutoksia, tulee myös niiden olla MPL:n alaisia. Tämä mahdollistaa kaupallisten lisämoduulien tekemisen erilaisten ohjelmointirajapintojen kautta.

Tämä kaksijakoinen suhtautuminen muutoksiin on erityisen kiinnostava yrityksille, jotka haluavat räätälöidä jonkin Open Source -ohjelmiston omiin tarkoituksiinsa. MPL-lisenssi onkin nähty siltana GPL-lisenssin vapauksiin vakavasti suhtautuvan katsontakannan ja kaupallisten lisenssien välillä.

Toinen mielenkiintoinen ominaisuus MPL-lisenssissä on sen suhtautuminen ohjelmistopatentteihin. Mikäli eri yritykset ottavat osaa kehitystyöhön, luovuttavat ne ohjelmistopatenttinsa kaikkien osapuolien käyttöön. Tämä ei ole ongelma Suomessa, jossa ohjelmistopatentteja ei myönnetä yhtä löysin perustein kuin Yhdysvalloissa. Yhdysvaltojen leväperäinen patenttikäytäntö ohjelmistojen suhteen on tämän kohdan takana.

3.5 Artistic License

Lisenssi on kehitetty sen ajatuksen pohjalta, että ohjelmistot voidaan rinnastaa taideteoksiin. Lisenssillä on haluttu jättää ohjelmiston alkuperäiselle kehittäjälle “taiteellinen” valta ohjelmiston kehityksessä [8]. Lähdekoodi on samalla pyritty pitämään suhteellisen avoimena. Tämä lisenssi on vaikutuk-

siltaan miltei samanlainen kuin LGPL.

Tässä lisenssissä on kuitenkin muutamia porsaanreikiä, jotka sallivat ohjelmistoon tehtyjen muutosten varjolla lisenssin vaihtamisen. Menemättä kovinkaan tarkasti yksityiskohtiin voidaan sanoa, että tietyistä teknisistä syistä voidaan tietyillä ehdoilla julkaista osa lähdekoodista jopa julkisomaisuutena. Monien mielestä Artistic License -lisenssi on tämän takia käyttökelvoton.

Nykyään monet alunperin Artistic License lisenssin alaiset ohjelmistot, ovat kaksoislisensoituja. Tämä tarkoittaa sitä, että käyttäjä voi valita joko GPL:n tai alkuperäisen Artistic License -lisenssin. Näin on lisensoitu mm. Perl-tulkki [9].

Artistic License ei ole lakimiesten kirjoittama ja tämän huomaa lisenssin tekstistä. Sen kirjoittaja, Larry Wall, ei pyrkinytkään sitä tehdessään täydelliseen tarkkuuteen, vaan hän pyrki esittämään omia mielipiteitään siitä, kuinka vapaita ohjelmistoja tulisi levittää. Myöhemmin tätä lisenssiä on käytetty monissa projekteissa. Edellämainitun tultua ilmi joutuivat useat projektit vaihtamaan lisensointiaan. Projekteissa, joissa kehittäjiä on ympäri maailmaa, ei ole helppoa vaihtaa lisenssiä, koska kaikkien koodia kirjoittaneiden pitää suostua lisenssimuutokseen. Tämä lisenssi on hyvä esimerkki siitä, mitä huonosti suunnitellut lisenssit aiheuttavat.

3.6 Lisenssin valinta

Lisenssiä valittaessa on oltava hyvin tarkkana, koska kullakin lisenssillä on hyvät ja huonot puolensa. Nämä kannattaa harkita tarkoin, ennen kuin te-

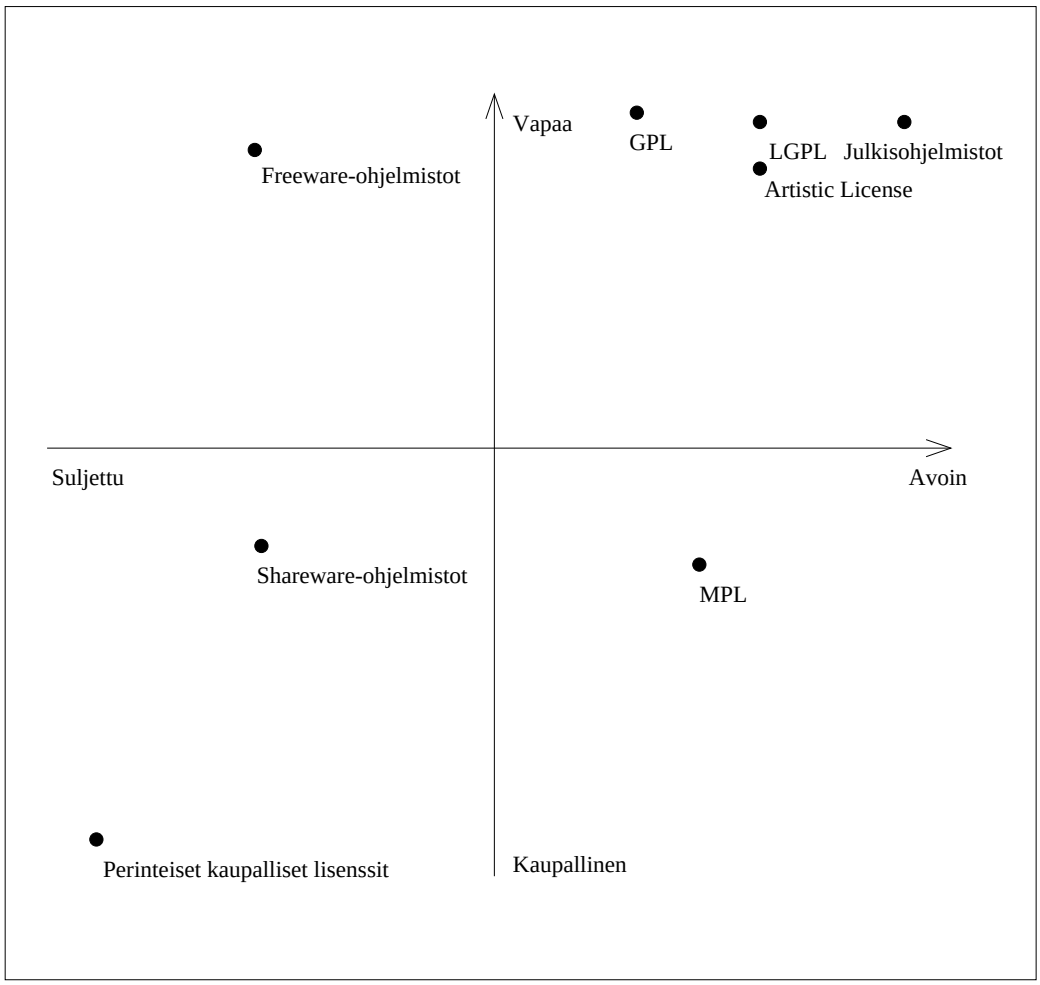
kee päätöksensä. Tiettyjen lisenssien valinta karkottaa mahdollisia osallistujia projektista. Tietysti täytyy myös olla selvillä miten paljon omaa tekijänoikeutta halutaan suojata.

Mikäli ohjelmistoa tehdään yritykselle, tulee tarkoin miettiä lisenssin valinnasta aiheutuvat vaikutukset yrityksen toimintaan. Open Source -ohjelmistokehitystä tekevien yritysten rahoituslähdejakauma on varsin erilainen kuin tavallisissa ohjelmistoyrityksissä.

Kuvassa 1 on asetettu erilaiset ohjelmistolisenssit kenttään, jonka akseleina ovat lähdekoodin avoimuus ja lisenssin kaupallisuus. Kuvan tarkoituksena on selventää erilaisten lisenssien eroavaisuuksia. Kuvasta näemme, että vaikka GPL on erittäin vapaa, pyrkii se kuitenkin olemaan vapaa jossakin määrin avoimuuden kustannuksella. LGPL puolestaan on avoimempi kuin GPL, koska se sallii kaupallisten ohjelmien linkittämisen sillä lisensoituihin ohjelmiin. MPL puolestaan pyrkii olemaan käyttökelpoinen lisenssi ohjelmistotaloille säilyttäen silti avoimuutensa.

Näemme myös kaksi eri ääripäätä ohjelmistolisensseissä: julkisomaisuutta olevat julkisohjelmistot ja perinteiset kaupalliset ohjelmistolisenssit. Kuitenkin on huomattava, että mikä tahansa julkisomaisuutta oleva ohjelmisto voidaan lisensoida uudelleen millä tahansa lisenssillä.

Lisenssien valinta ei aina ole helppoa. Esimerkiksi GGI-projektissa oli pitkän aikaa epäselvää millaista lisenssiä projektissa pitäisi käyttää [10]. GGI (General Graphics Interface) on usean osaprojektin yhteenliittymä. Sen tavoitteina on tehdä useissa eri käyttöjärjestelmissä ja laitteistoarkkitehtuureissa toimiva grafiikkaohjelmointikirjasto, syöttölaiteohjelmointikirjasto ja



Kuva 1: Ohjelmistolisenssien eroavaisuudet

käyttöjärjestelmäytimeen lisättävä moduuli, jolla hoidetaan nopea ja turvallinen grafiikkakortin käsittely.

Projektissa mukana olevat henkilöt halusivat sallia kaupallisten moduulien tekemisen heidän tekemällään rajapinnalla. Tämä aiheutti ongelman, koska eri käyttöjärjestelmien ytimet on lisensoitu eri tavalla. GGI-projektissa päätettiin lopulta käyttää X-konsortion käyttämän BSD-tyylisen lisenssin kaltaista lisenssiä käyttöjärjestelmäytimessä käytettäville ajureille. Ajurien ja

käyttöjärjestelmäytimen välisen kerroksen koodiin sovelletaan käyttöjärjestelmästä riippuvaa lisenssiä. Kaikki ohjelmointikirjastot käyttävät X-tyylistä lisenssiä. Muut osat ovat joko yleisomaisuutta tai sen kirjoittajan valitseman lisenssin alaisia.

Tämä valinta saattaa vaikuttaa monimutkaiselta ja sitä se onkin. Se on kuitenkin ainoa tapa, jolla voidaan tyydyttää kaikkia osapuolia. GGI-projektin lisensointi oli erityisen vaikea juuri käyttöjärjestelmäytimeen tehtävän moduulin lisensoinnin takia. Sovellusohjelmakehityksessä tällaisia lisensointiongelmia harvemmin tulee. Yleisimmin tunnettu ongelma on kaupallisen kirjaston käyttäminen ohjelmassa, joka on lisensoitu GPL:n alla. Tällöin ohjelmaa ei saa uudelleenlevittää binäärimuodossa lainkaan.

4 OHJELMISTOKEHITYS

4.1 Perusasiat

Open Source -projektit ovat kuin mitä tahansa ohjelmistotekniikan projekteja perusasioiltaan ja työkaluiltaan. Koodin kirjoitus on samaa, tehdään sitä sitten työn tai harrastuksen takia; motivaatio tosin on harrastuksessa usein parempi.

Projekteihin ei usein ole käytettävissä rahaa lainkaan tai hyvin vähän. Niitä tehdään vapaa-ajalla tai muun työn ohessa. Tämä asettaa vakavia rajoituksia sille, kuinka nopeata kehitys voi olla. Tästä syystä kehitys voi olla hyvinkin hidasta.

Suurimpien projektien vetäjät kuten Linus Torvalds (Linux-käyttöjärjestelmäydin), Dirk Hohndel (XFree86), Miguel de Icaza (GNOME) ja monet muut ovat saanet tai hankkineet töitä, joissa he voivat keskittyä päätoimisesti omaan Open Source -projektiinsa.

Ohjelmistotuotannollisissa asioissa Open Source -projektit eroavat jonkin verran perinteisistä kaupallisista projekteista. Koska joukko on hyvin irtonainen ja kiinteää aikataulua ei ole, on kehittäjien kiinnittäminen tiettyyn ohjelmistotuotannolliseen prosessiin hankalaa. Niinpä perinteistä vesiputousmallia ei ole käytössä Open Source -projekteissa, vaan ohjelmistot kasvatetaan. Kasvatuksella tarkoitetaan Frederick P. Brooksin Mythical Man-Month kirjassa mainitsemaa tapaa tehdä ohjelmistoja [11]. Tämä on luonnollinen tapa tehdä ohjelmistoja Open Source -ympäristössä, jossa jatkuva mielenkiinnon ylläpitäminen on tärkeää. Kaikki vesiputousmallin vaiheet ovat nivoutuneet

yhteen ja niiden sykliä toistetaan monta kertaa. Yhden syklin voidaan katsoa olevan aika edellisestä julkistuksesta saadun palautteen käsittelystä uuden version julkistukseen.

4.2 Katedraali ja basaari

On olemassa pääasiassa kaksi tapaa tehdä ohjelmistokehitystä "Open Source" -ympäristössä. Nämä kaksi tapaa ovat ns. katedraali- ja basaari-tyylit. Jako katedraali- ja basaari-tyyleihin on Eric S. Raymondin keksintöä. Hän esitteli tämän ajattelutavan kirjoituksessaan "The Cathedral and the Bazaar" vuonna 1998 [12]. Raymond kuvailee siinä erilaisia tapoja tehdä Open Source -ohjelmistokehitystä ja kertoo omista kokemuksistaan basaari-tyylin käytöstä. Tämä dokumentti aiheutti julkaisunsa jälkeen räjähdysmäisen kasvun Open Source -projekteissa. Luettuaan sen päätti Netscapen johto julkistaa oman seittiselaimensa Open Source -tuotteena.

Katedraali on perinteisempi tapa, jota on käytetty useimmin vanhoissa Open Source -projekteissa. Tällä tavalla johdetussa projektissa on pieni joukko taitavia ihmisiä, jotka päättävät siitä, mitkä muutokset tehdään varsinaiseen koodipuuhun, ja vain heillä on oikeus tehdä nämä muutokset. Katedraalissa julkaisujen välit ovat harvassa ja niiden laatu on yleensä hyvä. Julkaisujen hitauden takia kehitys usein etenee hitaasti. Tämän takia käyttäjäkunta on passiivisempi tekijöille lähetettävien korjauksien ja itse kehitystyöhön osallistumisen suhteen.

Basaari on uudempi ja dynaamisempi tapa hoitaa sama asia. Julkaisujen välit saattavat olla hyvinkin pienet, jopa kerran päivässä tapahtuvia julkaisu-

ja on nähty joissakin projekteissa, kun kehitys käy kuumimmillaan. Aktiivisessakin kehityksessä oleva lähdekoodi on näissä projekteissa varsin julkinen ja siihen pääsee tekemään muutoksia iso joukko henkilöitä. Näissäkin projekteissa on asetettu jonkinlaisia rajoituksia sille, kuka muutoksia saa tehdä, ettei kuka tahansa pääsisi tuhoamaan muiden työtä. Koska julkaisuja on suhteellisen usein, voivat aktiiviset käyttäjät antaa palautetta jatkuvasti. On erittäin tärkeää, että käytettävissä on kaiken aikaa toimiva ohjelma, jotta kehittäjillä säilyisi mielenkiinto projektiin. Mielenkiinto on yleensäkin Open Source -projekteissa hyvin tärkeää, koska ainoat motivaattorit, jotka tuovat kehittäjiä Open Source -projekteihin, ovat yleinen mielenkiinto, "hack value" ja halu saada käyttöönsä ohjelmisto, jota tarvitsee.

Basaari-tyyli on nykyään vallalla oleva tapa järjestää Open Source -projekteja. Se näyttääkin tulosten perusteella olevan yleisesti ottaen parempi tapa. On kuitenkin projekteja, jotka ovat pitäytyneet katedraaliajattelussa. Eräs näistä on Linux-ytimen kehitys. Sen kehitys on järjestetty katedraalimaisesti siten, että kullakin osa-alueella on oma vastuuhenkilönsä. Tämä vastuuhenkilö ottaa vastaan kaikki omaan vastuualueeseensa liittyvät korjaukset ja virheraportit ja välittää ne edelleen Linus Torvaldsille. Hän tekee sitten lopullisen päätöksen siitä, mitkä pääsevät viralliseen julkistukseen. Tällainen hyväntahtoinen diktatuuri on tähän mennessä toiminut tälle projektille erittäin hyvin.

Katedraali ja basaari eivät ole toisensa poissulkevia. Projektiin kohdistuvan kiinnostuksen vaihdellessa voidaan vaihdella näiden kehitysmallien välillä. Näin on tehty esimerkiksi GTK -- -projektissa, jossa aktiivisten kehittäjien lukumäärä on vaihdellut melko paljon. Kun kehitys on ollut kiivainta, on käytetty basaari-tyyliä, kun taas hiljaisina aikoina on käytetty katedraali-tyyliä.

Kumpaakin tyyliä voidaan käyttää, riippuen projektin tarpeista.

4.3 Yhteydenpito

Koska Open Source -projekteihin osallistuvat ihmiset ovat usein maantieteellisesti hyvinkin hajallaan, ovat hyvät yhteydenpitokanavat välttämättömyys. Usein tärkeimmät yhteydenpitokanavat ovat sähköposti ja sähköpostilistat. Nämä ovat projektien elinehto. Sähköpostissa vasteajat ovat kuitenkin niin suuret, että pelkän sähköpostin käyttö hidastaa projektin etenemistä. Lisäksi sähköpostissa henkilöiden vuorovaikutus on hyvin rajoitettua.

IRC (Internet Relay Chat) on erittäin suosittu tapa käydä välittömämpää palautetta tarvitsevaa keskustelua. IRC on keskustelufoorumi, jossa ihmiset voivat liittyä "kanaville", joissa varsinainen keskustelu sitten käydään tekstimuotoisena. IRC:n käyttö on joissakin projekteissa jopa tärkeämpää kuin sähköpostilistat. IRC:n välityksellä pidetään kokouksia, jotka muuten olisivat mahdottomia välimatkojen takia.

Kolmas tärkeä tiedonjakokanava on seittisivut. Useimmilla projekteilla on omat seittisivunsa. Näiden sivujen kautta välitetään tietoa tietenkin myös käyttäjille. Koska sähköposti ja seittisivut ovat hyvin rajallinen media, tapaavat suurimpien projektien henkilöt eri konferensseissa ja vastaavissa tilaisuuksissa aina silloin tällöin. Näin edistetään myös projektihenkilöiden yhteenkuuluvuutta.

4.4 Työkalut

Open Source -projekteja varten on kehitetty useita erilaisia työkaluja. Miltei aina nämä työkalut ovat itsessään Open Source -ohjelmistoja. Näistä varmasti tunnetuin ja käytetyin on CVS (Concurrent Versioning System). Sitä käytetään lähdekoodin varastona ja useiden yhtäaikaisten versioiden kehitysalustana. CVS on versionhallintajärjestelmä, jonka avulla on mahdollista hajauttaa kehitystyö eri puolille verkkoa. Mitään lukitusmekanismia ei käytetä tiedostojen muokkaamisen rajoittamiseksi, vaan päällekkäisyydet hoidetaan palvelimen päässä siinä vaiheessa, kun tiedostoihin tehtyjä muutoksia ollaan kirjaamassa palvelimelle. Tämä mahdollistaa tehokkaan rinnakkaisen kehitystyön. Yhdistettynä tehokkaaseen kehitysympäristöön, johon CVS on integroitu, ei projektin nopealle etenemiselle ole teknisiä esteitä.

CVS:ään liittyen on tehty muutamia työkaluja helpottamaan sen hallittavuutta. Yksi tällainen työkalu on Bonsai. Se on alunperin kehitetty Netscapen sisäistä käyttöä varten. Kun Netscapen selain julkistettiin Open Source -tuotteena, julkaistiin sen rinnalla myös muutamia projektinhallintaan liittyviä yrityksen omia työkaluja. Bonsai on yksi näistä työkaluista. Bonsai on CVS:n kehityspuidenhallintaohjelma. Sen avulla voi tehdä kyselyitä CVS:ssä olevan koodin tilasta ja siihen tehdyistä muutoksista. Sen avulla voi muun muassa selvittää kuka on tehnyt jonkin tietyn koodirivin.

Virheiden seurantaan ja virheraportointiin on useita eri työkaluja. Suosituimmat niistä ovat Bugzilla ja GNATS. Bugzilla on Netscapen tuottama seittipohjainen virheraporttien hallintajärjestelmä. GNATS puolestaan on GNU-projektin sähköpostipainotteinen vastaava järjestelmä, joka on kuitenkin monipuolisempi kuin Bugzilla tai muut vastaavat järjestelmät. Siihen

on viime aikoina kehitetty myös seittipohjaisia käyttöliittymiä.

Lähdekoodin tarkasteluun seittisivujen kautta on olemassa työkalu nimeltä LXR. Se tehtiin alunperin muuttamaan Linux-käyttöjärjestelmäytimen lähdekoodi HTML-muotoon siten, että funktioiden nimet ovat linkkejä niiden määritelmiin ja niin edespäin. Kun suureen projektiin tulee uusia kehittäjiä, voidaan heidät tutustuttaa lähdekoodiin tämän työkalun avulla.

Käännösympäristön hallintaan tehtyjä vapaita työkaluja käytetään myös hyvin usein. Tällaisia työkaluja ovat mm. autoconf, automake, autoheader ja aclocal. Nämä muodostavat yhdessä voimakkaan paketin, jolla on helppo hallita monille eri arkkitehtuureille käännettäviä ohjelmistoja. Open Source -ohjelmistojen eräs erikoisuus on juuri se, että niitä voidaan käyttää hyvin useissa eri käyttöjärjestelmissä.

4.5 Irtonaisen joukon johtaminen

Open Source -projekteja ei perinteisessä mielessä johdeta lainkaan. Niitä voidaan yrittää ohjata haluttuun suuntaan. Usein Open Source -projekteissa vallitsee meritokratia. Se, joka tekee projektia varten eniten työtä tai tuottaa eniten käyttökelpoista ohjelmakoodia, saa eniten valtaa siihen mihin suuntaan projektia kehitetään. Tämä henkilö on usemmiten projektin jonkinlainen johtaja. Joissakin projekteissa toimitaan eräänlaisessa valaistuneessa diktatuurissa, jossa kaikki saavat esittää mielipiteensä ja tehdä ehdotuksia, mutta on yksi henkilö, joka viime kädessä päättää mitä tehdään. Näin toimii mm. Linux-käyttöjärjestelmäytimen kehitys, jossa diktaattorina toimii Linus Torvalds.

Aina on vaarana, että kehittäjäjoukko tulee liian erimieliseksi ja kehitystyö haarautuu kahteen tai useampaan eri suuntaan. Open Source -lisenssit antavat tähän mahdollisuuden. Näin kävi vapaille BSD-unixeille. Kehittäjien erimielisyydet johtivat kolmeen eri kehityshaaraan (FreeBSD, OpenBSD, NetBSD).

Tällaiset haarautumiset saattavat olla myös hallittuja, kuten GCC-kääntäjän kehityksessä. GCC-projektista erkaantui EGCS-projekti, jossa kääntäjää kehitettiin basaari-tyylillä eikä katedraali-tyylillä kuten alkuperäisessä GCC-projektissa. Kuitenkin kaikki muutokset saatettiin alkuperäisten kehittäjien saataville. Myöhemmin kävi niin, että EGCS-projektista tuli niin suosittu, että siitä tuli virallinen GCC:n kehityshaara.

Open Source -projektien johtamisessa on tärkeintä yrittää pitää kehittäjiä kiinnostuneina ja kehityssuunta yhtenäisenä. Näin voidaan estää mahdolliset haarautumiset. Kehittäjien mielenkiinto on tärkeää, koska suuri osa Open Source -kehityksestä on vapaaehtoista, eikä siitä anneta minkäänlaista korvausta. Tähänkin sääntöön on olemassa poikkeuksia. Joidenkin projektien henkilöt ovat töissä yrityksessä, joka maksaa heille palkkaa kehitystyöstä. Näillä henkilöillä on usein paljon enemmän aikaa käytettävissä projektiin. Niinpä he ovat useimmiten projektin johtohahmoja.

Sähköpostin välityksellä käytävissä keskusteluissa voivat tunteet kuumeta hyvin helposti. Tällaisissa tilanteissa Open Source -projektissa tulisi olla henkilö, joka pystyy laukaisemaan ristiriitatilanteet ennen kuin niistä kehittyy niin pahoja, että kehitys haarautuu tai pahimmassa tapauksessa loppuu. Tunteiden kuumenemista on esiintynyt useita kertoja Linux-ytimen kehitykseen keskittyvällä sähköpostilistalla. Erimielisyydet on onneksi pystytty rat-

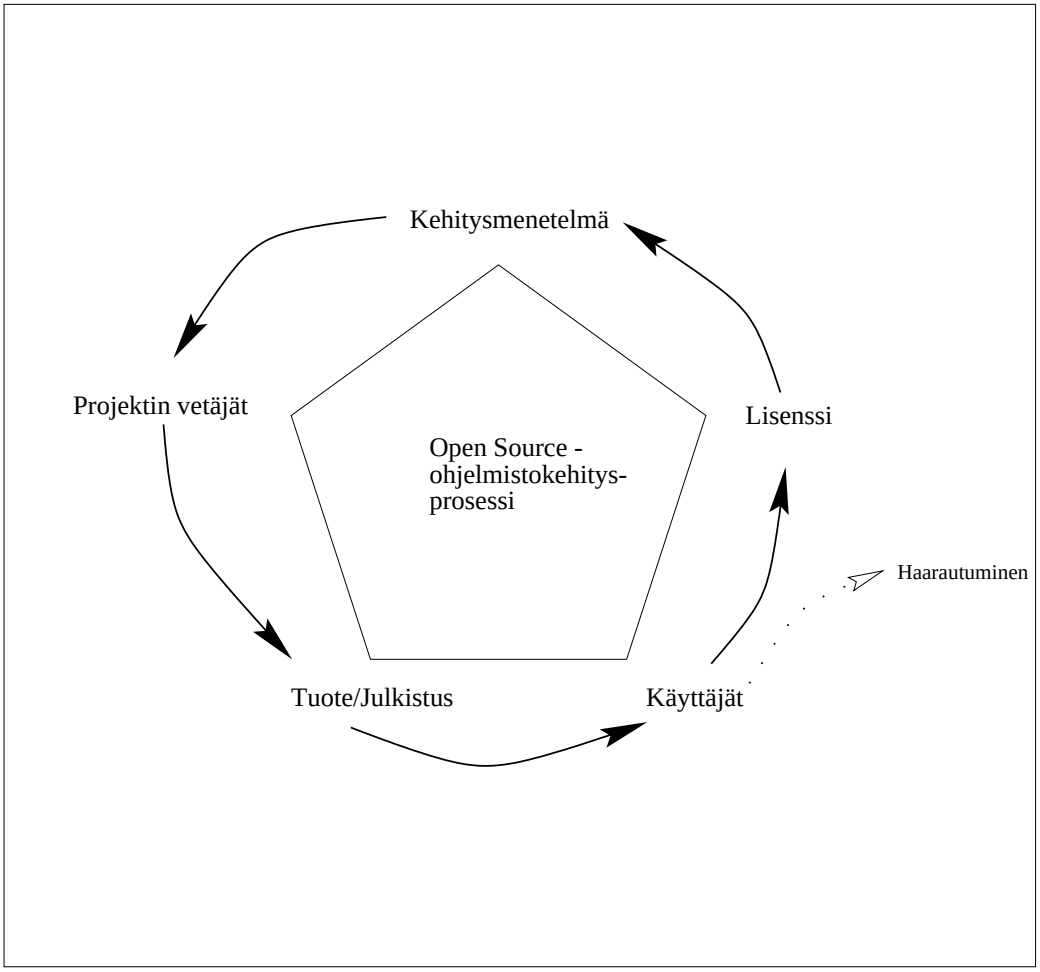
kaisemaan.

Irtonainen kehitysjoukko tuo esiin myös tiedonvälitysongelman: kuinka saadaan tiedotettua kaikille projektissa oleville kaikki tarpeellinen tieto. Esimerkiksi pitäisi huolehtia siitä, ettei samaa asiaa tehtäisi kahteen kertaan tai kahdella eri tavalla. Tähän ongelmaan ei ole helppoa ratkaisua. Mikään ei kuitenkaan korvaa hyvää dokumentaatiota projektin jokaisessa vaiheessa. Open Source -projekteissa dokumentaation täytyy olla digitaalisessa muodossa kaikkien saatavilla, jotta siitä olisi jotain hyötyä. Apuvälineinä tiedon välittämisessä voidaan käyttää esimerkiksi sähköpostilistoja ja seittisivuja. Mnemonic -projektissa, kuten muutamissa muissakin projekteissa, on tehty erillinen sähköpostilista CVS:n lokiviestien lähettämistä varten [13]. Tällä listalla oleville lähetetään sähköpostiviesti aina, kun joku kehittäjistä tekee muutoksen kehityspuuhun. Menettely on osoittautunut hyväksi tavaksi tiedottaa aktiivisimmille kehittäjille mitä on tehty.

4.6 Ohjelmistokehityksen kulku

Open Source -ohjelmistokehitys kulkee sykleissä, kuten edellä jo mainittiin. Tämä sykli kulkee erilaisten vaiheiden läpi, joihin vaikuttavat eri ihmiset (Kuva 2). Sykli alkaa projektin vetäjistä, kun he luovat ohjelmiston joko omien tarpeittensa tai käyttäjiltä saadun palautteen perusteella.

Projektin vetäjät tuottavat ohjelmiston, eli kirjoittavat lähdekoodin ja sulauttavat ohjelmistoon muilta saadut korjaukset. Suurissa projekteissa projektin vetäjät toimivat pikemminkin kehityksen ohjaajina kuin aktiivisina kehittäjinä. Esimerkiksi Linus Torvalds ei nykyään kirjoita kovinkaan paljoa



Kuva 2: Open Source -ohjelmistokehitysmalli

Linux-ytimen koodista, vaan valtaosa koodista tulee erilaisilta kehitysryhmiltä. Hän pääasiassa vain koordinoi näitä ryhmiä.

Projektin vetäjät päättävät myös siitä, missä vaiheessa ohjelmisto kannattaisi julkaista. He huolehtivat ohjelmiston levittämisestä FTP-palvelimille, julkistuksesta, tiedottamisesta ja niin edelleen.

Käyttäjät ottavat ohjelmiston käyttöönsä ja testaavat sitä. Kun he haluavat parannuksia tai korjauksia ohjelmistoon, voivat he vaikuttaa ohjelmis-

ton seuraavaan versioon. Vaikutusmahdollisuudet riippuvat siitä lisenssistä, jonka alla ohjelmisto on julkistettu, ja projektin vetäjien valitsemasta kehitysmallista. Mikäli on kyse basaari-tyylisestä ohjelmistokehityksestä, voivat käyttäjät ja kehittäjät olla hyvinkin lähellä toisiaan.

Mikäli käyttäjät ovat tyytymättömiä projektin johtajiin tai heidän valitsemaansa kehitysmalliin, voivat he aloittaa oman projektinsa. Tämä on kuitenkin suhteellisen harvinaista, sillä tehtävä työmäärä kaksinkertaistuu ja käytettävissä olevien henkilöiden määrä puolittuu tai vähenee vieläkin enemmän.

Kun käyttäjien antama palaute, joka voi olla virheraportteja, korjauspaketteja, parannusehdotuksia jne., on käsitelty ja toteutettu, voidaan aloittaa sama sykli jälleen alusta. Kehityksen käydessä kuumimmillaan voidaan tämä sykli käydä läpi muutamassa päivässä. Joissakin projekteissa julkistuksia on saattanut tulla useita samana päivänä.

Kaupallisissa ohjelmistoissa käyttäjien vaikutusmahdollisuudet ovat lisensoinnin takia miltei olemattomat. Heidän täytyy tyytyä antamaan ehdotuksia ja tekemään virheraportteja. Julkaisutiheyden ollessa kaupallisissa ohjelmistoissa hidas jää asiakkaiden antaman palautteen vaikutus huomattavan pieneksi. Asiakkaalle tulee helposti käsitys, ettei hänellä ole kovinkaan suuria vaikutusmahdollisuuksia.

5 ESIMERKKIPROJEKTEJA

5.1 Yleistä

Erilaisia Open Source -projekteja on olemassa useita satoja. Niiden koko vaihtelee yhden henkilön pikkuprojektista usean sadan henkilön suurprojekteihin. Taulukossa 1 on pieni osa yleisesti käytettyjä Open Source -ohjelmia. Taulukosta näemme, että ohjelmia löytyy miltei kaikkiin tarkoituksiin. Täydellisempiä listauksia erilaisista Open Source -ohjelmistoista löytyy käyttöjärjestelmälevityksien kotisivuilta, mm. Redhat. Näissäkään ei ole kaikkia vaan vain ne, jotka on hyväksytty mukaan käyttöjärjestelmälevitykseen. Joka tapauksessa listalla on useita satoja ellei tuhansia ohjelmistoja.

Ohjelmisto	Kuvaus	Lisenssi	Url
XFree86	X-ikkunointipalvelin	BSD	`http://www.xfree86.org`
Emacs	Tekstieditori ja paljon muuta	GPL	`http://www.gnu.org`
GIMP	Kuvankäsittelyohjelma	GPL	`http://www.gimp.org`
Mozilla	Seittiselain	MPL	`http://www.mozilla.org`
GPG	Kryptogafiasovellus	GPL	`http://www.gnupg.org`
Mutt	Sähköpostin lukuohjelma	GPL	`http://www.mutt.org`

Taulukko 1: Open Source -ohjelmistoja

Projekteissa käytettävät tavat ovat melko samanlaisia. Ainoat eroavaisuudet tulevat katedraali- tai basaarityylien käytöstä. Katedraalityylisistä projekteista on harvoin saatavilla tarkkaa tietoa juuri niiden katedraalimaisuuden takia.

5.2 GTK -- -projekti

GTK -- (GIMP Toolkit--) on C++-rajapinta yleisesti käytössä olevaan GTK+ ikkunointityökaluohjelmistokirjastoon. GTK+ on olioperiaatteilla suunniteltu, mutta C-kielellä toteutettu ohjelmistokirjasto. GTK+:lle on olemassa ohjelmointirajapintoja useille eri ohjelmointikielille.

GTK -- ohjelmointirajapintaa käytetään aktiivisesti muissa projekteissa, kuten GNOME, Terraform ja Mage. GNOME on työpöytäympäristö, johon liittyy useita eri ohjelmistoja. Terraform on korkeuskäyräkarttojen manipulaatioon tarkoitettu ohjelma. Mage on seikkailupeli.

GTK --:n lisenssiksi on valittu LGPL lähinnä siksi, että se on yleiskäyttöinen kirjastorajapinta, eikä ole haluttu estää sen käyttöä kaupallisissa tuotteissa. LGPL on luonnollinen valinta, koska se on myöskin GTK+-projektissa käytettävä lisenssi.

Tätä kirjotettaessa GTK --:n viimeisin vakaa versio oli 1.0.2. Siinä oli yli 110 luokkaa. Projekti toimii läheisessä yhteistyössä GTK+- ja GNOME-projektien kanssa. Muutokset, joita tehdään GTK+:aan, heijastuvat myös GTK --:een.

Projektilla on kolme pääylläpitäjää ja 10 muuta merkittävää kehittäjää. Lisäksi projekti saa korjauksia ja parannuksia lukuisilta henkilöiltä. Projektin suosio on viimeaikoina ollut vähäistä ja kehitys on muuttunut katedraalimaiseksi. Aikaisemmin kehitys oli basaarivoittoista. Tero Pulkkinen, yksi projektin vetäjistä, toivoo, että kiinnostus projektia kohtaa kasvaisi ja kehitys voitaisiin muuttaa basaarimaisemmaksi [14].

Projektissa käytetään normaaleja työkaluja. Tärkeimpiä näistä ovat CVS, autoconf/automake, libtool, egcs, emacs, docbook, perceps ja IRC. Erityisesti IRC:n katsotaan olleen hyödyllinen työkalu kommunikoitaessa GTK+:n ja GNOME:n kehittäjien kanssa. Lisäksi IRC:n kautta on tutustuttu uusiin GTK -- -projektista kiinnostuneisiin henkilöihin.

Projektipäällikkö Tero Pulkkinen pitää projektin suurimpina ongelmina seuraavia asioita:

- Ihmisten kiinnostumaan saaminen. Ohjelmistokirjastojen ohjelmoiminen ei ole yhtä kiinnostavaa kuin käyttäjälle näkyvien asioiden tekeminen varsinkin, kun GTK+-projektin tuotoksia voidaan käyttää suoraankin. Tietoa koodissa olevista virheistä saa, mutta ei juurikaan valmiita ratkaisuja näihin ongelmiin ennen kuin ohjelman koodi on todella hyvässä kunnossa.

- Olemassa olevan koodin ylläpito. Koodin keräämistä ei pidetä ongelmana, mutta sen tason pitäminen korkeana on todellinen ongelma.

- Yleisten periaatteiden saaminen kaikkien tietoon. Samoja asioita tehdään koodiin eri tavalla ja sama asia tehdään uudelleen eri kohdissa.

- Kommunikaatio. Useimmat keskustelut käydään sähköpostilistoilla ja vain harvoin käytetään IRC:tä. Näin vasteajat saattavat olla pahimmassa tapauksessa useita päiviä.

- Uusien kehittäjien saaminen ja vanhojen menettäminen. Kun koodi alkaa olla suhteellisen valmis, kuvitellaan usein, että projekti on niin suuri, että sen koodiin tutustuminen vie liikaa aikaa. Tämä luulo ei kuitenkaan ole välttämättä totta, mutta se karkottaa uusia mahdollisia

kehittäjiä. Jos koodiin tutustuminen jätetään tekemättä voidaan tehdä turhaa työtä, joka näkyy koodin ylläpidon tarpeen kasvuna. Toinen tapaus, joka lisää koodin ylläpidon tarvetta, on projektissa kauan olleen kehittäjän lähteminen projektista. Tämän henkilön tekemää koodia on muiden projektiin jäävien pidettävä yllä.

- Tekniset ongelmat. Kääntämisprosessin pitkä kesto aiheutti ongelmia julkaisutahdin pitämisessä nopeana. Lisäksi GTK+:n monimutkaisuus ja muutokset ovat tuoneet omat ongelmansa.

Kaiken kaikkiaan Tero Pulkkinen pitää GTK-- -projektia onnistuneena, koska se on mahdollistanut C++:lla tehtävän ohjelmistokehityksen GTK+:lle. Sittemmin GTK-- on saanut useita kilpailijoita, mutta luultavasti GTK-- on edelleen eniten käytetty C++-rajapinta GTK+-kirjastoon.

5.3 APACHE-projektit

Apache on seittipalvelinohjelmisto [15]. Sen tehtäviin kuuluu http-kyselyihin vastaaminen ja kysytyn sivun lähettäminen asiakkaalle. Apachessa on lisäksi mahdollista tuottaa dynaamisia sivuja erilaisten ohjelmointirajapintojen kautta.

Apache-projekti alkoi vuoden 1995 alussa seittimestareiden ryhmittymän jakaessa keskenään NCSA-httpd:hen tekemiään korjauksia. NCSA-httpd oli tuohon aikaan Internetin suosituin seittipalvelin. Sitä ei kuitenkaan enää aktiivisesti kehitetty. Tähän tarpeeseen perustettiin Apache-projekti. Se otti pohjakseen NCSA-httpd:n koodin.

Projektin on alkuperäisestä kehitysalustastaan johtuen ollut pakko valita BSD-tyylinen lisenssi. Apache on alunperin rakennettu NCSA-httpd:n koodin pohjalta, joka on lisensoitu BSD-tyylisellä lisenssillä. Myöhemmin miltei kaikki NCSA:lta saatu koodi on korvattu Apache-projektin omalla koodilla. Vaikka mitään sidonnaisuutta vanhaan koodiin ei enää ole, on Apachen koodi silti pidetty BSD-lisenssin alaisena. BSD-tyylisellä lisenssillä on haluttu edistää ulkopuolisten moduulien kirjoitusta. Tässä onkin onnistuttu varsin hyvin. Muiden tahojen Apachelle tekemiä moduuleja on monia kymmeniä.

Apachen ympärille kasvoi aktiivinen käyttäjäkunta suureksi osaksi siksi, että Apache-palvelin on ylivoimainen verrattuna muihin vapaasti saataviin http-palvelinohjelmistoihin. Apachen moduulirajapintaa hyödyntämään syntyi useita eri projekteja. Myöhemmin perustettiin Apache Software Foundation, jonka tarkoituksena on toimia Apache-palvelinprojektin ja muutaman Apacheen liittyvän projektin suojelijana.

Netcraft Inc:n tekemän tutkimuksen mukaan Apache on tällä hetkellä suosituin Internetissä käytetty http-palvelin [16]. Tutkimuksen mukaan yli 50 prosenttia maailman http-palvelimista on Apache-pohjaisia. Apache ohittaa seuraavana tulevan Microsoft IIS-palvelimen noin 30 prosentilla. Apache on lisäksi voittanut lukuisia eri palkintoja parhaana http-palvelinohjelmistona.

5.4 AIDE-projekti

Aide (Advanced Intrusion Detection Environment) on ohjelmisto, jolla on tarkoitus erilaisin menetelmin selvittää, onko tarkasteltavalle palvelimelle tai työasemalle murtauduttu ja sen tietoja luvattomasti muutettu [17].

Tämä ohjelmisto on kehitetty korvaamaan Tripwire-niminen kaupallinen, samaan tarkoitukseen kehitetty ohjelmisto. Ohjelmiston tavoitteena on paikata joitakin Tripwiressa olevia puutteita. Ajanmukaisilla tarkistussumma-algoritmeilla pyritään parantamaan saavutettavaa varmuutta koskemattomuudesta. Ohjelmisto toimii lukemalla konfiguraatiotiedoston ja sen perusteella luo tietokannan, johon talletetaan konfiguraatiotiedostossa valittujen tiedostojen attribuutteja ja niiden tarkistussummia. Tätä tietokantaa verrataan samalla ohjelmalla aikaisemmin luotuun tietokantaan. Tietokantojen eroavaisuuksista ohjelma päättelee, miten tiedostoja on muuteltu.

Aidessa on pyritty konfiguraatiotiedosto pitämään Tripwiren kaltaisena. Kuitenkin siten, että tiedostoja voi valita helposti säännönmukaisilla lauseilla. Tietokannan luomisessa on käytetty tehokkaita algoritmeja tarkistussummien laskemiseen ja tiedostojen valintasääntöjen tarkistukseen. Tietokannan rakenne on suunniteltu siten, että ohjelman eri versioiden luomaa tietokantaa voi lukea sekä uudemmalla että vanhemmalla versiolla ohjelmasta. Nämä ominaisuudet tekevät siitä ylivoimaisen muihin ilmaisiin eheystarkistimiin verrattuna.

Projektia käytettiin kirjoittajan testinä siitä, miten Open Source -projektit toimivat. Motivaationa projektin aloittamiselle olivat Tripwire-ohjelmistossa olevat puutteet. Tripwiren myöhäisemmät versiot ovat olleet kaupallisia, joten ne eivät ole kaikkien käytettävissä. Tämän tyyppisistä ohjelmistoista on aikaisemminkin yritetty tehdä vapaita versioita, mutta yksikään ei ole saavuttanut toiminnallisuudessa Tripwiren tasoa. Vapaalle versiolle tämän tyyppisestä ohjelmistosta oli siis olemassa selvää kysyntää.

Projektissa on alusta asti ollut mukana kirjoittajan lisäksi Pablo Virolai-

nen, joka teki projektia saadakseen Ohjelmistotekniikan laitoksen Ohjelmistojen testaus-kurssille ohjelman, jota opiskelijat voisivat testata. Tarkoituksena oli, että ohjelmasta tehtäisiin ensin joten kuten toimiva versio ja julkaista se. Ensimmäisellä julkistuksella oli tarkoitus houkutella mahdollisimman suuri joukko ihmisiä mukaan kehitykseen.

Suunnitelman toteutus onnistui kohtuullisen hyvin. Versio 0.1 julkistettiin 13. elokuuta 1999. Julkistuksesta ilmoitettiin Freshmeat:ssä. Freshmeat on verkkojulkaisu, joka on erikoistunut uusien Open Source -ohjelmistojen julkaisuista kertomiseen [18]. Ensimmäinen versio haettiin ftp- ja seittipalvelimilta noin 370 kertaa. Hyvin pian julkistuksen jälkeen ensimmäinen virheraportti saapui. Korjauspaketteja on tähän menessä tullut vähän. Projektia varten luodulla sähköpostilistalla on tällä hetkellä 10 henkilöä. Seuraavia versioita on haettu jakelupalvelimilta noin 220 per versio. AIDE:lla on noin 20 aktiivista käyttäjää, joista yksi on Tampereen Teknillisen Korkeakoulun Tietotekniikan osaston Unix-ylläpito.

Projektia varten ei ollut käytettävissä rahaa lainkaan, ainoastaan kahden ihmisen työpanos. Käytettävissä olevat laitteet olivat tietotekniikan osaston Solaris-käyttöjärjestelmää käyttävät työasemat ja palvelimet sekä kirjoittajien omat Linux työasemat.

Työkaluina projektissa on käytetty autoconf/automake/autoheader työkaluja käännösympäristön hallintaan. Emacs tekstieditoria käytettiin lähdekoodin kirjoitukseen. Ohjelman kääntämiseen käytettiin GCC-kääntäjää, Flex-leksikaalista analysaattoria ja Bison-syntaksianalyysigeneraattoria. Ensimmäisen julkistuksen jälkeen projektissa alettiin käyttää CVS-versionhallintaohjelmistoa kehityksen hajauttamiseksi verkkoon. Tätä ennen oli pro-

jektissa käytetty RCS-versionhallintaa. RCS on rajoitettuneempi kuin CVS, mutta sopii paremmin pieniin ympäristöihin. Sähköpostilistojen ylläpitoon käytettiin Majordomo-ohjelmistoa. Myöhemmät julkistukset allekirjoitettiin käyttäen GPG-ohjelmistoa. GPG on GPL-lisensoitu kryptografiasovellus.

Projekti on vielä kovin alkuvaiheessa, joten mitään kauaskantoisia tuloksia ei tämän projektin perusteella kannata vielä arvioida. Kaiken kaikkiaan projekti näyttää tarpeeksi elinvoimaiselta jatkaakseen ainakin versioon 1.0 saakka. Open Source -ohjelmistokehityksen tutkimiseen projekti on antanut jonkin verran valaistusta. Syntyneet käsitykset ovat varmasti aiheuttaneet heijastuksia muualle tähän diplomityöhön. Uusien kehittäjien saaminen kiinnostumaan projektista on ollut kovin vaikeaa.

Projektin tuloksena on syntynyt miltei kaikki suunnitellut piirteet sisältävä suhteellisen virheetön ohjelma. Ohjelma sisältää 20000 riviä koodia. Suuri osa tästä koodista on kerätty muista lähteistä. Ohjelman toteuttaminen muutaman kuukauden sisällä ei olisi ollut mahdollista ilman ulkopuolisen koodin käyttöä. Nämä ovat Open Source -ohjelmistojen suurimpia vahvuuksia: nopea kehitys ja koodin uudelleenkäyttö.

6 SOVELTAMINEN YRITYSMAAILMASSA

6.1 Perusteita Open Source -mallin käytölle

Perinteisissä ohjelmistoyrityksissä on varjeltu lähdekoodia erittäin tarkoin, koska se on usein yrityksen ainoa myyntiartikkeli ja pääasiallinen tulonlähde. Mikäli tämä tulonlähde annetaan vapaaseen levitykseen, se menetetään osittain ellei täysin. Tämä saattaa vaikuttaa hullulta ja tietyissä tilanteissa se sitä onkin.

Miksi sitten kannattaisi ryhtyä tekemään Open Source -ohjelmistokehitystä? Kaikki ohjelmistokehitys tuskin koskaan tulee olemaan Open Source -ohjelmistokehitystä. Yleiskäyttöisten ja jokapäiväisten ohjelmistojen kehitys saattaa hyvinkin tulla pääosin Open Source -ohjelmistokehitykseksi, koska kun joku tekee jostakin ohjelmistosta vapaasti saatavan version, syö se vääjäämättä markkinoita muilta samaan tarkoitukseen tehdyiltä kaupallisilta versioilta. Näin ko. ohjelmistotuotteen rahallinen arvo laskee mitättömiin - itse asiassa se voi laskea niin pieneksi, että sitä ei yksinkertaisesti enää kannata pitää suljettuna tuotteena. Tämä kehitys on nyt nähtävissä Unix-käyttöjärjestelmissä. Ilmaiset Unix-kloonit ovat valtaamassa niin suuren markkinaosuuden, että muiden on vaikea pitää asemaansa. Ensimmäinen suuri kaupallista Unix-käyttöjärjestelmää kauppaava yritys, joka on tehnyt siirron Open Sourcen suuntaan, on Sun Microsystems, jonka Solaris-käyttöjärjestelmä on nyt saatavissa lähdekoodimuodossa miltei ilmaiseksi. Lisenssi, jolla Sun Solaris-käyttöjärjestelmää jakaa, ei ole täysin Open Source -lisenssi, mutta tämä on kuitenkin liike siihen suuntaan.

Toinen suuntaus, joka tukee Open Source -liikettä, on halu maksaa sisällöstä eikä ohjelmistosta, joka mahdollistaa palvelun. Tämä trendi on nähtävissä mm. Internet-palveluntarjoajien liittymäpaketeissa. Niissä tulee hyvin usein mukana kattava paketti ohjelmistoja, jotka mahdollistavat palveluihin käsiksi pääsemisen. Asiakkaat eivät kuitenkaan maksa näistä ohjelmistoista mitään vaan ne tulevat kylkiäisinä. Maksu tulee mahdollisuudesta käyttää kyseistä palvelua.

Miltei kaikki Open Source -ohjelmistot on lisensoitu siten, että niillä ei ole minkäänlaista takuuta. Tämä saattaa olla kynnyskysymys joillekin yrityksille Open Source -ohjelmistojen käyttöönotossa. Tätä ongelmaa pidetään usein pahempana kuin se todellisuudessa on. On olemassa yrityksiä, jotka tarjoavat kaupallista tukea Open Source -ohjelmistoille. Esimerkiksi Redhat myy tukipalvelua omalle käyttöjärjestelmälevitykselleen ja Wilberworks myy tukipalvelua GIMP-kuvankäsittelyohjelmalle. Toisaalta eräiden kaupallisia ohjelmistoja myyvien yritysten tarjoamasta teknillisestä tuesta on paljon vähemmän hyötyä kuin Internetin uutisryhmistä ja sähköpostilistoilta saatavasta ilmaisesta tuesta. Esimerkiksi jos Linux-käyttöjärjestelmäytimestä löydetään turvallisuusreikä, on ongelmaan olemassa korjaus yleensä kolmen tunnin sisällä vian julkistamisesta. Monilla yrityksillä tällaisen korjauksen tuottaminen saattaa kestää useita kuukausia. Tämän aktiivisuuden takia Linux käyttäjäkunta on saanut 1997 Infoworld-lehden Paras Tuotetuki -palkinnon.

6.2 Open Source -yrityksien toimintatavat

Kirjassa “Open Sources: Voices from the Open Source Revolution” Brian Behlendorf on sitä mieltä, että Open Source -ohjelmistokehitys on elinkel-

poinen alusta toimivalle yritystoiminnalle [19]. Tämän todistavat oikeaksi monet yritykset, jotka eivät voisi toimia ilman Open Source -ohjelmistoja. Tällaisia yrityksiä ovat mm. Redhat, Cygnus, Cyclic ja Gnome-support.

Brian Behlendorf on myös sitä mieltä, että yrityksen, joka aikoo käyttää Open Source -ohjelmistokehitystä, tulee miettiä tarkoin yritysmalliaan. Open Source -ohjelmistoja tuottavan yrityksen rahoitusrakenne on huomattavan erilainen verrattuna perinteiseen ohjelmistoyritykseen.

Mistä yritys sitten saa rahansa Open Source -ohjelmistoissa? Osa tuloista tulee edelleen ohjelmistopakettien myynnistä. Ihmiset haluavat ostaa jotakin käsin kosketeltavaa. Mukana tulevat ohjekirjat ovat toinen erittäin vahva motivaattori. Kaikki ihmiset eivät kertakaikkiaan viitsi tai eivät verkkoyhteyden rajoitetun kaistan takia pysty hakemaan kaikkia haluamiaan ohjelmistoja verkosta.

Toinen rahanlähde on teknillisen tuen ja takuun antaminen Open Source -ohjelmistoille. Vaikka teknillistä tukea saa kohtuullisen helposti erilaisista uutisryhmistä ja seittisivuilta, on silti helpompaa soittaa jollekulle henkilölle ja saada häneltä vastaus ongelmaan. Tiedon penkomiseen käytetty aika jää tässä pois.

Kolmas mahdollisuus rahan tekemiseen on olemassa olevan Open Source -ohjelmiston laajentaminen tai kustomointi asiakkaan tarpeisiin. Tätä ideaa hyödyntää Gnome Support. Sen tukema ja laajentama ohjelmisto on GNOME ja siihen liittyvät apuohjelmat [20]. GNOME on kokoelma työpöytäohjelmistoja.

On myös olemassa toisenlainen tapa lähestyä rahan ansaitsemista Open

Source -ohjelmien kirjoituksella. Tätä tapaa käyttävät yritykset CoSource ja SourceXchange. Näiden yritysten toimintasuunnitelma on kerätä projekteja, joita kukaan ei ole halukas tekemään Open Source -lisenssillä ja joista joku on valmis maksamaan. Kun jokin projekti on saanut tarpeeksi rahallista tukea yhdestä tai useammasta lähteestä, se käynnistetään ja siihen hankitaan kehittäjiä. Näille kehittäjille maksetaan jonkinlainen korvaus tämän projektin toteuttamisesta. Tämä on vielä varsin uusi liiketoiminnan muoto ja kummatkin yritykset vasta hakevat omia toimintatapojaan.

6.3 Open Source -ohjelmistot tuotteen osana

Open Source -ohjelmistoja levitetään usein jonkin toisen tuotteen osana. Tämä on täysin mahdollista, koska lisenssit sallivat ohjelmistojen uudelleenlevittämisen binäärimuodossa. Levitettäessä kokonaisia ohjelmia muuttumattomana tuotteen osana ei lisenssiongelmia todennäköisesti tule. Open Source -ohjelmiston kehittäjät eivät välttämättä pidä siitä, jos heidän tekemäänsä ohjelmaa käytetään hyväksi tuotteessa ilman, että kyseinen yritys antaa mitään takaisin ohjelman jatkokehitykseen. Tällainen toiminta aiheuttaa usein enemmän huonoa julkisuutta kuin mitä ohjelmiston käytöstä on hyötyä.

Mikäli Open Source -ohjelmistosta käytetään vain jokin osa koodista, täytyy olla tarkkana ohjelmiston lisenssin kanssa. Esimerkiksi GPL-lisenssin kohdalla on huolehdittava, että lähdekoodiversio on saatavilla julkisesti koko ohjelmistosta. Ei siis vain alkuperäiseen koodiin tehdyt muutokset, vaan koko ohjelman lähdekoodi tulee olla julkisesti saatavilla. Tämä ei usein kuitenkaan ole haluttavaa, mikäli tehdään perinteistä, suljettua ohjelmistokehitystä. BSD-tyylisillä lisensseillä ongelma ei ole näin vakava, mutta niidenkin

asettamat vaatimukset täytyy ottaa huomioon päätöksiä tehtäessä. Kunkin Open Source -lisenssin vaatimukset ovat hiukan erilaiset ja ne täytyy punnita tarkasti, ennen kuin otetaan käyttöön Open Source -koodia suljetussa ohjelmistossa.

Corel on käyttänyt onnistuneesti vapaasti saatavilla olevia Open Source -ohjelmistoja. Corel on päättänyt ottaa verkkotietokoneensa käyttöjärjestelmäytimeksi Linuxin. Mikäli yritys antaa käyttämilleen Open Source -ohjelmistoille täyden tukensa, on se kaikkien edun mukaista. Corelin tapauksessa Linux-yhteisö sai apua ARM-arkkitehtuurille tapahtuvassa siirtotyössä. Corel puolestaan sai suhteellisen pienellä investoinnilla luotettavan käyttöjärjestelmän omaan verkkotietokoneeseensa. Corelin käyttöjärjestelmälevityksen betatestauslisenssistä nousi häly, koska sen luultiin olevan liian rajoittava suhteessa Linux-käyttöjärjestelmäytimen GPL-lisenssiin. Corel julkaisi selvennyksen, jossa todettiin, että rajoittava lisenssi koski vain Corelin itsensä kehittämiä ohjelmiston osia, joita ei oltu kehitetty Open Source -lisenssin alaisena. Tämä tapaus osoittaa kuinka herkästi ja voimaakkasti Linux-yhteisö reagoi tällaisiin tapauksiin.

Apache-seittipalvelinohjelmistoa käyttää hyödykseen useamkin kaupallinen yritys. Yhteistyö näiden yritysten ja Apache-ryhmittymän välillä on sujunut hyvin. Apachea hyödyntäviä yrityksiä ovat muiden muassa IBM ja Sun Microsystems. Molemmat tukevat aktiivisesti Apachen kehitystyötä.

7 EDUT JA HAITAT

7.1 Resurssit

Usein Open Source -projektia aloitettaessa resurssit ovat hyvinkin rajalliset. Mikäli saadaan tarpeeksi monta henkilöä kiinnostumaan projektista ja ohjelmiston käyttäjäkunta kasvaa tarpeeksi suureksi, on resursseja huomattavan paljon. Tässä tulee usein esiin eräänlainen kana-muna ongelma. Riittäviä resursseja toimivan ohjelman tuottamiseen ei saa ennen kuin on jonkinlainen toimiva ohjelmisto. Joissakin projekteissa on tehty niin, että on ensin tuotettu osittain toimiva prototyyppi, joka voidaan myöhemmin heittää osittain tai kokonaan pois.

Yleisesti ottaen, jos projekti on mielenkiintoinen tai kehitettävä ohjelma koetaan tarpeelliseksi, ollaan paljon valmiimpia tarjoamaan apua. Useat Open Source -projektit ovat kuolleet kaikessa hiljaisuudessa mielenkiinnon puutteen vuoksi. Tietysti riippuu myös projektin vetäjän karismaattisuudesta, pystyykö hän pitämään kehittäjien mielenkiintoa yllä. Vaikka itse idea olisi hyvä, ei se aina takaa menestynyttä ohjelmistoa. Esimerkiksi GNU-projektin eräänä osaprojektina oli tehdä vapaa taulukkolaskentaohjelma. Vaikka taulukkolaskennan katsotaan yleisesti olevan tärkeä, ei tämä projekti silti saanut tuulta siipiensä alle ja se vaipui unohduksiin. Myöhemmin erillinen ryhmä kehittäjiä on saanut aikaan käyttökelpoisen ja elinvoimaisen taulukkolaskentaohjelman GNOME-projektin osana.

7.2 Testaus

Eräs asia, joka Open Source -projekteissa on erityisen hyvällä tolalla, on testaus. Kun suurin osa käyttäjistä kokee, että heidän mielipiteellään on painoarvoa ja että heidän tekemänsä virheraportti saa aikaan korjauksen, tekevät he paljon mieluummin virheraportteja. Tämä on usein ongelmana suljetussa ohjelmistokehityksessä. Virheraporttien käsittely ja niistä aiheutuvat muutokset saattavat näkyä vasta seuraavassa ohjelmistoversiossa, joka voi olla ajallisesti hyvinkin kaukana. Tällöin käyttäjät kokevat ettei heidän mielipiteillään ole vaikutusta.

Lisäksi käyttäjät, joilla riittää taitoa virheen etsimiseen ja korjaamiseen, myös tekevät itse korjauksia ohjelmistoon ja luovuttavat korjauksensa alkuperäisille kehittäjille. Kun tämä yhdistetään nopeaan julkaisutahtiin ja aktiiviseen kehittäjäkuntaan, saadaan suurempi osa ohjelmistoissa olevista virheistä poistettua. Näin ollen myös laatu paranee.

Usein kuitenkaan aluillaan oleva projekti ei saa kovinkaan paljon tällaisia korjauksia. Vasta kun projekti on vakiintunut ja sen käyttäjäkunta on tasaantunut, voi odottaa saavansa hyviä korjauksia projektissa aktiivisesti mukana olevilta käyttäjiltä.

Virheraportteja tulee runsaasti projektista riippumatta. Suuremmissa projekteissa ongelmana saattaa olla pikemminkin virheraporttien suuri määrä kuin niiden puute. Tämän takia useissa projekteissa käytetään jotakin virheidenseurantajärjestelmää.

7.3 Alkuun pääsy ja jatkuvuus

Kuten muuallakin, on alkuun pääseminen usein vaikeaa. On tärkeää saada projekti hyvälle alulle, etteivät mahdolliset kehittäjät katoa huonon alkujulkistuksen takia. Näin on käynyt mm. Mnemonic-projektille, jonka tavoitteena on kehittää GPL-lisenssin alainen ilmainen seittiselain. Tämä projekti on näennäisesti täsmälleen samassa asemassa kuin vuosi sitten. Projekti aloitettiin vuonna 1997. Vuonna 1998 kehittäjät päättivät lähteä kokonaan puhtaalta pöydältä. Vasta 1999 kesällä he saivat jotakin näkyvää aikaan, jonka seurauksena kehittäjien määrä ja yleinen kiinnostus projektia kohtaan on kasvanut.

Hyvän alkujulkistuksen lisäksi voidaan alkuun pääsyä helpottaa hyvällä tiedotuksella. On olemassa useita seittipalveluja, joissa Open Source -ohjelmistojen tekijät voivat ilmoittaa ohjelmistaan tai niihin ilmestyneistä korjauksista. Tällaisia palveluja ovat mm. Freshmeat [18] ja Linux Apps List [21].

Eräs sekä alkuun pääsyyn että jatkuvuuteen vaikuttava asia on, projektin vetäjän karismaattisuus. Hyvät henkilötaidot ovat yleensä erittäin tärkeitä myös Open Source -projektien vetäjille. Karismaattisuutta voi näissä projekteissa osittain korvata hyvällä ohjelmointitaidolla ja ohjelmistosuunnittelutaidolla.

Yleisesti Open Source -projektien jatkuvuudesta ei ole takeita. Yleensä ne jatkuvat niin kauan kuin pääkehittäjää/kehittäjiä kiinnostaa. Mikäli ohjelma on tarpeeksi hyvä tai tarpeellinen, löytyy joku muu, joka jatkaa ohjelmiston kehitystä, kun alkuperäinen kehittäjä siirtyy tekemään jotakin muuta.

Näin on käynyt esimerkiksi GIMP-kuvankäsittelyohjelmalle. Alkuperäiset kehittäjät valmistuivat ja siirtyivät työelämään, eikä heillä enää ollut aikaa jatkaa kehitystyötä. Ohjelmisto oli kuitenkin niin tärkeä monelle, että kehittäjiä löytyi ja työ jatkui. Useissa muissakin projekteissa on niiden johto vaihtunut useaankin kertaan.

Yleensä, jos ohjelmisto on tarpeeksi tärkeä, projekti selviää. Aina on joku, jolla on tarpeeksi aikaa tai halua pitää yllä projektia. Nykyään, kun suurimmat Open Source -projektit ovat kasvaneet valtaviin mittasuhteisiin, saattaa näiden projektien ympärille syntyä pieniä yritysryppäitä. Esimerkiksi Linux-käyttöjärjestelmädistribuutioita on noin kymmenen kappaletta.

7.4 Suunnittelu

Liian usein Open Source -projekteissa jätetään suunnittelu ja määrittely aivan liian pienelle huomiolle. Tästä ei kuitenkaan ole yleensä haittaa, koska projektit ovat kohtuullisen pieniä eivätkä arkkitehtoniset ratkaisut ole kovinkaan tärkeitä näissä ohjelmistoissa. Suuremmissa projekteissa tulee olla järkevä ja suhteellisen helposti laajennettavissa oleva arkkitehtuuri, jotta ne pysyisivät elossa.

Ohjelmiston arkkitehtuurin aiheuttamat rajoitteet ja virheet hoidetaan usein siten, että kun ohjelmiston versionumerossa vaihtuu ensimmäinen kokonaisluku, tehdään täysi tai miltei täysi uudelleenkirjoitus. Ohjelmisto voidaan joutua kirjoittamaan tyhjästä pari kolmekin kertaa, ennen kuin arkkitehtuuri vakiintuu. Esimerkiksi GNU-projektin kääntäjän, GCC:n, arkkitehtuuria on muutettu usein tehokkuuden ja siirrettävyyden takia. Nykyisin

arkkitehtuuri on vakaa. GCC on optimoiva monikielikääntäjä, joka tuottaa binäärikoodia useammalle prosessoriarkkitehtuurille kuin mikään muu kääntäjä maailmassa. Nykyään GCC:n kehityksessä keskitytään lähinnä uusien optimointien ja uusien kielten lisäämiseen sekä standardien mukaiseen kielten toteutukseen.

Yksi suunnittelun ja arkkitehtuurin korjauksia edistävä asia on julkiset sähköpostilistat ja seittisivuilla olevat keskustelufoorumit. Näissä kehittäjät saavat kuulla kunniansa, mikäli heidän tekemänsä ratkaisut vaikeuttavat muiden kehittäjien työtä. Tämä julkisen nolaamisen pelko kannustaa tekemään parempia ohjelmistoja.

Eräs ongelma, joka vaivaa joitakin Open Source -projekteja, on ominaisuuksien rajaaminen. Projektissa mukana olevat henkilöt haluavat, että heidän tekemänsä ohjelmisto tekee kaiken mahdollisen. Tällaisia projekteja ovat GNU Emacs ja Mnemonic. Kummankin projektin kehitystyötä on haitannut huomattavasti ominaisuuksien paljous. Emacs-tekstieditorilla voi esimerkiksi lukea sähköposteja ja Usenet-uutisia, selata seittiä ja tehdä ohjelmistokehitystä erittäin monipuolisesti.

7.5 Asiakkaan näkökulma

Asiakkaalle näkyvin etu on tietenkin se, ettei tarvitse maksaa varsinaisesta ohjelmistosta mitään. Ainoastaan ohjelmiston jakamiseen käytetystä mediasta ja mahdollisista ohjekirjoista joutuu maksamaan.

Toinen etu on ilmainen tuki, jota saa monista uutisryhmistä ja sähköpos-

tilistoilta sekä lukuisilta seittisivuilta. Lisäksi on yrityksiä, jotka tarjoavat maksullisia tukipalveluja niille, jotka eivät luota ilmaiseen tukeen. Tukikysymykseen on kiinnitetty paljon huomiota. Sitä pidetään yhtenä suurimpana syynä Open Source -tuotteiden hitaalle leviämiselle. Tämä on kuitenkin ollut liioiteltua. Kaupallisten ohjelmistojen tukipalvelut ovat usein yhtä huonoja. Hyödyllisempää tietoa saa, jos etsii seitistä vastausta. Kyse on pikemminkin massan hitaudesta. Kestää oman aikansa ennen kuin vakiintuneet markkina-asemat muuttuvat.

Open Source -projekteilla tuotetut ohjelmat ovat yleisesti ottaen parempilaatuisia kuin niiden kaupalliset versiot. Tätä on tutkittu puolueettomissa tutkimuksissa 1989 [22] ja 1998 [23]. Molemmilla kerroilla todettiin, että GNU-projektin tuottamissa Unixin perustyökaluissa on vähemmän virheitä kuin kaupallisten Unixien vastaavissa ohjelmissa.

Asiakkaalle näkyviä haittoja on yleisesti dokumentaation huono tai välttävä laatu. On kuitenkin olemassa Open Source -ohjelmistoja, joiden dokumentaatio on hyvää ja sitä pidetään yllä.

8 YHTEENVETO

Open Source -ohjelmistojen merkitys on kasvanut jo pitkän aikaa ohjelmisto-teollisuudessa ja luultavasti tulee kasvamaan vastakin. Mitä enemmän Open Source -ohjelmistoja kirjoitetaan sitä enemmän ne tulevat korvaamaan vastaavia kaupallisia tuotteita. Tätä trendiä auttaa suurten yritysten välillä käytävä kilpailu, jossa yritykset antavat ohjelmistoja ilmaiseksi vahingoittaakseen vastustajaansa. Kuluttajat tottuvat siihen, että on olemassa ilmaisia ohjelmistoja eikä heidän tarvitse maksaa jokaisesta ohjelmistosta.

Open Source -ohjelmistokehityksestä ei ole hopealuodiksi, mutta se tuo uuden mielenkiintoisen käänteen yrityskeskeiseen kultuuriin. Open Source -malli tuo vääjäämättä muutoksen ohjelmistotuotteiden kauppaan. Ne eivät enää ole vain erikoisohjelmia, vaan täydellisen perusohjelmistopaketin voi saada muutamalla kymmenellä markalla. Tämä nakertaa tehokkaasti joidenkin yritysten voittomarginaaleja. Perusohjelmistoista tulee jokapäiväisiä massamarkkinatuotteita, joiden hinta on hyvin alhainen.

Aivan kaikkiin ohjelmistotuotteisiin Open Source -malli ei sovi. Räätälöidyt ohjelmistot tulevat luultavasti olemaan hyvinkin pitkään suljettuja ohjelmistoja. Armeijoiden erikoisjärjestelmät ovat hyvä esimerkki tällaisista ohjelmistoista. Toisaalta voidaan väittää, että todellinen turvallisuus tulee julkisista järjestelmistä, joiden turvallisuuden ovat tarkastaneet useat vastaavassa asemassa olevan instanssit (peer review).

Open Source -ohjelmistot elävät nyt murroskautta. Ne ovat lopullisesti lyömässä itseään läpi. Suuret yritykset ovat kiinnostumassa niistä; ei pelkästään niiden käyttöominaisuuksien, vaan myös niiden markkina-arvon takia.

Open Source -ohjelmistoja tukevien yritysten määrä on kasvanut huimasti viimeisen vuoden aikana. Open Source -ohjelmistoja tukevat yritykset ovat pääsääntöisesti olleen hyvin menestyviä, ja niiden arvo on noussut huikeasti. Esimerkiksi Redhatin osake on New Yorkin pörssin suurimpia nousijoita.

Laaja aktiivisesti testaava käyttäjäkunta ja teknisesti oikeat päätökset takaavat Open Source -ohjelmistojen laadun. Tätä laatua on vaikea kaupallisten yritysten taata, koska vastaava laatu vaatii valtavat resurssit. Lisäksi yrityksien on vaikeaa pitää tekniset syyt tärkeimpänä kriteerinä päätöksiä tehtäessä. Taloudelliset syyt painavat miltei aina yrityksissä enemmän.

Open Source -ohjelmistokehityksessä on tietenkin ongelmansa. Ongelmat eivät kuitenkaan ole sen ihmeellisempiä kuin missä tahansa muussa monen hengen projektissa. Ohjelmistokehitysmalli tuo hiukan eri tyyppisiä ongelmia esiin kuin mihin perinteisissä ohjelmistoprojekteissa on totuttu. Nämä ongelmat ovat kuitenkin ratkaistavissa.

Open Source -ohjelmistojen valtava määrä on omiaan luomaan varmuutta siitä, että Open Source -malli toimii. Yhtenä osoituksena Open Source -ohjelmistojen käyttökelpoisuudesta voidaan pitää sitä tosiasiaa, että tämä diplomityö on kirjoitettu täysin Open Source -työkaluilla.

LÄHDELUETTELO

Viitteet

[1] E. S. Raymond: New Hacker's Dictionary, MIT Press,1996
ISBN: 0262680920

[2] B. Perens: Open Source Definition Version 1.4 30. syyskuuta 1999
`http://www.opensource.org/osd.html`

[3] GNU-projektin ja FSF:n kotisivut, 30. syyskuuta 1999
`http://www.gnu.org`

[4] Berkeley Standard Distributionin ohjelmistolisenssi, 30. syyskuuta 1999
`http://www.freebsd.org/copyright/license.html`

[5] Free Software Foundation: GNU General Public License Version 2 30. syyskuuta 1999
`http://www.gnu.org/copyleft/gpl.html`

[6] Free Software Foundation: GNU Lesser General Public License Version 2.1 30. syyskuuta 1999
`http://www.gnu.org/copyleft/lesser.html`

[7] Netscape Public License ja Mozilla Public License, 30. syyskuuta 1999
`http://www.mozilla.org/MPL/`

[8] Artistic License, 30. syyskuuta 1999
`http://www.perl.com/language/misc/Artistic.html`

[9] Perl-tulkin ja kielen kotisivu, 30. syyskuuta 1999
`http://www.perl.com`

[10] GGI projektin kotisivu, 30. syyskuuta 1999
http://www.ggi-project.org

[11] F. P. Brooks: The Mythical Man-Month 20th Anniversary edition, Addison Wesley; ISBN:0-201-83595-9 1995

[12] E. S. Raymond: The Cathedral and the Bazaar, 30. syyskuuta 1999
http://www.tuxedo.org/~esr/writings/cathedral-paper.html

[13] Mnemonic projektin kotisivu, 30. syyskuuta 1999
http://www.mnemonic.org

[14] Tero Pulkkisen sähköpostihaastattelu 12-17.8.1999

[15] Apache seittipalvelinohjelmiston kotisivu, 30. syyskuuta 1999
http://www.apache.org/httpd.html

[16] NetCraft Inc: Web Server Survey 21.8.1999
http://www.netcraft.com/survey/Reports/

[17] Aide-projektin kotisivu, 30. syyskuuta 1999
http://www.cs.tut.fi/~rammer/aide.html

[18] Freshmeat, verkkojulkaisu, 30. syyskuuta 1999
http://freshmeat.net

[19] C. DiBona, S. Ockman, M. Stone, E. S. Raymond, K. McKusick, S. Bradner, R. Stallman, M. Tiemann, P. Vixie, L. Torvalds, R. Young, L. Wall, B. Behlendorf, B. Perens, T. O'Reilly, J. Hamerly, T. Paquin, S. Walton: Open Sources : Voices from the Open Source Revolution 1999
http://www.oreilly.com/catalog/opensources/book/toc.html
ISBN:1-56592-582-3

[20] GNOME-projektin kotisivu, 30. syyskuuta 1999
http://www.gnome.org

[21] Linux Apps, verkkojulkaisu, 30. syyskuuta 1999
http://www.xnet.com/~blatura/linapps.shtml

[22] B. Miller, L. Fredriksen, B. So: An Empirical Study of the The Reliability of UNIX Utilities 1989
ftp://grilled.cs.wisc.edu/technical_papers/fuzz.ps

[23] B. P. Miller, D. Koski, C. P. Lee, V. Maganty, R. Murthy, A. Natarajan, J. Steidl: Fuzz Revisited : A Re-examination of the Reliability of UNIX Utilities and Services 1998
ftp://grilled.cs.wisc.edu/technical_papers/fuzz-revisited.ps