Heta Rekilä

Avoin lähdekoodi ja ilmaiset piirto-ohjelmat

Tietotekniikan kandidaatintutkielma

19. joulukuuta 2016

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Heta Rekilä

Yhteystiedot: heta.l.rekila@student.jyu.fi

Työn nimi: Avoin lähdekoodi ja ilmaiset piirto-ohjelmat

Title in English: Open Source and Free Drawing Programs

Työ: Kandidaatintutkielma

Sivumäärä: 22+0

Tiivistelmä: Tutkielmassa selvitetään avoimen lähdekoodin ja sen kehityksen piirteitä ja toimintatapoja, avoimeen lähdekoodikehitykseen osallistujien motivaatioita sekä arvioidaan avoimen lähdekoodin hyviä ja huonoja puolia. Lisäksi tutkielmassa verrataan avointa lähdekoodikehitystä kaupalliseen kehitykseen, ja pohditaan kaupallisuuden ja avoimen lähdekoodin yhdistämistä. Myös avoimelle lähdekoodille oleellisia lisenssejä käsitellään. Näitä osa-alueita peilataan esimerkkien avulla avoimen lähdekoodin piirto-ohjelmiin GIMP:iin ja Kritaan. Tutkielmassa havaitaan, että molemmista piirto-ohjelmista löytyy avoimen lähdekoodin piirteitä ja muita siihen liittyviä ominaisuuksia.

Avainsanat: avoin lähdekoodi, avoimen lähdekoodin sovellusprojekti, avoin lähdekoodikehitys, piirto-ohjelma, kaupallisuus

Abstract: The purpose of this study is to look into the typical aspects of open source and open source development, the motivations of the people who participate in open source development, and to evaluate open source development. Commercial and open source development are compared, and combining commercialism with open source is discussed. Also, the relevant licenses for open source are introduced. Two open source drawing programs GIMP and Krita are used as examples when studying these above mentioned aspects. The study shows that these two drawing programs have typical open source aspects and other qualities connected to that.

Keywords: open source, open source software, open source software development, drawing program, commercialism

Sisältö

1 Johdanto

Tutkimuksen aiheena on avoimen lähdekoodin eri ominaisuudet ja sen piirteiden näkyminen ilmaisissa piirto-ohjelmissa. Avoin lähdekoodikehitys on yksi tapa tehdä sovelluskehitystä, joten tutkielmassa tarkoituksena on tehdä katsaus avoimen lähdekoodin määritelmään ja avoimen lähdekoodisovelluksen kehitykseen. Avoimessa lähdekoodikehityksessä käytettäviin toimintatapoihin tutustutaan myös, ja selvitetään, mitä syitä on juuri avoimen lähdekoodin käytön ja sen kehitykseen osallistumisen taustalla. Lisäksi tutkielmassa arvioidaan avointa lähdekoodia, ja perehdytään avoimen lähdekoodikehityksen ja kaupallisen kehityksen suhteeseen. Tutkimus toteutetaan kirjallisuuskatsauksena ja esimerkkinä toimivat avoimesti toteutetut ilmaiset piirto-ohjelmat GIMP ja Krita.

Piirto-ohjelmat tulevat mukaan tutkimukseen luontevasti, sillä ne ovat avoimen lähdekoodikehityksen tuloksia. Piirto-ohjelmista ei yleisestikään ole tehty kovin paljon aikaisempaa tutkimusta, minkä takia aihe on kiinnostava. Piirto-ohjelmista saatu oma käyttäjäkokemus lisää myös mielenkiintoa aiheeseen. Näistä kahdesta piirto-ohjelmasta saatava tieto on otettu kyseisten ohjelmien verkkosivustoilta. (Foundation 2016; Team 2016)

Tutkielman luvussa 2 kerrotaan, mitä avoin lähdekoodi on, ja millainen avoimen lähdekoodin sovellusprojekti yleensä on. Luvussa käsitellään lisäksi avoimen lähdekoodin projekteihin osallistuvien ihmisten motivaatioita, ja pyritään luomaan jonkinasteinen kuva avoimen lähdekoodin yhteisöstä. Viimeisenä luvun sisältössä arvioidaan avointa sovelluskehitystä sen hyvien ja huonojen puolien kautta.

Luvussa 3 avointa lähdekoodikehitystä verrataan kaupalliseen sovelluskehitykseen ja pohditaan avoimen lähdekoodin ja kaupallisuuden yhdistämistä ja sen tuomia mahdollisuuksia. Lisäksi luvussa esitellään avoimeen lähdekoodiin liittyviä lisenssejä, tärkeimpänä GPL eli General Public License, pääosin niille ominaisten piirteiden kautta.

Luvussa 4 on tutkielman yhteenveto. Siinä kootaan tutkimuksen pääkohdat.

2 Avoimen lähdekoodin tyypilliset piirteet

Avoin lähdekoodi ja avoimen lähdekoodin sovellus tarkoittavat sitä, että ohjelmiston käyttäjillä on oikeus käyttää ohjelmistoa, muokata sen koodia ja toimintaa sekä levittää sitä edelleen ilman, että heidän tarvitsee maksaa näistä teoista minkäänlaista rojaltia tai maksua alkuperäiselle tekijälle (Feller 2005; Lessig 2005). Avoimeen lähdekoodiin sisältyy siis ajatus siitä, että ohjelman tai ohjelmistojen totetutustapojen pitäisi olla kaikille nähtävänä ilman rahallista korvausta. Tähän kuitenkin kuuluu käsitys alkuperäisen tekijän ”moraalisesta oikeudesta” tulla tunnistetuksi täksi (Feller 2005), joten kyseessä ei siis ole toisen tekemän ohjelman tai koodin varastamisesta.

Muun muassa Mozilla-internetselain, GNU/Linux-käyttöjärjestelmä sekä tämän tutkielman kannalta oleelliset piirto- ja kuvankäsittelyohjelmat Krita ja GIMP ovat tällaisia avoimen lähdekoodin ohjelmistoja.

Avoimen lähdekoodin sovellusprojekti voidaan määritellä seuraavasti: se on sovelluksen kehittämistä yhteistyöhön perustuvissa epävirallisissa verkostoissa, joissa kehitykseen osallistuu niin ammattilaisia kuin harrastelijoitakin. Projektin osana koodi jaetaan ilmaiseksi kehittäjille jä käyttäjille. (Gosh 2005) Avoimen lähdekoodiprojektin joitakin tunnuspiirteitä ovat modulaarisuus, panostus bugien eli ohjelman ongelmakohtien etsimiseen ja korjaamiseen, sekä konfiguroinnin ja julkaisun tarkka hallinta (Fitzgerald 2005). Vertaisarvioinnilla on myös tärkeä osa projektin hallinnassa (Fitzgerald 2005; Robbins 2005). Arviointi on aktiivista, se tehdäänkin yleensä aina ennen kuin jäsenen lähettämä koodi hyväksytään osaksi virallista ohjelmaa, ja myös hyväksynnän jälkeen. (Fitzgerald 2005)

Avoimen lähdekoodisovelluksen kehitysprojektia voidaan kuvata hyvin basaarimallin avulla (Fitzgerald 2005; Raymond 1999): käytössä ei ole mitään tiettyä toimintamallia, vaan kaikki osallistujat toimivat omalla, itselleen sopivalla tavalla. Projektissa ei ole käytössä virallista mallia estämään yhden toiminnan toteuttamista kahtena eri versiona, jolloin syntyy useita eri versioita samasta toiminnon toteutuksesta. Tämä johtaa ”evoluutioon”, jolloin parhaiten toteutettu moduuli tai ohjelma otetaan osaksi varsinaista sovellusta. Tämän voi nähdä joko hyvänä tai huonona tapana: Linuksen lain (Raymond 1999) mukaan enemmän ihmi-

siä tarkoittaa parempaa tulosta, kun taas Brooksin laki ilmaisee (Fitzgerald 2005), että jos myöhästyneeseen projektiin tuodaan mukaan lisää ihmisiä, se vain myöhästyttää projektia lisää.

2.1 Ihmiset avoimen lähdekoodin takana

Yleisesti ottaen ihmisellä on kahdentyyppisiä motivaatioon vaikuttavia tekijöitä: ulkoisia ja sisäisiä (Ryan ja Deci 2000). Yleensä on luultu, että suurimman motivaation antavat ulkoiset tekijät (tässä tapauksessa muun muassa parempiin työtehtäviin pääsy tai uralla eteneminen), mutta sisäiset tekijät, kuten itsensä ilmaiseminen ja niin sanottuun ”flow”-tilaan pääsy, nousevatkin suurempaan rooliin. (Lakhani ja Wolf 2005) Muita olennaisia motivaattoreita ovat lisäksi osallistujalla oleva käyttäjän tarve, ohjelmointitaitojen kehittäminen ja osallistumisesta saatu älyllinen stimulaatio. Sisäisiä motivaatioon vaikuttavia tekijöitä ovat nautintoon perustuvat tekijät (Lindenberg 2001), jolloin tärkeimmässä osassa on osallistujan itse tuntema tyytyväisyyden tunne, sekä sitoutumiseen ja yhteisöön perustuvat motivaattorit (Lindenberg 2001), jotka avoimen lähdekoodin sovelluskehitysprojekteissa ovat osana suuren yhteisöllisyyden vuoksi. Yhteisöllisyys tuo lisäksi mukanaan siitä syntyneet normit ja ”hakkeri”-kulttuurin. (Lakhani ja Wolf 2005)

Ulkoiset motivaattorit voivat olla esimerkiksi työstä saatava palkka (yritykset voivat hyvinkin palkata työntekijöitä osallistumaan avoimeen lähdekoodiprojektiin, jos ne itse työskentelevät avoimen lähdekoodin parissa) tai jos osallistujalla itsellä on tarve tekeillä olevalle ohjelmalle. Näitä kutsutaan hyödyiksi, jotka ovat saatavina heti. (Lakhani ja Wolf 2005) Krita-piirto-ohjelmassa päästiin vuosina 2009–2010 merkittävästi eteenpäin, kun säätiön rahankeruun avulla saatiin kokopäivätyö mahdolliseksi 24 viikon ajan (Foundation 2016). Tämä osoittaa palkan olevan joissakin tapauksissa tärkeä motivaattori. On myös olemassa ulkoisia motivaattoreita, joista saatava hyöty tulee jälkikäteen. Näihin kuuluvat muun muassa urakehityksen eteneminen ja ohjelmointitaitojen kehittyminen. (Lakhani ja Wolf 2005)

Palkkaa saavien ja vapaaehtoisien osallistujien välillä on kuitenkin eroja motivoivien tekijöiden suhteen: palkkaa saavat kokevat tärkeäksi ohjelman käyttötarpeen omiin töihinsä liittyen ja arvostavat enemmän ammatillista statusta, kun taas vapaaehtoisina työskentelevät kokevat

tärkeämmäksi taitojen parantamisen ja tarpeen ei-ammatilliseen käyttöön. (Lakhani ja Wolf 2005) Mikään yksittäinen motivaattori ei kuitenkaan nouse toista selvästi enemmän esiin, joten tästä voidaan päätellä, että avoin lähdekoodiyhteisö on heterogeeninen. Tätä puoltaa myös Gosh (Gosh 2005).

Avoimen lähdekoodiyhteisön rakennetta voi olla vaikea määritellä, koska sen toiminta ei ole rahallista, ja tämän takia informaatiota on hankala kartoittaa tarkasti. (Gosh 2005) Jollakin tasolla se kuitenkin onnistuu ja pääpiirteittäin sitä voidaan määritellä hierarkian, modulaarisuuden ja yhdistyneisyyden kautta.

Yhteisö on samalla sekä hyvin keskittynyttä että laajalle levinnyttä: pieni osa yhteisöstä tekee paljon koodia ohjelmiin, kun taas yhteisössä on suuri joukko ohjelmoijia, jotka tuottavat vain vähän koodia (Gosh ja Prakash 2000). Pienen ryhmän koodin tuottaminen ei onnistuisi ilman tätä suurempaa ryhmää, mutta tällainen rakenne, jossa on vahva johtaja ja suuri modulaarisuus, ilmenee niin yksittäisissä ohjelman moduuleissa kuin esimerkiksi versiohallinnan kaikissa projekteissa (Gosh 2005). Pienen joukon tai yhden ihmisen tärkeys näkyy välillisesti Kritan sivuilla: siellä on mainittu tärkeiden saavutusten ja kehityksessä tapahtunieden harppausten kannalta tärkeät ihmiset nimeltä, mikä kertoo pääpainon olleen enemmän näiden ihmisten harteilla kuin jonkin suuremman joukon (Foundation 2016). Lisäksi kummankin piirto-ohjelman lähdekoodissa on havaittavissa suurta modulaarisuutta (Foundation 2016; Team 2016).

Julkaisupäivien välillä tapahtuu yllättävän vähän kontaktia ja kommunikointia projektiin osallistujien välillä, mutta suurin osa yhteisön jäsenistä on säännöllisessä yhteydessä pieneen osaan muita jäseniä. (Gosh 2005) Kommunikaation vähäisyyden tuovat myös esille Khajani ja Sulaiman (Khanjani ja Sulaiman 2011). Yhteisön yhdistyneisyyden eri tasot ja sen vaihtelevuus ovat havaittavissa tässä kommunikoinnin ja kontaktin jakautuneisuudessa.

Yhteenvetona voidaan sanoa, että avoin lähdekoodiyhteisö on motivaatioiltaan heterogeeninen yhteisö, jossa korostuu modulaarisuus ja pienen aktiivisen ryhmän panostus, joskin koko yhteisöä tarvitaan projektien onnistumisen saavuttamiseksi.

2.2 Avoimen lähdekoodin sovelluskehityksen arviointia

Avoimen lähdekoodin sovelluskehitys on tuottanut monia menestyneitä ohjelmistoja (muun muassa aikaisemmin mainitut Mozilla-internetselain ja GNU/Linux-käyttöjärjestelmä), ja näistä voidaan johtaa kahteen ryhmään kuuluvia menestykseen vaikuttavia tekijöitä: ihmisiin ja ohjelmistoon liittyvät tekijät. (Weinstock ja Hissam 2005) Ihmiset, eli tässä tapauksessa ohjelmoijat, vaikuttavat avoimen lähdekoodisovelluksen mahdolliseen menestykseen omilla valinnoillaan: jos projekti ei ole itsestä kiinnostava, niin siihen ei osallistuta (Weinstock ja Hissam 2005). Yhteisön jäsenet osallistuvat siis yleensä vain sellaisiin projekteihin, joissa he näkevät jotakin tekemisen arvoista. Rahoituksella on myös joskus merkitystä, koska aina sovellus tai ohjelma ei saa kerättyä tarpeeksi kehittäjiä ilman rahoitusta ja projekti kaatuu siihen (Weinstock ja Hissam 2005; Khanjani ja Sulaiman 2011).

Ohjelmiston tulee olla myös hyvin suunniteltu ja avoimen lähdekoodin tapauksessa jaettu järkevästi moduuleihin. Tämä helpottaa uusien tekijöiden mukaantuloa, ohjelman kehittämistä pienemmissä, omalle osaamiselle sopivissa osissa (Weinstock ja Hissam 2005), sekä itsenäistä kehittämistä (Lerner ja Tirole 2005). Sopivien työkalujen tarjoaminen on myös oleellista onnistuneen ja menestyneen sovelluksen luomisessa, kuten Mozilla-internetselaimen kanssa aikoinaan huomattiin (Weinstock ja Hissam 2005). Näihin työkaluihin kuuluvat muun muassa kunnon versiohallintaohjelmistot ja bugien raportointiin tarkoitetut tietokannat (Weinstock ja Hissam 2005), ongelmien etsimistyökalut ja muu tekinen tuki, postituslistat, projektin omat verkkosivut, sekä FAQ- ja wiki-sivustot (Robbins 2005). Nämä kaikki piirteet löytyvät sekä Kritasta (Foundation 2016) että GIMP:stä (Team 2016).

Aika on myös yksi tekijä, joka saattaa vaikuttaa menestykseen positiivisesti, toisin kuin kaupallisen sovelluksen tapauksessa. Avoimen lähdekoodin sovelluksilla on mahdollisuus kypsyä, eli vaikka sovellus näyttäisi alkuun jäävän epäonnistuneeksi, se saattaakin ajan kanssa muuttua menestykseksi (Weinstock ja Hissam 2005), kuten GIMP:lle on käynyt. Kaupallisessa tilanteessa on yleensä toisin: jos jokin sovellus tai projekti näyttää epäonnistuvan, sen rahoitus lopetetaan ja projekti lakkautetaan.

Menestykseen tarvitaan siis sitoutuneet projektin johtajat, toimiva tuote eli sovellus, jonka pitäisi tarjota jokin yleinen palvelu, olla teknisesti mielenkiintoinen ja uutta tuova. Lisäksi

kehittäjien kannattaa olla myös käyttäjiä. (Weinstock ja Hissam 2005) Tapana on myös julkaista sovelluksen seuraava toimiva versio aikaisin, ja julkaista usein. Tällöin on erikseen kehitysversio tai useampi, ja yleensä vain yksi vakaa versio. (Robbins 2005) Sekä Kritasta (Foundation 2016) että GIMP:stä (Team 2016) on olemassa vain yksi vakaa versio, joka tarjotaan käyttöön.

Avoimen lähdekoodikehityksen puolestapuhujat ovat yleensä sitä mieltä, että avoin lähdekoodikehitys tuottaa laadukkaita ja luotettavia sovelluksia nopealla aikataululla, ja että sovellukset ovat helposti saatavilla ilman lisäkuluja (Fitzgerald 2005). Lisäksi vahvuuksina nähdään sovellusten turvallisuus, testauksen tiheys ja aktiivinen bugien raportointi (Khanjani ja Sulaiman 2011; Ebert 2012). Tämä on totta, mutta kehityksessä on myös ongelmia ja haasteita. Näitä haasteita löytyy niin ohjelmistotekniikan, liiketoiminnan kuin sosiokulttuuristen piirteidenkin saralta (Fitzgerald 2005). Ohjelmistotekniikan suhteen ongelmana on liian vähäinen kehittäjien taito suhteessa siihen, miten avoimen lähdekoodikehityksen kiinnostus on kasvanut. Tämä vaikuttaa olennaisesti, koska kuten aiemmin mainittiin, avoimen lähdekoodin sovelluskehityksessä on tyypillistä, että projektissa on ydinjoukko tai ihminen, joka vastaa suurimmasta osasta toteutuksesta ja sillä tavalla projektin onnistumisesta (Gosh 2005). Jos näitä henkilöitä ei ole tarpeeksi, projektit eivät onnistu.

Liiallinen modulaarisuus aiheuttaa myös ongelmia. (Fitzgerald 2005) Monet ohjelmat sisältävät liikaa sisäisiä viittauksia toisesta moduulista toiseen, ilman oikeaa tarvetta tehdä näin. Tämä luo turhaa riippuvuutta ja monimutkaisuutta ohjelmaan. Koodin laadun ylläpidossa voi olla vaikeuksia, koska mukaan pääsevät kaikki ohjelmoijat, myös sellaiset, joiden taito ei riitä tekemään tarpeeksi laadukasta koodia. (Fitzgerald 2005) Lisäksi yhteisössä arvostetaan enemmän koodin tuottamista kuin muita yhtä tärkeitä osa-alueita, kuten testaamista ja dokumentointia (Khanjani ja Sulaiman 2011; Fitzgerald 2005). Huono suunnittelu ja dokumentaatio ovat myös avoimen lähdekoodin tyypillisiä heikkouksia (Khanjani ja Sulaiman 2011).

Liiketoiminnan kannalta yksi ongelmista on se, että yhteisössä ei ole tarpeeksi kykyä tai yhtenäisyyttä tekemään strategisia päätöksiä, jotka auttaisivat viemään kehitystä eteenpäin (Fitzgerald 2005). Tämä saattaa johtua siitä, että avoimen lähdekoodin projekteissa ei ole keskitettyä johtoa (Khanjani ja Sulaiman 2011). Sosiokulttuuriset ongelmat sisältävät muun muassa

sen tilanteen, kun avoimesta lähdekoodista on tullut valtavirran ilmiö ja se on vakiintunut liikaa, joten alan tärkeimmät tekijät lähtevät kokiessaan, että siitä on tullut liian valtavirran ilmiö. Tämä laskee avoimen lähdekoodikehityksen tasoa. (Fitzgerald 2005). Muita ongelmia ovat muun muassa johtavien avoimen lähdekoodin pioneerien mahdollinen loppuunpalaminen (Fitzgerald 2005) sekä yhteisön sisäiset erimielisyydet ja jäsenten välinen kilpailu (Fitzgerald 2005; Khanjani ja Sulaiman 2011). Kommunikaatiota ei tapahdu aina riitävästi projektien aikana (Khanjani ja Sulaiman 2011; Gosh 2005), mikä voi aiheuttaa ongelmia. Lisäksi kaikkien kehittäjien kiinnostus ja/tai aika ei välttämättä riitä projektin loppuunviemiseen asti, mikä heikentää projektin ylläpitoa ja kehitystä.

Krita- ja GIMP-piirto-ohjelmien tapauksessa ongelmia ei voi olla liikaa, sillä molemmat ovat pitkäikäisiä, edelleen toimivia ja käytettyjä sovelluksia. Sovelluksille yleisinä haasteina voidaan kutenkin pitää yhteisön sisällä syntyviä erimielisyyksiä ja liiallista modulaarisuutta, joka saattaa kasvaa, jos siitä ei pidä huolta.

3 Avoin lähdekoodi ja kaupallisuus

Avoimen lähdekoodin sovelluskehityksessä on selkeä ero kaupalliseen kehitykseen monessakin eri näkökulmassa. Projekteihin osallistujat koostuvat yleensä suuresta määrästä vapaaehtoisia, toisin kuin kaupallisella puolella (Mockus, Fielding ja Herbsled 2005). Tietysti, kuten edellä mainittiin, on myös mahdollista, että avoimeen lähdekoodikehitykseen osallistuja saa myös palkkaa, mutta näiden määrä on huomattavasti pienempi vapaaehtoisten määrään (Lakhani ja Wolf 2005). Lisäksi osallistujat saavat itse valita, mitä he tekevät, eikä projekteissa ole erikseen määrätty, kuka toteuttaa mitäkin. Kehityksessä ei myöskään ole tarkkaa järjestelmätason suunnittelua tai projektisuunnitelmaa, eikä aikataulua. (Mockus, Fielding ja Herbsled 2005) Nämä kuuluvat tärkeänä osana kaupalliseen kehitykseen.

3.1 Avoin lähdekoodikehitys verrattuna kaupalliseen kehitykseen

Yleisesti avoimen ja kaupallisen kehityksen eroavaisuuksia voidaan vertailla muun muassa rahallisen hyödyn tavoittelun, kehittämisen, vastuun, ja sovelluksen vaatimusten määrittelyn osalta. Kuten voi olettaa, kaupallisessa kehityksessä tarkoitus on juurikin saada rahallista hyötyä sovelluksen myynnistä, kun taas avoimen lähdekoodikehityksen perusideana on tarjota sovellus ilmaiseksi kaikkien saataville. (Khanjani ja Sulaiman 2011) Tämä on käyttäjän näkökulmasta tärkeää: avoimen lähdekoodin sovellus saattaa vedota joihinkin kuluttajiin enemmän, koska sen voi hankkia ilmaiseksi itselleen (Krishnamurthy 2005). Tämä pätee myös piirto-ohjelmien kohdalla.

Kaupallisessa kehityksessä painotetaan kehityksessä käytettävää tarkkaa mallia ja keskitettyä johtoa, toisin kuin avoimessa kehityksessä: avoimen kehityksen pääpaino on ihmisten omissa valinnoissa eikä kenelläkään ole virallisesti erityisrooleja (Khanjani ja Sulaiman 2011; Mockus, Fielding ja Herbsled 2005). Vastuu laadusta on avoimessa lähdekoodiprojektissa koko osallistuvalla yhteisöllä, eikä vain yhdellä tietyllä yksiköllä, kuten kaupallisella puolella johdolla. Lisäksi avoimessa lähdekoodikehityksessä käyttäjien vaatimusten selvitys on helpompaa, sillä useasti sovelluksen tai ohjelman kehittäjät ovat itse myös sen tulevia käyttäjiä (Khanjani ja Sulaiman 2011; Lakhani ja Wolf 2005). Tämä ei ole niin yleistä

kaupallisten ohjelmistojen kehityksen parissa. Vaikka näiden kahden eri kehitystavan välillä on huomattavia eroja, molemmissa toteutuu kuitenkin kaikki samat kehitykseen liittyvät askeleet (Weinstock ja Hissam 2005): vaatimusten määrittely, suunnittelu, toteutus, testaus, julkaisu ja ylläpito. Molempien tavoitteena on myös luoda sovellus, joka toimii ja jota käyttäjien on miellyttävä käyttää.

3.2 Avoimen lähdekoodin ja kaupallisuuden yhdistäminen

Avoimen lähdekoodin ja sen ideologian yhdistäminen kaupallisen hyödyn tavoitteluun saattaa kuulostaa ensin oudolta, mutta tämä on onnistuttu toteuttamaan. Esimerkiksi yhtiö RedHat on yksi tunnetuimmista yhtiöistä, joiden liiketoiminta perustuu avoimeen lähdekoodiin.

Yritykset voivat käyttää avointa lähdekoodia hyödykseen niin kulujen pienentämiseksi kuin tulojen kasvattamiseksikin. Avoimien yhteisöjen kanssa tehty yhteistyö ja heidän tuottamansa koodin sisällyttäminen yrityksen omaan tuotteeseen vähentää tarvetta tehdä jo kirjoitettu koodi itse, ja siten vähentää tästä syntyviä kustannuksia (Krishnamurthy 2005). Tätä hyödyntämistä on kasvattanut se, että avoin lähdekoodikehitys usein tarjoaa ydintoimintoja standardiratkaisuina, jolloin yritykselle jää enenmmän aikaa panostaa ylätason toteutukseen, joka on heille tärkein, voittoa tuottava osa (Ebert 2012).

Yrityksillä on mahdollisuus hyödyntää avointa lähdekoodia tulojensa kasvattamiseksi muun muassa tarjoamalla tukipalveluita avoimen lähdekoodisovellusten suosion kasvun myötä. Näihin tukipalveluihin voi kuulua esimerkiksi avoimen sovelluksen käyttöönoton avustus ja asennus, ohjelman käytön koulutus ja käytön aikana tapahtuva ongelmatilanteiden tuki. (Krishnamurthy 2005) Aikaisemmin mainittu RedHat on juuri tällainen yritys, sillä se tarjoaa näitä palveluja Linux-käyttöjärjestelmää varten.

Yhteistyölle antaa myös hyvän pohjan se, että avoimen lähdekoodiyhteisön tarkoitus ei ole kilpailla kaupallisen puolen tuotteiden kanssa — yhteisön jäsenet eivät yleensä välitä siitä, tuottaako heidän sovelluksensa rahaa vai ei (Krishnamurthy 2005). Lisäksi tätä auttaa se, että avoimen lähdekoodin ohjelmat ovat ilmaiseksi saatavilla, joten niistä on helppo tehdä globaaleja tuotteita: veroista tai piratismista ei tarvitse huolehtia, kuten yleensä joudutaan tekemään kaupallisten sovellusten kanssa.

Avoimen lähdekoodin tekijät voivat kuitenkin määrätä ja hieman myös rajoittaa oman tuotteensa käyttöä lisenssivalinnalla. GPL eli General Public Licence on hyvin yleisesti käytössä, mutta tästä huolimatta avoimen lähdekoodin yhdistäminen sovellukseen ja sovelluksen myynti on silti mahdollista. (McGowan 2005; Krishnamurthy 2005) GPL:ää ja muita avoimeen lähdekoodiin liittyviä lisenssejä käsitellään tarkemmin alaluvussa 3.3.

Avoimen lähdekoodin ympärille on kehittynyt erilaisia liiketoimintamalleja, ja ne voidaan luokitella kolmeen eri luokkaan: jakelija, ohjelmistotuottaja (GPL- ja ei-GPL-malli) sekä kolmannen osapuolen palveluntarjoaja. (Krishnamurthy 2005) Jakelijamallissa yritykset, kuten RedHat, tuottavat rahaa tarjoamalla avoimen lähdekoodin sovelluksen CD:n muodossa, tukipalveluja yritysasiakkaille ja pitkäaikaista päivityspalvelua. Jotkut asiakkaat saattavat vierastaa verkosta ladattavia tuotteita, joten fyysisen kopion myynti on kannattavaa. Tukipalveluissa on se etu, että yritysasiakkaat haluavat, että tuen taso ja vastuuvelvollisuus on taattu, mikä ei välttämättä onnistu foorumeilta tai postituslistoilta apua etsittäessä. (Krishnamurthy 2005) Tähän tuen epävarmuuteen on myös hyvä varautua, jos ei aio hankkia sitä ostopalveluna toiselta yritykseltä (Ebert 2012). Päivityspalvelun avulla yritysasiakas saa saumattomasti itselleen uusimman version, eikä päivityksestä tarvitse huolehtia itse (Krishnamurthy 2005).

Ohjelmistotuottajamallissa, jossa taustalla ei ole GPL-lisenssiä, avointa lähdekoodia voidaan ottaa osaksi omaa tuotetta joko pieninä osina tai jopa kokonaisena projektina pakettikokonaisuuteen (Krishnamurthy 2005). Tässä tapauksessa oman tuotteen lähdekoodia ei ole pakko antaa muiden näkyville, sillä GPL ei ole tässä tilanteessa rajoittamassa lisenssiä (McGowan 2005). Ohjelmistotuottajamallissa, jossa on mukana GPL-lisenssi, avointa lähdekoodia voi yhä käyttää osana omaa sovellusta, mutta oma koodi on pakko laittaa julkisesti näkyville. Kaupallinen käyttö siis onnistuu yhä. (Krishnamurthy 2005) Tärkein ero näiden kahden mallin välillä on siis siinä, millä tavalla halutaan olla yhteydessä käyttäjän kanssa: ei-GPL-mallissa halutaan käyttäjän vain käyttävän ohjelmistoa, kun taas GPL-mallissa käyttäjän odotetaan olevan halukas käymään kaksisuuntaista keskustelua tuottajan kanssa.

Kolmannen osapuolen palveluntarjoajilla on vain yksi tavoite: tarjota avoimen lähdekoodin sovellukselle täysi tuki. Tässä tapauksessa ei ole väliä, miten sovellus on hankittu. Kuten edellä mainittiin, ostetuissa tukipalveluissa on yleensä laadukkaampaa ja nopeampaa palvelua, ja monesti tämänkaltaiset yritykset toimivat asiakkaan lähistöllä, jolloin paikan päällä

tehtävä avustaminenkaan ei ole mahdotonta. (Krishnamurthy 2005)

On myös mahdollista, että avoin lähdekoodiprojekti itse tarjoaa maksullisia palveluita tai versioita tuotteestaan. Näin on tehty Krita-piirto-ohjelman tapauksessa. (Foundation 2016) Krita-säätiö tarjoaa ilmaisen Krita-ohjelman lisäksi maksullista Krita Gemini -sovellusta, joka toimii tietokoneiden lisäksi myös tablettilaitteilla ja tukee kosketusominaisuutta. Gemini on yhä avointa lähdekoodia, ja sen kehittämisen tuloksia sovelletaan myös ilmaiseen Krita-versioon. (Foundation 2016) Krita-säätiöllä on lisäksi myynnissä maksullisia tutoriaaleja ohjelmiston käytöstä. GIMP-ohjelmistolla vastaavanlaisia palveluja ei ole (Team 2016), vaan varainkeruu tapahtuu lähinnä lahjoitusten muodossa.

Avoin lähdekoodikehitys on vaikuttanut kaupallisen puolen kehitysprosesseihin (Ebert 2007). Ketterän kehityksen osaaminen on kasvanut ja siitä on tullut hyväksytty kehitysmuoto myös yritysmaailmassa. Lisäksi avoimen lähdekoodin projektit tarjoavat hyvän harjoittelupaikan globaalia ohjelmistokehitystä varten. (Ebert 2012)

Ebert listaa joukon ohjeita avoimen lähdekoodin käyttäjille tai projekteihin osallistujille, joissa toistuu jo edellä mainittuja huomioita yhdistämisestä. Päätöksenteossa, miten avoin lähdekoodi otetaan mukaan tuotteeseen ja otetaanko sitä mukaan ollenkaan, pitää tarkastella valintakriteerejä selkeästi, jotta ratkaisu toimii myös kehitysvaiheen jälkeen. Avoimen lähdekoodin tarjoajan saavutettavuus on myös hyvä tarkistaa: tulevaisuudessa esiin tulevat ongelmat saadaan hoidettua helpommin, jos tarjoajalta pystyy saamaan tukea. (Ebert 2012) Tämän voi hoitaa myös ulkoisena palveluna (Krishnamurthy 2005). Lisäksi on tärkeää hallinnoida avoimen lähdekoodin käyttöä komponentin perusteellisella testaamisella ennen tuotteeseen lisäämistä, sekä ottamalla huomioon avoimen lähdekoodikomponentin yleensä hyvin tiheä päivitysväli, niin että oma tuote on toimiva (Ebert 2012).

Avoimesta lähdekoodikehityksestä on myös otettu mallia kaupallisessa kehityksessä: avoimen kehityksen tavat ja lähetymistapa kehitykseen on otettu käyttöön, mutta kohteena onkin kaupallinen sovellus, joka jaetaan suljetun yhteisön sisällä. Esimerkiksi yksi yritys hallinnoi sovelluksen kehitystä, mutta muut osallistuvat yritykset tuottavat suurimman osa koodia avoimesti. (Ebert 2012)

3.3 Avoin lähdekoodi ja lisenssointi

Avoin lähdekoodi ja sen ideologia eroavat merkittävästi kaupallisen kehityksen tarkoituksesta, joten koodiin on liitettävä omanlaisensa lisenssi eli lupa tekijältä, jonka ehtojen mukaan käyttäjällä on valtuudet koodiin. Lisenssin on siis täytettävä tietyt ehdot, jotta se voidaan laskea olevan avoimen lähdekoodin lisenssi (McGowan 2005). Lisenssissä on mainittava muun muassa seuraavat ehdot: lähdekoodi on oltava saatavilla, lisenssin pitää sallia koodin käyttö osana toista ohjelmaa, eikä se saa estää tästä tulevaa rahallista tuottoa. Myynnistä saadusta tuotosta ei myöskään saa pyytää rahallista korvausta. (Initiative 2007)

Lisenssiltä vaaditaan lisäksi yksityiskohtaisempia piirteitä: lisenssi ei muun muassa saa syrjiä ihmisiä, ryhmiä tai käyttötarkoituksen perusteella. Lisenssissä pitää myös turvata alkuperäisen tekijän oikeudet (Initiative 2007), esimerkiksi muokkaajan pitää julkaista alkuperäinen koodi ja oma koodinsa eri tiedostoissa (McGowan 2005). Avoimen lähdekoodin lisenssointi ei saa myöskään vaikuttaa toisiin sen yhteydessä jaettuihin ohjelmiin tai koodiin, jos ne vain jaetaan yhdessä ilman ohjelmien välistä vuorovaikutusta ajovaiheessa.

Yleisin lisenssi, joka toteuttaa edellä mainitut ehdot, on jo aikaisemmin mainittu GPL eli General Public License. GPL:n yksi tunnetuimmista ominaisuuksista on se, että jos käyttää GPL:n alla olevaa koodia osana omaa ohjelmaansa tai muokkaa alkuperäistä, on tämä uusi ohjelma myös GPL:n alainen (McGowan 2005), jolloin uusi koodi pitää muun muassa olla kaikille nähtävissä. Tästä käytetään termiä ”copyleft”. Lisäksi, jos uuden koodin jakaa eteenpäin, on sen mukana annettava myös kopio GPL-lisenssistä.

GPL:ssä viitataan paljon myös alkuperäisestä työstä johdettuihin töihin, joita ovat juurikin nämä ohjelmat, joihin on otettu osaksi GPL:n alla olevaa koodia. Johdannaistöiden määritelmä on helppo, jos lainattu koodi on kiinteä osa uutta ohjelmaa. Siitä, onko sellainen työ johdettu työ, joka on kokonaan omaa koodia ja se vain vuorovaikuttaa GPL:n alla olevan koodin kanssa, ollaan montaa mieltä: jotkut katsovat, että tällöin koko uusi työ on GPL:n alla, toiset eivät. (Rosen 2001; McGowan 2005)

GPL:ssä on vielä muutama mainitsemisen arvoinen piirre. GPL ei ota huomioon sitä, mitä kautta käyttäjä on saanut oikeudet itselleen: esimerkiksi jos käyttäjälle ohjelman jakanut taho menettää oikeutensa GPL:n ehtojen rikkomisen takia, tämä ei vaikuta käyttäjään, jos

hän toimii ehtojen mukaan (GNU 2007a). GPL:n vaikutus johdetun työn lisenssiin ei koske alkuperäisen työn tekijää. Tekijällä saattaakin olla yksi versio avoimena lähdekoodina ja toinen yksityisenä (McGowan 2005). GPL ei myöskään siirrä vastuuta koodin mahdollisesti aiheuttamista vahingoista alkuperäiselle tekijälle (McGowan 2005). Yritysnäkökulmasta tämä riski pitää tiedostaa ja siihen tuleekin varautua etukäteen (Ebert 2012).

GPL ei määrittele koodiin kohdistuville oikeuksille käyttöaikaa (GNU 2007a). Tämä voi kuulostaa pelottavalta, sillä periaatteessa tekijänoikeuksien haltija voi mielivaltaisesti poistaa oikeudet käyttäjiltä (McGowan 2005), mikä voi johtaa vaikeuksiin, jos on hyödyntänyt tätä koodia omassa tuotteessaan. Tämä ei kuitenkaan avoimen lähdekoodin yhteisöissä vaikuta merkittävästi, sillä jos yksi poistaisi muiden oikeudet, muut todennäköisesti tekisivät saman perässä, jolloin poistoista syntyvä ketju vain haittaisi kehittämistä ja loisi eripuraa yhteisössä. (McGowan 2005) Tämän nojalla avointa lähdekoodia hyödyntäviä liiketoimintamalleja (Krishnamurthy 2005) onkin olemassa, sillä hyödyntämisellä ei nähdä olevan liian isoja riskejä.

Joskus tilanne voi olla sellainen, että GPL ei olekaan paras ratkaisu avoimelle lähdekoodille. Näin voi olla, jos esimerkiksi halutaan tarjota kaupallisille kehittäjille mahdollisuus käyttää avointa lähdekoodia osana heidän projektejaan ilman, että heidän tarvitsee miettiä, onko heidän tekemänsä ohjelma johdettu työ ja sekoittaisiko se silloin lisenssien käyttöä. (McGowan 2005) Tällaisia lisenssejä ovat muun muassa LGPL (Lesser General Public License), BSD-lisenssit ja MIT-lisenssi. LGPL:n mukaan on olemasssa avoimen lähdekoodin kirjasto, joka itsessään sekä siitä johdettujen töiden osalta on GPL:n kaltaisten ehtojen alla (GNU 2007b). Ero tulee siitä, että jos on olemassa ohjelma, joka vain käyttää tätä kirjastoa tekemättä siihen mitään muutoksia, siihen voidaan soveltaa tämän uuden osan kehittäjän haluamaa lisenssiä. Lisenssin valinnan rajoituksena tosin on, että siinä pitää olla mahdollisuus loppukäyttäjän muokata ja takaisinmallintaa ohjelmaa debuggauksen nimissä. (McGowan 2005)

BSD ja MIT tarjoavat avoimen lähdekoodin kaikkien käyttöön sillä ehdolla, että koodin tekijänoikeudet ja takuusitoumus ilmoitetaan jakelun yhteydessä. BSD vaatii lisäksi vielä, että tekijältä on saatava lupa, ennen kuin hänen nimeään voi käyttää johdetun tuotteen mainostamiseen. (McGowan 2005)

Sekä GIMP että Krita ovat lisenssien suhteen tyypillisiä avoimen lähdekoodin sovelluksia. Molemmat kuuluvat GPL-lisenssin piiriin (Team 2016; Foundation 2016).

4 Yhteenveto

Avoimen lähdekoodin tyypillisiksi piirteiksi voidaan sanoa jo määritelmässä (Feller 2005) ilmenevät piirteet: käyttäjällä on vapaus käyttää, muokata ja levittää koodia edelleen seuraaville käyttäjille, eikä tästä toiminnasta tarvitse maksaa alkuperäiselle kehittäjälle tai kehittäjille mitään. Korostuvat piirteet ovat siis vapaus ja maksuttomuus. Lisäksi muita avoimelle lähdekoodille ja sen kehitykselle ominaisia piirteitä ovat suuri yhteisöllisyys ja modulaarisuus. Yhteisöllisyyden merkitys näkyy siinä, että avoimen lähdekoodin sovellusprojektien kehitys tapahtuu suurissa ryhmissä, joissa vallitsee omat norminsa ja toimintatapansa. Modulaarisuus puolestaan näkyy sekä yhteisössä että itse koodissa: yhteisössä on pieniä joukkoja, jotka tuottavat enemmän koodia kuin muut, ja projektissa eri osat ovat hyvin pitkälle jaettuna omiin lokeroihinsa. Avoimelle lähdekoodille on myös tyypillistä, että ohjelman koodi on vapaasti kaikkien saatavilla.

Esimerkkeinä käytettyihin piirto-ohjelmiin nämä tyypilliset piirteet peilautuvat selkeästi. Molemmat ohjelmat ovat ilmaisia, ja ohjelmien kehitys tapahtuu yhteisöissä, joissa esiintyy vahvoja johtajia, joiden panos on huomattava. Molempien ohjelmien koodi on myös hyvin modulaarista, ja koodi on julkaistu kaikkien nähtäville internettiin.

Sekä GIMP että Krita ovat menestyneitä avoimen lähdekoodin sovelluksia, joten niistä molemmista löytyy menestykseen positiivisesti vaikuttaneita tekijöitä. Molempien parissa on oletettavasti työskennellyt ohjelmista kiinnostuneita kehittäjiä, ja molemmat on jaettu järkeviin, helposti lähestyttäviin osakokonaisuuksiin, jotta kehittäjien on helpompi osallistua projekteihin. Lisäksi molemmat projektit tarjoavat riittävät työvälineet, jotta kehitys pysyy mielekkäänä. Sovellusten haasteina voidaan nähdä avoimelle lähdekoodin sovellukselle tyypilliset haasteet: liiallinen modulaarisuus ja yhteisössä syntyvät erimielisyydet.

Avoimen lähdekoodin ja kaupallisuuden suurimpia eroja ovat rahallisen tuoton tavoittelu, kehityksessä käytettävän mallin tarkkuus ja johtajuuden aste projektin sisällä. Kummassakin kehitystavassa on kuitenkin kaikki samat kehitysaskeleet, ja tavoitteena molemissa on luoda toimiva, käyttäjäystävällinen tuote. Avointa lähdekoodia on myös mahdollista hyödyntää kaupallisesti: sitä voi jakaa eteenpäin osana omaa koodiaan tai tuotteena, jonka yh-

teyteen yritys tarjoaa palveluita. On myös mahdollista tarjota pelkästään palveluita, yleensä asennus- tai tukipalveluita asiakkaille. GIMP kerää rahaa lähinnä lahjoituksina, mutta Krita myy lisäksi maksullista versiota omasta tuotteestaan ja tarjoaa kattavampaa tukea maksua vastaan.

Avoimen lähdekoodin lisenssoinnissa on käytössä yleisesti lisenssinä GPL eli General Public License. Lisenssi vaatii sen alla olevan koodin oltavan kaikkien saatavilla, mutta ei estä koodin käyttöä osana toista ohjelmaa, vaikka toisella ohjelmalla tehtäisiinkin rahallista tuottoa. GPL:n ehkä tunnetuin ominaisuus on seuraava: jos GPL:n alla olevaa koodia käyttää integroituna osana omaansa tai tekee muokkauksia GPL:n alla olevaan koodiin, on tämä uusi tuote myös GPL:n alaisuudessa olevaa koodia. Kumpikin piirto-ohjelmista ovat GPL-lisenssin piirissä.

Lähteet

Ebert, Christof. 2007. “Open Source Drives Innovation”. *IEEE Software:* 105–109.

———. 2012. *Global Software and IT: A Guide to Distributed Development, Projects, and Outsourcing.* Hoboken, New Jersey: Wiley-IEEE Press.

Feller, Joseph. 2005. *Perspectives on Free and Open Source Software.* Cambridge, Massachusetts: MIT Press.

Fitzgerald, Brian. 2005. “Has Open Software a Future?” Teoksessa *Perspectives on Free and Open Source Software,* toimittanut Joseph Feller, 95–106. Cambridge, Massachusetts.

Foundation, Krita. 2016. *Krita Website.* `http://www.krita.org/`.

GNU. 2007a. *GPL.* `https://www.gnu.org/licenses/gpl-3.0.en.html`.

———. 2007b. *LGPL.* `https://www.gnu.org/licenses/lgpl-3.0.en.html`.

Gosh, Rishab Aiyer. 2005. “Understanding Free Software Developers: Findings from the FLOSS Study”. Teoksessa *Perspectives on Free and Open Source Software,* toimittanut Joseph Feller, 23–45. Cambridge, Massachusetts.

Gosh, Rishab Aiyer, ja Vipul Ved Prakash. 2000. “The Orbiten Free Software Survey”. First Monday. Saatavilla `http://firstmonday.org/ojs/index.php/fm/article/view/769/678`.

Initiative, Open Source. 2007. *Open Source Definition.* `http://opensource.org/osd/`.

Khanjani, Atieh, ja Riza Sulaiman. 2011. “The Aspects of Choosing Open Source versus Closed Source”. *IEEE Symposium on Computers and Informatics:* 646–649.

Krishnamurthy, Sandeep. 2005. “An Analysis of Open Source Business Models”. Teoksessa *Perspectives on Free and Open Source Software,* toimittanut Joseph Feller, 279–296. Cambridge, Massachusetts.

Lakhani, Karim R., ja Robert G. Wolf. 2005. “Why Hackers Do What They Do”. Teoksessa *Perspectives on Free and Open Source Software,* toimittanut Joseph Feller, 3–21. Cambridge, Massachusetts.

Lerner, Josh, ja Jean Tirole. 2005. “Economic Perspectives on Open Source”. Teoksessa *Perspectives on Free and Open Source Software,* toimittanut Joseph Feller, 47–78. Cambridge, Massachusetts.

Lessig, Lawrence. 2005. “Open Code and Open Societies”. Teoksessa *Perspectives on Free and Open Source Software,* toimittanut Joseph Feller, 349–360. Cambridge, Massachusetts.

Lindenberg, Siegwart. 2001. “Intrinsic Motivation in a New Light”. *Kyklos:* 317–342.

McGowan, David. 2005. “Legal Aspects of Free and Open Source Software”. Teoksessa *Perspectives on Free and Open Source Software,* toimittanut Joseph Feller, 361–391. Cambridge, Massachusetts.

Mockus, Audris, Roy T. Fielding ja James D. Herbsled. 2005. “Two Case Studies of Open Source Software Development: Apache and Mozilla”. Teoksessa *Perspectives on Free and Open Source Software,* toimittanut Joseph Feller, 163–209. Cambridge, Massachusetts.

Raymond, Eric. 1999. “The Cathedral and the Bazaar”. *Knowledge, Technology and Policy:* 23–49.

Robbins, Jason. 2005. “Adopting Open Source Sofware Engineering (OSSE) Practices by Adopting OSSE Tools”. Teoksessa *Perspectives on Free and Open Source Software,* toimittanut Joseph Feller, 245–264. Cambridge, Massachusetts.

Rosen, Lawrence. 2001. “The Unreasonable Fear of Infection”. Saatavilla `http://www.rosenlaw.com/html/GPL.pdf`.

Ryan, Richard M., ja Edward L. Deci. 2000. “Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions”. *Contemporary Educational Psychology:* 54–67.

Team, The GIMP. 2016. *GIMP Website.* `http://www.gimp.org/`.

Weinstock, Charles B., ja Scott A. Hissam. 2005. “Making Lightning Strike Twice”. Teoksessa *Perspectives on Free and Open Source Software,* toimittanut Joseph Feller, 143–159. Cambridge, Massachusetts.