Matti Vuori

Avoimen lähdekoodin ohjelmien käytettävyydestä

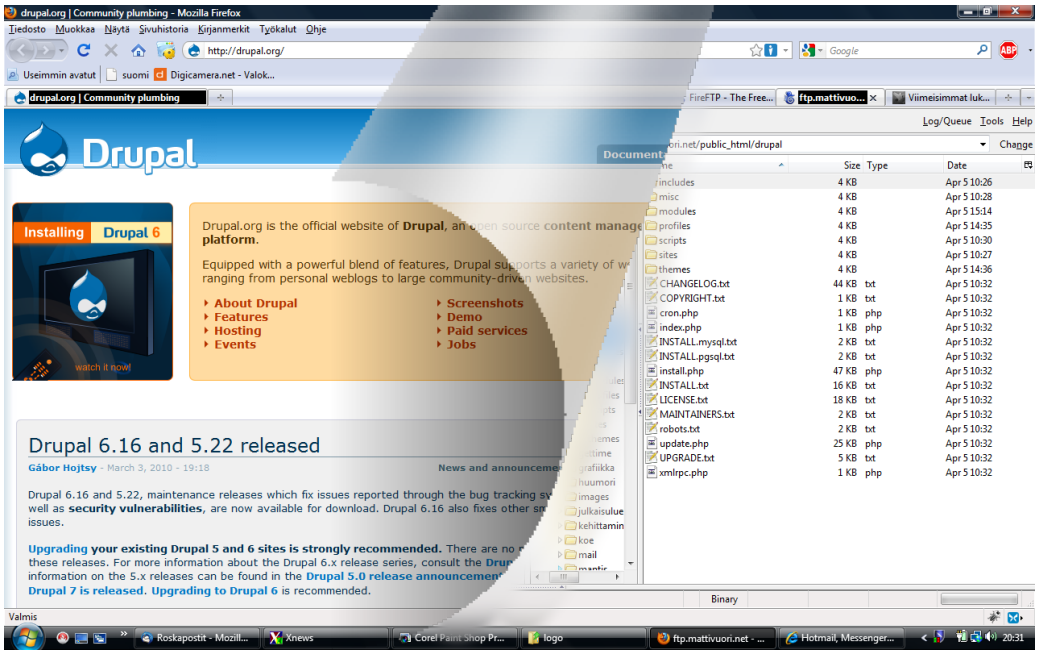
Edistääkö avoimen lähdekoodin käyttö ohjelmien käyttävyyttä vai ei? Siitä on vielä oikeastaan aikaista tehdä diagnoosia, mutta erilaisia käytettävyyttä edistäviä tai haittaavia tekijöitä voidaan tunnistaa. Edistäviä piirteitä näyttäisikin olevan monia.

Sisällysluettelo

Johdanto

Mitä se avoin lähdekoodi tarkoittaa? Ks. Wikipedian artikkeli Avoin lähdekoodi: http://fi.wikipedia.org/wiki/Avoin_l%C3%A4hdekoodi

Avoimen lähdekoodin ohjelmia on monenlaisia. Tässä tekstissä rajoitutaan stereotyyppisiin palvelinpohjaisiin tietojärjestelmätuotteisiin, joilla on laajahko kansainvälinen kehittäjäyhteisö ja myös käyttäjiä eri maissa (ensimmäisen sivun kuvassa oleva Drupal on hyvin tunnettu esimerkki sellaisesta).



Kuva 1. Drupal on menestyksekäs avoin sisällönhallintaohjelma.

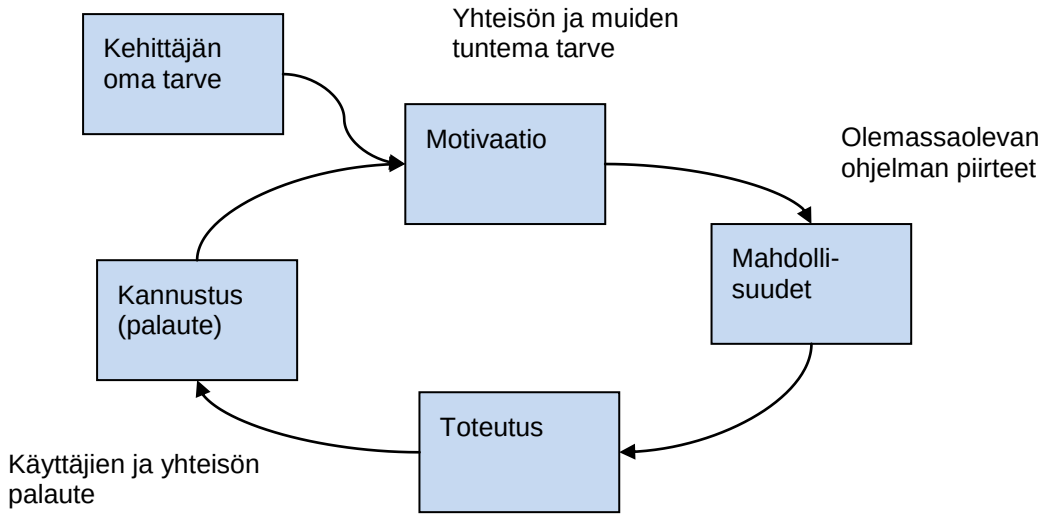
Niillä on ollut hieman huono maine johtuen ensimmäisten ohjelmistosukupolvien luontaisesta ”kotikutoisuudesta”. Nyt ohjelmistot ja tavat kehittää niitä ovat uudistuneet ja on aika esittää kysymys ”edistääkö avoimen lähdekoodin käyttö ohjelmien käyttävyyttä vai ei” ilman, että siihen on päivänselviä vastauksia. Tässä tekstissä ei tehdä mitään tilastollista tai muutakaan kattavaa analyysiä asiasta, vaan esitellään erilaisia käytettävyyttä edistäviä ja haittaavia tekijöitä.

Kyse ei ole käyttöliittymien piirteistä, vaan asioista, jotka:

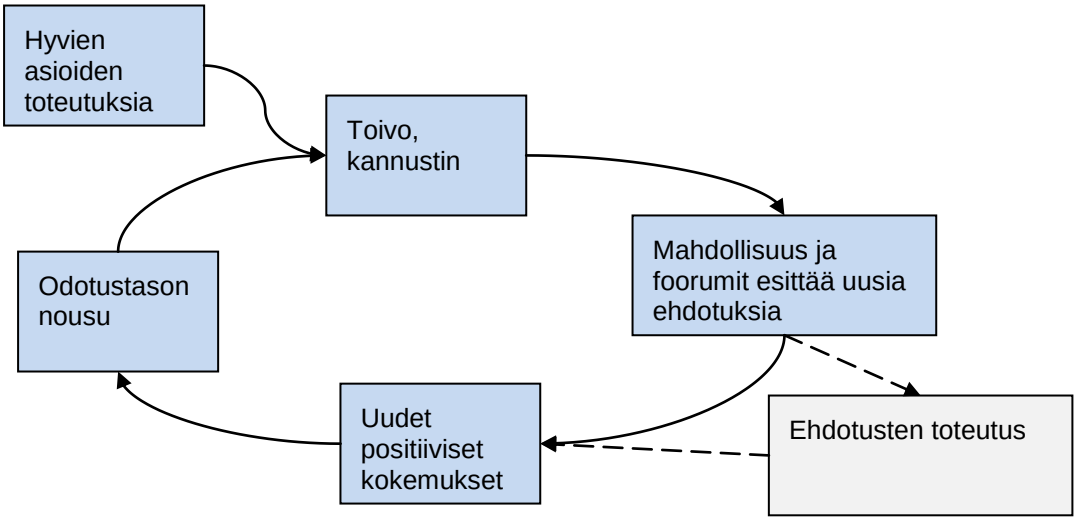
- Lisäävät motivaatiota.
- Auttavat ihmisiä tekemään hyviä valintoja.
- Edistävät hedelmällistä vuorovaikutusta ja viestintää.
- Edistävät kehittäjien oppimista.
- Nostavat vaatimustasoa.
- Edistävät tarjolla olevien valmiiden ratkaisujen hyödyntämistä.
- Saavat aikaan hyvien asioiden vakiintumista, mutta toisaalta tukevat uudistumista.
- Edistävät hyvän käytettävyyden aikaansaamista tehokkaalla ja tyydytystä tuottavalla tavalla.
- Jne…

Yhteisöjen ja käytettävyysihmisten tehtävä on sitten tukea hyviä asioita ja toisaalta pyrkiä poistamaan heikkouksia.

Avoimen kehittämisen positiiviset kierteet



Kuva 2. Kehittäjien positiivinen kierre



Kuva 3. Yhteisön positiivinen kierre

Käytettävyyttä edistäviä tekijöitä

Tavoitteet ja prioriteetit

- **Ihmiset tekevät välineitä itselleen** – toimintoja toteutetaan todelliseen tarpeeseen. Tällöin on **motivaatio** toteuttaa toiminnot siten, että ne oikeasti toimivat hyvin. Suljettuja ohjelmistoja kehitettäessä tuotetaan varallisuutta yrityksen omistajille – se ei ole kovin innostavaa.
- **Ei tarvetta rajoittaa tai rampauttaa** ohjelmaa kaupallisisista syistä. Kaupallisia ohjelmia pitää usein sovittaa tiettyyn markkinapositioon, jos valmistaja myy useaa erilaista konfiguraatiota tuoteperheenä.
- **Kehittäjät suosivat hyvää toimintaa** enemmän kuin estetiikkaa. Ei ole esimerkiksi vaaraa kelvottomista ikoniriveistä, joista ei saa selvää mikä tekee mitäkin (perinteinen ongelma perinteisissä GUI-käyttöliittymissä).
- Tavoitteena ei ole tehdä kauppaa, vaan toteuttaa hyvä ohjelma! Ei ole vaaraa, että tuotetta muutetaan ”turhaan” tai sille tehdään huonontavia imagopäivityksiä.
- Laajeneva **liiketoimintakriittinen käyttö nostaa vaatimuksia ja laatutasoa**. Ja vaatimuksista kuullaan koko ajan. Niistä keskustellaan foorumeilla eikä tieto jää tuotepäälliköille tai myyjille. Kaikki tietävät, missä mennään ja nostavat profiiliaan.

Prosessit

- **Kehittäjillä on aikaa** eri tavalla kuin suljetuilla ohjelmilla. Ei ole asiakkaan päättämiä deadlineja parin viikon päässä, vaan työt voidaan tehdä hyvin. Aikataulupaineet yhdessä heikon muutoksenhallinnan kanssa tuottavat aina huonoja ratkaisuja.
- Avoimen lähdekoodin ohjelmaa **voidaan parantaa myöhemmin** miten monta kertaa tahansa, eikä parannus tarvitse budjettia tai projektia – vain sen, että joku tekee sen ja saa myytyä muutoksen asioista päättäville (päätöksenteon organisointi vaihtelee projekteissa hyvin paljon).
- Käyttäjien **ehdotuksia voidaan ottaa vastaan ja käsitellä avoimesti**. Monilla suljetuilla ohjelmilla suljetaan tietoisesti korvat käyttäjien impulsseiltä, koska on vaarana joutua antamaan korvauksia ehdotusten hyödyntömisestä. Räätälöidyillä ohjelmilla taas silmät ja korvat suljetaan budjettisyistä.
- Ohjelmistokehitys on aina **oppimisprosessi**. Kehittäjäyhteisön oppiminen edellyttää asioiden käsittelyä, dialogia, avoimuutta. Avointen ohjelmistojen sähköiset kehittämisfoorumit ovat tässä omiaan. Suljetuilla ohjelmilla ei perinteisesti ole tällaista yhteistä oppimisfoorumia, vaikka siihen on toki muita keinoja.

Käyttöliittymien parhaat käytännöt

- Avoimessa kulttuurissa muiden avointen ohjelmien **parhaiden piirteiden kopiointi on hyve** (eikä muodosta oikeusjutun riskiä).
- **Kehittäjät voivat oppia tutustumalla muihin ohjelmiin** – suljetuilla ohjelmilla tämä on vaikeaa. Mikä tahansa ohjelma voidaan asentaa palvelimelle tai vaikka omalle läppärille ja sen toimintaan ja käyttöön voi rauhassa perehtyä.

Yhteisöllisyys

- **Yhteisön suora ja julkinen palaute** foorumeilla tai sähköpostilistoilla suljettujen neuvotteluhuoneiden sijaan. Parhaimmillaan syntyy asioita edistävää parviälyä.
- Kehittäjäyhteisön **sisäinen vuorovaikutus ja dialogi**. Suljetun ohjelmiston kehittäjät toimivat tiukoissa rooleissa ja voi olla, että tiimit eivät saa olla toistensa kanssa vuorovaikutuksessa lainkaan.
- Suljettuja ohjelmia kehitetään 2000-luvun yksilöllisessä organisaatiokulttuurissa, jossa ammattilaiset pyrkivät pimittämään tietojaan muilta edistääkseen omaa menestystään. Avoimia ohjelmia sen sijaan kehitetään kulttuurissa, jossa **avoimuus on yhteisössä menestymisen avain**.
- Käyttäjien avoin vuorovaikutus kehittäjien kanssa. Tämä on suljetuilla ohjelmilla äärimmäisen harvinaista. Käyttäjien sijaan toimitaan niiden edustajien tai kaikkitietävän tuotepäällikön kanssa.

Avoimuus ja räätälöitävyys

- Avoimen ohjelmiston täysi **avoimuus räätälöinnille**.
- Ohjelmissa voidaan käyttää edistyneitä avoimen lähdekoodin **kirjastoja** (esim. Ajax-kirjastot), joiden käytölle on esteitä suljetuissa ohjelmistoissa.
- **Konfiguroitavuus** on yleensä hyvä. Esimerkiksi Drupal-ohjelmassa voi sisältöjä kirjoittaa käyttäen erilaisia merkkaustapoja, joita voi kuka tahansa tehdä lisää tarpeen mukaan. Suljetuissa ohjelmissa olisi aina valittu ”paras” merkkaustapa. Drupalin ratkaisu mahdollistaa sen, että ohjelman installaatioon voidaan valita käyttäjien tarpeiden ja osaamisen mukaan jokin HTML- tai Wiki-tyylinen merkkaustapa.
- Kielenä on usein PHP, jolla on muita kieliä **helpompi tehdä räätälöintejä** – ei tarvitse esim. luoda erilaisia käännösympäristöjä, koska PHP on tulkattava kieli.
- Lokalisointi-arkkitehtuuri tukee räätälöintiä. Lokalisoinnin helppouteen panostetaan näissä ohjelmissa koko ajan enemmän, koska kansainvälistäminen on niiden elinehto.
- **Hajautettua kehittämistä tukeva moduuliarkkitehtuuri** mahdollistaa toimintojen laajentamisen ja toisaalta ohjelmiston konfiguraation keventämisen, mikä selkeyttää ylläpitoa ja nopeuttaa sen toimintaa.

Teknologiavalinnat

- Ohjelmissa käytetään **tuttuja teknologioita, joiden toimivuudesta on varmuus** ja joille on työkaluja. Tämän johdosta myös käyttäjille tulee vähemmän yllätyksiä. On vähemmän vaaraa, että mukaan tulee jokin kolmikirjaiminen käyttöliittymätekniikka, joka onkin sitten kömpelö ja hidas.
- Ohjelmat tehdään selain-, palvelin- ja käyttöjärjestelmäriippumattomiksi. Käyttäjäkokemus on tällöin stabiili eikä esimerkiksi tarvitse vaihtaa selainta vain sovelluksen käyttämiseksi.

Ohjelmiston arvioitavuus

- **Asiakas voi vapaasti arvioida ohjelmistoa** ennen sen mahdollista hankintaa. Kokeiluasennuksille ei ole mitään rajoitteita. Voidaan tarvittaessa tehdä hyvät **analysoinnit ja vaikka testausta** ja niiden perusteella hyvät suunnitelmat räätälöinnille ja käyttöönotolle. Suljetuilla ohjelmistoilla on aina suuria rajoituksia niiden kokeilukäytölle.

Käytettävyydelle haitallisia tekijöitä

Tavoitteet ja prioriteetit

- Yleensä ei ole **käytettävyystavoitteita** (eikä välttämättä muitakaan tavoitteita), vaan ohjelmat kehittyvät orgaanisesti.
- Kehittäjät tekevät usein ohjelmia itselleen… tämä mainittiin etuna, mutta se on myös haitta, koska silloin **on sokea ratkaisujen puutteille** ja se, mikä sopii ohjelman perinpohjin tuntevalle ohjelmoijalle ei sovi kaikille muille.

Ohjelmien suunnitteluratkaisut

- Ohjelmien **toimintojen määrä on suuri** ja kasvaa koko ajan (tätä onneksi hallitaan usein konfiguraation räärälöinnillä).
- Monet avoimet ohjelmat elävät vielä **”nörttikulttuuria”** ja esim. niiden konfigurointi tapahtuu tiedostoja editoimalla ja on siis kovin hankalaa. Ja tämä hankaluus heijastuu ohjelman asennuksen räätälöinnin kautta käyttäjille.

Prosessit

- Ei yleensä kunnollista **konseptisuunnittelua**, vaan ohjelmat kasvavat orgaanisesti ja perusasioita on vaikea muuttaa myöhemmin.
- Graafinen suunnittelu saattaa olla heikkoa, koska projekteissa ei aina ole graafikkoja.
- Ei systemaattista **käytettävyyden analysointia tai testausta**.
- **Yhteisödynamiikka** ei välttämättä tue parhaita ratkaisuja.

Sitten ne muut avointen ohjelmistojen tyypit

Muistutuksena siitä, että avoimia ohjelmistoja on monenlaisia. Seuraavilla tyypeillä on paljon huonompi ennuste:

- Hätäisesti useasta avoimesta ohjelmista integroidut kokonaisuudet. Valmistaja tekee oman käyttöliittymän liian pienin panostuksin. Mukana ei ole yhteisöä.
- Päivänperhot, jotka eivät kestä aikaa, vaan jäävät versioon 0.1. Aina kannattaakin katsella avointa ohjelmistoa ja sen kypsymistä, ennen kuin harkitseekaan sen käyttöä tai muuta hyödyntämistä tai sen yhteisöön osallistumista.
- Kaksoislisensoidut ohjelmat. Niiden avoin versio voi olla kovin rajoittunut.
- Jne…

Suurten toimijoiden projektit, kuten OpenOffice muistuttavat toisaalta enemmän suljettuja ohjelmistoja.

Johtopäätökset

Jo mittanauha kertoo, että avoimen lähdekoodin ohjelmilla on paljon potentiaalia hyvälle käytettävyydelle. Mutta on myös sudenkuoppia, joita pitää torjua.

Ai niin, ja paras uutinen on se, että edelläkuvatut asiat pätevät pitkälti (eivät siis täysin) myös muihin ohjelmistojen piirteisiin.