Odprto programje

Primož Peterlin
Univerza v Ljubljani, Medicinska fakulteta, Inštitut za biofiziko, Lipičeva 2, 1000 Ljubljana
primoz.peterlin@biofiz.mf.uni-lj.si

Povzetek

Uspeh Linuxa je postavil na glavo nekaj dogem o industriji programske opreme. Analiziramo model odprtega programja, ki ga uporablja Linux. Predstavimo, kaj je odprto programje, katere so njegove prednosti in orišemo sociološke temelje, ki vladajo v skupnosti uporabnikov razvijalcev odprtih programov. Raziščemo, kdaj je prehod na odprt model razvoja smiseln in navedemo nekaj ekonomskih modelov poslovanja v svetu odprtega programja.

Abstract

The Open Source Software

The success of Linux contradicts a few established dogmas about the software industry. The open source software model used by Linux is being analysed. We present the idea of open source software, list its benefits and outline the sociological foundations of the community of users-developers of open source software. We analyse when the transition to open source model is feasible and list a couple of open source business models.

I think there's a world market for about five computers.
(Thomas J. Watson, IBM, 1943)

Uvod

Industrija programske opreme je v pol stoletja prehodila velikansko pot. Morda edina stvar, ki v vseh teh letih ni doživela nobene spremembe, je način organiziranja dela skupine programerjev. Zaprto, tako rekoč samostansko delovanje je ostanek časov, ko so bili računalniki izjemno redki, delo na njih pa omogočeno le posebej posvečenim, ko se je ocenjevalo, da bi na svetu potrebovali sedem do deset računalnikov in ko si niti vizionarji niso upali razmišljati o računalniških mrežah, kaj šele o tem, da bo svetovno omrežje povezovalo desetine in stotine milijonov ljudi. Ti časi so mimo, in čas je, da ponovno pretehtamo tudi model samostanskega razvoja programov. Predstavljeni odprti model razvoja programja ni le drobna tehnična izboljšava, ampak po petdesetih letih prva velika sprememba v načinu organiziranja dela.

Se je Brooks zmotil?

Linux in drugi sorodni projekti so velik izziv za teoretike in izvajalce načrtovanja dela projektne skupine za razvoj programja. Kako lahko skupina študentov, ki se poznajo le z interneta in niso organizirani v nobeno formalno strukturo, v prostem času napiše operacijski sistem, ki se po zmogljivosti in stabilnosti kosa z izdelkom multinacionalke s proračunom manjše države?

Zakaj izziv? Fred Brooks v svoji knjigi The Mythical Man-Month (Brooks 1975) sočno povzema konvencionalno modrost o načrtovanju programskih projektov: "Če projektu, ki zamuja, priključimo nove sodelavce, bo le še bolj zamujal." V svetu konvencionalnih modelov vodenja je praktično nemogoče, da bi na videz kaotično organizirana skupina nekaj sto programerjev, h kateri dinamično pristopajo novi člani, sploh kdaj karkoli napravila. Toliko manj, če je ta skupina organizirana brez formalnega vodje, hierarhije ali strukture, brez formalnih zadolžitev in odgovornosti. Programski projekti, znani kot prosto programje (free software) ali programje z dostopno izvorno kodo (open-source software), torej očitno delujejo po drugačnih načelih. Uspeh in publiciteta, ki so je ti projekti deležni zadnjih nekaj let, nas silijo, da si ta načela podrobneje ogledamo.

Kaj sploh so odprti programi?

So to programi, ki jih lahko uporabljamo, ne da bi nam bilo treba za to plačati?

Uporaba odprtih programov res ne terja plačila. Vendar pa lahko tudi Microsoft Internet Explorer uporabljamo, ne da bi nam bilo treba za to plačati. Microsoft ponuja neokrnjeno različico programa na svoji spletni strani in vsak, ki želi, jo lahko presname na svoj disk in uporablja. Vendar Internet Explorer kljub temu še ni odprt program. Dobimo namreč

samo prevedeno kodo, izvorne pa ne. Zato smo ne glede na to, ali znamo ali ne, prikrajšani za to, da bi odpravili napako, dodali kaj novega k programu ali ga priredili tako, da bi deloval na drugem operacijskem sistemu, in za to, da bi izboljšano izvedbo programa delili z ostalimi. Še več, pravzaprav niti prevedene kode ne smemo deliti, saj nam kot uporabniku pogodba dovoljuje le, da ga shranimo na disk in uporabljamo.

Prostost ni stvar cene. Nič nenavadnega se nam ne sme zdeti, da moramo za odprte programe plačati. Del denarja bo pri tem ostal v trgovini, del ga založnik porabi za kritje svojih stroškov, del pa ga, če da kaj nase, nameni tudi avtorjem programja na CD-ju.

Dobimo z odprtimi programi tudi izvorno kodo?

Razpoložljivost izvorne kode je res nujen pogoj, da je program odprt, sama po sebi pa še ne pomeni odprtosti programov. Znan tak primer, ki ga verjetno poznajo mnogi študenti naravoslovno-tehničnih fakultet, je zbirka podprogramov iz učbenika Numerical Recipes. Res je dostopna z izvorno kodo (in njeno razlago), a tudi z nadvse restriktivnim dovoljenjem za uporabo. To nam sicer dovoljuje vgradnjo podprogramov iz zbirke v svoje programe in razširjanje le-teh, a le, če ne objavimo izvorne kode programa. Kljub temu, da so dostopni z izvorno kodo vred in da za njihovo uporabo ni treba posebej plačevati, ti podprogrami niso prosti, saj je njihova uporaba dovoljena le tistim, ki so kupili učbenik.

Kaj so torej odprti programi?

Na kratko: odprti programi so tisti programi, ki uporabniku dovoljujejo uporabo, razmnoževanje, razširjanje, razumevanje delovanja, spreminjanje in izboljševanje programa. Natančneje pa lahko kot odprto programje opredelimo vse programe, ki izpolnjujejo naslednje pogoje:

- *Dovoljenje za uporabo brez dodatnih omejitev.* Dovoljenje za uporabo mora zadostovati za uporabo programa, ne da bi moral uporabnik pristajati na dodatne omejujoče pogoje. Če je program del paketa, dovoljenje za uporabo ne sme omejevati pravic do uporabe programa zunaj tega paketa.
- *Prosto razširjanje.* Dovoljenje za uporabo programa ne sme preprečevati nadaljnjega razširjanja programa. Uporabnik sme program deliti brezplačno ali ga prodajati, ga vključevati v zbirke ali distribucije programov in za to ni dolžan plačevati odškodnine avtorju programa.
- *Dostopnost izvorne kode.* Program mora biti dostopen v izvorni kodi, sme se ga razširjati tako v izvorni kot v prevedeni obliki. Če se razširja le v prevedeni obliki, mora biti jasno označena možnost brezplačnega dostopa do izvorne kode. Izvorne kode tudi ni dovoljeno pisati namenoma nepregledno.
- *Izvedena dela in integriteta avtorjeve kode.* Program se sme spreminjati, izvedena dela pa se smejo razširjati pod enakimi pogoji kot izvirnik. Avtor lahko zahteva, da se spremembe in dopolnitve razširjajo kot popravki izvorne kode ločeno od izvirnika in jih uporabnik vključi pred prevajanjem programa.
- *Enakopravnost uporabnikov in načinov uporabe.* Dovoljenje za uporabo programa ne sme razlikovati uporabnikov ali skupin uporabnikov niti ne sme omejevati načinov uporabe programa.

Zakaj tako? Dovoljenje za uporabo odprtega programja ne varuje avtorja, temveč proces nastajanja odprtega programja. Zagotavlja, da bo programje dostopno za neodvisni zunanji pregled in nadaljnji razvoj. Moč odprtega programja je v kreativnem potencialu najširše baze uporabnikov razvijalcev, ki je s prvotnim avtorjem izenačena glede dostopa do izvorne kode programa, in spodbujana k spremembam in dopolnitvam programa.

Kaj pomeni Open Source, Freeware, GPL, Copyleft, Public domain, Shareware?

Najbolj preprosto avtor spravi svoj program v obtok tako, da ga izroči v javno last (angl. public domain). Z zahtevo, da se ga navaja kot avtorja programa, obdrži moralne pravice, lahko pa se navajanju tudi odpove. To omogoča, da program kdorkoli uporablja in razširja s svojimi izboljšavami vred. Program v javni lasti nima vgrajene varovalke pred tem, da si program kdo prilasti – ga spremeni ali dopolni in ga razširja naprej pod drugačnimi, bolj omejujočimi pogoji. Kdor prejme program v takšni spremenjeni obliki, z njim ne prejme tudi svoboščin, ki mu jih je namenil prvotni avtor, te mu je odvzel posrednik.

Ob zagonu projekta GNU sredi osemdesetih let je zato fundacija za prosto programje (Free Software Foundation; FSF) poskusila nadomestiti pomanjkljivo pravno varstvo programja v javni lasti z bolj zavezujočo pogodbo. Temeljno vodilo je bila zagotovitev enakih pogojev za vse uporabnike. Zasnovo so poimenovali copyleft (besedna igra z izrazom copyright); po njej mora vsak, ki razširja programje v spremenjeni ali

nespremenjeni obliki, z njim razširjati tudi pravice in svoboščine, ki jih je prejel, vključno s pravico do nadaljnjega razširjanja in spreminjanja. FSF je za programje, ki se razširja pod takimi pogoji, skovala izraz prosto programje (angl. free software). Slovenski izraz je na srečo jasnejši od angleškega in bralcem ni treba posebej pojasnjevati, da nosi v tej zvezi "free" enak pomen kot v "free speech", ne pa kot v "free beer" (Williams 2002).

Copyleft je splošni koncept, dovoljenje za razširjanje in razmnoževanje je GNU General Public License (Free Software Foundation, 1991) ali na kratko GPL. Vsakemu programu iz projekta GNU je priložen izvod tega dokumenta.

Copyleft omogoča, da bo program, izdan pod temi pogoji, ostal vselej dostopen pod enakimi pogoji. Če smo na primer v urejevalnik GNU Emacs dodali možnost vključevanja slik v besedilo, tako spremenjenega programa ne moremo ne brezplačno in ne proti plačilu zapreti in ponujati samo prevedene kode. Lahko ga le ponudimo naprej pod enakimi pogoji, kot smo ga sami dobili, z izvorno kodo vred. Rezultat je to, za kar si prizadeva FSF: širitev baze prostega programja.

Copyleft in GPL nista edina načina za objavo odprtega programja. Nekateri drugi avtorji svoje sicer odprto programje izdajajo pod ohlapnejšimi pogoji, ki dopuščajo tudi komercialno izrabo. Verjetno najbolj znana takšna primera sta operacijski sistem BSD Unix in okensko okolje X Window System. Enega in drugega lahko dobite kot odprto verzijo, obstajajo pa tudi komercialne. Na nekaterih delovnih postajah in grafičnih karticah tečejo izključno komercialne izvedbe X Window System. Če smo lastnik ene takih, potem za nas X Window System ni odprto programje.

Proti radikalni načelnosti fundacije za prosto programje se je v začetku 1998 postavila bolj pragmatična iniciativa za odprto programje (Open Source Initiative 1998), ki je v svojih vrstah zbrala nekaj vidnejših akterjev iz skupnosti Linuxa. Skupini imata iste cilje, vendar uporabljata drugačno retoriko. Medtem ko fundacija za prosto programje poudarja načelne vidike, se je iniciativa za odprto programje osredotočila na rezultate, kar se je pri komunikaciji s poslovno sfero izkazalo kot uspešnejše.

Vse zgoraj naštete zvrsti programja spadajo med odprto programje: programi v javni lasti, prosti programi, izdani pod pogoji copylefta (med njimi so vsi programi, izdani pod pogoji GNU GPL), in odprti programi, izdani pod drugačnimi pogoji. Skupina programov, ki ne ustreza prej zapisanim petim pogojem za odprto programje, sodi v skupino zaprtega (angl. proprietary) programja. Mednje spada večina komercialnega programja ter programov na pokušino (angl. shareware). Slednji sicer navadno dovoljujejo prosto razširjanje, praviloma pa niso dostopni v izvorni kodi in avtorji za njihovo uporabo zahtevajo plačilo.

Pregled licenc

Nekaj licenc, ki jih je iniciativa za odprto programje odobrila kot primerne:

- **GNU General Public License (GPL)**. Restriktivna in "dedna" – prenaša se na izvedena dela. Če v svoj program vgradimo kos programja, izdanega pod GPL, pade tudi naš program pod to licenco.
- **GNU Lesser General Public License (LGPL)**. Nekdaj imenovana GNU Library General Public License. Dopušča povezovanje knjižnic, izdanih pod LGPL, z drugimi programi, izdanimi pod komercialnimi licencami.
- **BSD License**. Permisivna, vztraja samo pri navedbi avtorstva.
- **X Consortium License**. Permisivna, glavni namen je popularizirati paket.
- **Artistic License**. Permisivna, pod določenimi pogoji dovoljuje tudi razširjanje le prevedenih programov.
- **Mozilla Public License (MPL)**. Podobna kot GPL, vendar ni "dedna" – omogoča dodajanje ne-prostega programja. Ni isto kot Netscape Public License, ki postavlja razvijalce v neenakopraven položaj glede na Netscape Communications Corp.
- **Q Public License**. Troll Tech izdaja knjižnico Qt pod dvojno licenco – QPL izključno za nepridobitno uporabo in komercialno licenco za pridobitno rabo.

Še druge licence, ki jih iniciativa za odprto programje priznava kot primerne, so: Academic Free License, Apache Software License, Apple Public Source License, Attribution Assurance Licenses, Common Public License, Eiffel Forum License, IBM Public License, Intel Open Source License, Jabber Open Source License, MITRE Collaborative Virtual Workspace License (CVW License), Motosoto License, Nethack General Public License, Nokia Open Source License, OCLC Research Public License, Open Group Test Suite License, Open Software License, Python license (CNRI Python License), Python Software Foundation License, Ricoh Source Code Public License, Sleepycat

License, Sun Industry Standards Source License (SISSL), Sun Public License, Sybase Open Watcom Public License, University of Illinois/NCSA Open Source License, Vovida Software License v. 1.0, W3C License, wxWindows Library License, X.Net License, Zope Public License in zlib/libpng license.

Kdo piše odprte programe? So to hekerji?

Da in ne, odvisno od tega, kaj razumemo pod izrazom heker. Večinoma se ta izraz povezuje z računalniškim kriminalom in vdiranjem v računalniške sisteme. Vdiralci v sisteme ne pišejo odprtih programov in navadno tudi nobenih drugih ne. Z redkimi častnimi izjemami namreč večina vlomilcev v računalniške sisteme pravzaprav sploh ne zna posebej dobro programirati in za vlamljanje večinoma uporabljajo programe, ki jih je napisal kdo drug.

V prvotnem pomenu izraz heker označuje zanesenjaka, ki živi in diha za programiranje in računalnike (Levy 2001, Raymond 1996, Himanen 2001). Odličnost pri pisanju programov je edini način za doseganje ugleda v tej skupnosti; položaj, naziv, starost in spol so drugotnega pomena, če sploh kaj veljajo. Hekerji so anarhistično svobodnjaška druščina, stara skoraj toliko kot računalniki sami. Izhodišče hekerstva je, da je pravica do programiranja neodtujljiva človekova pravica. Računalniki so koristni in dostop do njih mora biti neomejen za vse, ki si tega želijo. Vse potrebne informacije, ki jih potrebujemo za programiranje — izvorna koda, dokumentacije in podobno — morajo biti dostopne. Če kot hekerje razumemo te druge, potem drži, da odprte programe pišejo hekerji.

Prednosti odprtih programov

Odprti programi imajo glede na klasičen model razvoja programov nekaj prednosti:

- *Eksponentna rast.* Odprti programi so zares zaživeli šele z internetom, ki je povezal nadkritično maso ljudi. Njihovo število je dovolj veliko, da imajo realno možnost za uspeh tudi specializirani projekti, ki sicer zanimajo manj ljudi. Res veliki projekti pa v svoji kategoriji pritegnejo večino kreativnega potenciala.
- *Dolgoročna kredibilnost.* Če ima na milijone ljudi izvorno kodo določenega programa, je manj verjetno, da bo program čez noč izginil. Na primer: manj verjetno je, da bo nenadno izginotje doletelo spletni strežnik Apache kot pa urejevalnik WordPerfect. Tudi če bi celotna ekipa, ki vodi razvoj strežnika, čez noč iz neznanih razlogov izgubila zanimanje zanj in ga zapustila, se bi prej ali slej našel kdo, ki bo vzel v roke izvorno kodo in nadaljeval, kjer je bilo njeno delo končano. Če razvijalec opusti zaprt projekt, je z njim bolj ali manj konec.
- *Vzporedno odpravljanje napak.* Raymond je poimenoval to Linusov zakon: "Pri dovolj opazovalcih so vse napake v programu očitne." Ali kot je formuliral Jeff Dutky: "Odpravljanje napak je moč paralelizirati." Ta stopnja je morda ključna, zakaj model odprtega programja kljubuje Brooksovemu zakonu. Množičnost pomeni, da ni nujno, da tisti, ki napako najde, ve tudi, kako jo odpraviti, vendar se potem, ko je napaka diagnosticirana, prej ali slej najde kdo, ki to zna. Množičnost zagotavlja tudi, da je čas med objavo odkritja napake in predlogom za njeno rešitev bistveno krajši kot pri komercialnih ponudnikih programja.
- *Vzporedni razvoj.* V množici programerjev na internetu, ki samo čakajo na primeren izziv, je razvoj programov takorekoč zastonj, ekonomsko je upravičen hkratni razvoj več prototipnih rešitev. Tako trenutno na primer poteka razvoj namizij KDE in GNOME, dveh zvočnih podsistemov za Linux (OSS in ALSA) itn.
- *Odprto programje je recenzirano.* V znanosti so objave v vseh pomembnih revijah recenzirane. Preden gre članek v objavo, ga navadno pregleda več neodvisnih strokovnjakov ter poskuša v njem najti napake v razmišljanju in opozori avtorja na mogoče izboljšave. Šele po tem postopku gredo članki v objavo. Dostopnost izvorne kode odprtih programov omogoča enak postopek – kdorkoli se lahko poglobi v ustroj programa ter opazi morebitne napake ali pa predlaga izboljšave. Internet omogoča, da je med vsemi, ki si lahko ogledajo kodo, tudi dovolj strokovnjakov. Programi, katerih izvorna koda je dostopna dovolj dolgo, lahko tako doseže zavidljivo raven zanesljivosti. Primer za to sta stavni program TeX in pripadajoči program za izdelavo pisav Metafont, ki izvirata iz druge polovice sedemdesetih let. Njun avtor, profesor Donald Knuth, vodi tudi zanimiv način nagrajevanja poročil o napakah. Opis doslej še neodkrite napake v programu je prinesel najditelju napake ček profesorja Knutha za sprva simboličen 1 cent. Sorazmerno z dozorevanjem kode se je podvo-

jevala tudi nagrada. Za dele programov TeX in Metafont, napisane pred letom 1989, trenutno znaša 327 dolarjev in 68 centov.

Eden pogostih strahov v zvezi s projekti odprtega programja je nenadomestljivost avtorja: kaj se bo zgodilo z Linuxom, če se ga Linus Torvalds nekega dne naveliča? Kot podkrepitev navajajo obdobje v letu 1998, ko je zaradi Torvaldsove preobremenjenosti, osebnih razlogov in pomanjkanja časa razvoj Linuxa za krajši čas nekoliko zastal. Strah ni povsem odveč, ni pa vezan le na odprte projekte: kaj se bo zgodilo, če David Cutler, idejni oče Windows NT, po petnajstih letih spet zamenja delodajalca? Z izkušnjami, pridobljenimi v kriznem obdobju, je skupnost razvijalcev Linuxa razvila nekoliko stabilnejši način vodenja projekta. Novi koncept je uspešno prestal ognjeni krst jeseni 2001, ko je Alana Coxa kot človeka na poziciji številka dve brez večjih pretresov nadomestil Marcelo Tosatti. Na čelu večine večjih odprtih projektov so skupine, kar zmanjšuje pomen posameznika. Za manjše projekte to ne velja, vendar pa je te zaradi manjše kompleksnosti tudi lažje "posvojiti", če jih prvotni avtorji zapustijo.

Drugi strah je strah pred fragmentacijo, h kateri navidez nezadržno vodijo odprte licence. Model odprtega programja je za cepljenja in drobljenja odpornejši, kot se zdi. Linux s popolnoma odprtim razvojem lahko v tem pogledu primerjamo z drugim odprtim Unixom – BSD, ki je sicer tudi prosto dosegljiv v izvorni kodi, a ga razvijajo zaprte skupine. Linux je po skoraj desetih letih ostal enoten (imamo sicer nepreštevne "distribucije", vendar gre pri tem samo za garniranje istega skupnega jedra). BSD se je razcepil na 386BSD, BSDI, FreeBSD, NetBSD in OpenBSD. Prvi je medtem že izumrl, razvoj drugih pa poteka ločeno.

Kdaj lahko uspe projekt odprtega programja?

Redki odprti projekti zrastejo do razsežnosti, kot je na primer Linux. Navadno en (npr. Samba), dva (npr. KDE in GNOME) ali kvečjemu nekaj programov zasede določeno nišo, velika večina projektov pa zamre. Katere pogoje mora torej izpolnjevati projekt, da zaživi?

- Vsebina mora biti obenem dovolj dostopna ter dovolj zanimiva za razvijalce. Programi za krmiljenje manevrirnih raket ali nadzor jedrskih central ne zadoščajo prvemu pogoju, ker običajno dijaki in študenti nimajo doma ne enega ne drugega, da bi se lahko sami poizkušali v programiranju. Na drugi strani pa program za vodenje saldokontov ne predstavlja dovolj velikega izziva, da bi pritegnil širše množice. Za razliko od teh dveh pa npr. prevajalnik za Javo ali program za zapis MP3 zadovoljujeta oba pogoja.
- Pomembna je komunikacija z uporabniki. Pri uspešnih projektih vodje obravnavajo uporabnike kot sorazvijalce. Po elektronski pošti vse, ki so karkoli prispevali k projektu ali izrazili zanimanje zanj, obveščajo o kasnejših razvojnih različicah. Prednost odprtih programov je tudi njihova osebna nota – navadno se lahko obrnemo neposredno na avtorja namesto na posrednike. Potencialnega razvijalca v slabo voljo nič ne spravi tako kot to, da ne dobi nikakršnega odziva na svojo pošto s popravkom, ampak le-ta brez sledu ponikne na oddelku za stike s strankami. Tudi vnaprej pripravljeni odgovori v smislu "Spoštovani XY, prejeli smo popravke, ki ste jih poslali dne tega in tega in jih posredovali našemu razvojnemu oddelku. Hvala lepa in na svidenje" so komaj kaj boljši.
- Pogosto izdajanje novih, izpopolnjenih različic. Linux se razvoja z dinamiko, ki jo z zavistjo spremljajo komercialni proizvajalci programske opreme. Ključni prispevek k takšni dinamiki je dal avtor. V najbolj dinamičnih časih razvoja je tudi nekajkrat na teden izdal novo različico jedra, v kateri je upošteval do takrat prejete popravke in dopolnitve. S tem je določal in obdržal dinamiko razvoja.
- Uporabniki razvijalci na začetku potrebujejo nekaj, s čimer se da že kaj početi. Projekta ne moremo začeti tako, da v novičarski skupini razpredamo o tem, kakšen program bi radi napisali. Ljudem moramo ponuditi kaj, kar jih prepriča po eni strani o resnosti naših načrtov in po drugi strani o potencialih zastavljenega projekta, v katerih morajo videti tudi možnost za sodelovanje. Ena izmed glavnih kritik za začetne težave ob odprtju brskalnika Mozilla je bila ta, da je bila ob izidu zadeva neuporabna in se ni niti prevedla brez napak.
- Neposesivnost glede kode. Pomembno je, da si vodja projekta ne lasti kode. To pomeni, da zna prepoznati in sprejeti boljše rešitve, četudi na škodo svojih in da bo znal najti sposobnega naslednika, če prvotni avtor izgubi zanimanje za projekt ali ga ne bo več zmožen voditi.

Specifika programja kot ekonomske dobrine

Pojdimo od sociologije k ekonomiji in k tretjemu Raymondovemu članku (Raymond 1999). Enako kot drugi izdelki imajo programi dve ločeni zvrsti ekonomske vrednosti: prva je njihova tržna (vrednost izdelka kot končne dobrine), druga pa uporabna vrednost (vmesna dobrina). Zgledovanje po drugih industrijskih panogah pri analizi ekonomike pisanja programov zapelje v dve predpostavki:

- Programerji so večinoma plačani za ustvarjanje tržne vrednosti.
- Tržna vrednost programa je premosorazmerna razvojnim stroškom in njegovi uporabni vrednosti.

Nobena ni točna. Prva ne drži, ker so programi za nadaljnjo prodajo le zelo majhen del vsega napisanega programja – Raymond navaja oceno 5 % (Raymond 1999). Večino svojih delovnih ur prebijejo programerji ob pisanju in vzdrževanju programov, ki nikoli ne ugledajo luči sveta, ampak so zgolj za interno rabo.

Druga predpostavka v resnici velja za mnoge druge dobrine. Opazovanje obnašanja kupcev nas uči, da je pri programih drugače. Za primer vzemimo programski izdelek, katerega proizvajalec je propadel, ali pa manj dramatično – izdelek, ki ga je proizvajalec opustil. Vloženi razvojni stroški se s tem niso nič spremenili, niti se ni spremenila teoretična uporabna vrednost izdelka. Kljub temu zdrsne tržna vrednost takega izdelka proti nič. Kako to? Pri nakupu vrtne škropilnice ali šampona za lase kupci navadno niso posebej pozorni na to, ali ta model izdelovalec še izdeluje. Mnogi izdelki po propadu izdelovalca med zbiratelji dosežejo celo višje cene. Programi nikoli.

Kupci večinoma kupujejo program zaradi pričakovane zagotovitve vzdrževanja, kar ni obsega le tehnične pomoči, temveč tudi izdajo popravkov, gonilnikov za novejše naprave ipd. Industrija programske opreme je torej večinoma storitvena dejavnost, ki živi v iluziji, da je proizvodna dejavnost.

Če vzamemo ustaljeno vrednost, da je v tipičnem življenjskem ciklu programskega izdelka 75 % vseh stroškov vzdrževanje, odprava napak in izdelava razširitev, to pomeni, da je običajna tržna shema z visoko začetno ceno programa in brezplačno podporo slaba za obe strani: za kupca, saj v centrih za pomoč uporabnikov, ki v tej shemi niso dobičkonosni, praviloma pristanejo manj sposobni strokovnjaki, ki se jim da na voljo le najnujnejše vire.

V večini primerov je slaba tudi za proizvajalca. Financiranje podpore iz enkratnega zneska začetne cene je finančno smiselno samo pri hitro razvijajočem se trgu. Ko se trg ustali, je neogibno krčenje stroškov na račun podpore. Rezultat je enak najsi se to zgodi neposredno – z opustitvijo izdelka – ali posredno – tako, da je težko dobiti podporo za izdelek. Ker smo s tem izdelku uničili pričakovano prihodnjo vrednost, kupci uidejo drugam. Nekaj časa lahko sicer kupcem popravke predstavljamo kot nov izdelek, trajna rešitev pa je samo ena: pridobiti si moramo monopolni položaj na tržišču. V tem primeru kupci nimajo kam uiti. Zgodovina dovolj zgovorno postreže s primeri izdelkov, ki so bili kljub začetnemu solidnemu drugemu mestu izrinjeni iz niše: DR-DOS, WordStar, Quattro Pro ipd.

Trajnostni razvoj

Ali je mogoč trajnostni razvoj odprtega programja? Ali obstaja nevarnost, da bodo iz skupnega fonda odprtega programja želeli vsi samo črpati, nihče pa ne bo prispeval?

Nekateri si fond odprtega programja predstavljajo kot skupni pašnik, kjer vaščani pasejo živino. Paša mora biti zmerna, sicer pašnik zmendrajo v blaten dol, ki se le počasi spet zaraste. Tu imamo klasičen primer zapornikove dileme (Axelrod 1985): za vaščane kot celoto je seveda najbolje, če so pri izkoriščanju pašnika zmerni, medtem ko je odločitev posameznika ravno nasprotna: čimprej mora čimveč živine poslati na pašnik, dokler ta še ni zmendran v blato. Če je ne bo sam, jo bo sosed, ki razmišlja enako.

Opažanja kažejo, da izbrana ilustracija zgreši bistvo. Prvič, priliv odprtih programov se ves čas povečuje in ne zmanjšuje. In drugič, programov z rabo ne obrabimo. Ravno nasprotno, če jih odpremo, s tem spodbudimo druge uporabnike razvijalce, da prispevajo svoje popravke in dopolnitve. V zgornji prispodobi bi to pomenilo, da bolj ko pasemo, bolj sočna in zelena je trava.

Pa vendar, denimo, da smo napisali popravek, s katerim odpravimo nadvse nadležno napako v programu, ki ga uporablja veliko ljudi. Mnogi od njih se strinjajo, da ima popravek določeno tržno vrednost. Kako pobrati denar? Težava ni le v tem, da imajo vsi obstoječi sistemi prenosa denarja dosti prevelike lastne stroške, da bi bila takšna mikroplačila izvedljiva. Težava je, da je pravzaprav zelo težko zares oceniti vrednost takega dela.

Kaj lahko torej storimo? Popravek lahko ljubosumno hranimo zase ali pa ga zastonj prispevamo v

skupni sklad odprtega programja. V prvem primeru ne pridobimo nič. V drugem primeru morda tudi ne, morda pa bomo s svojim dejanjem spodbudili še koga drugega, da bo prispeval kak drug popravek, ki nas muči. Teorija iger pokaže, da je v takem idealiziranem primeru druga možnost, čeprav navidez altruistična, v resnici optimalna.

Ker se programi ne obrabljajo, je v tovrstnem sodelovanju pravzaprav pomembno le število tistih, ki prispevajo k projektu. Sorazmerno narašča tudi zahtevnost vodenja projekta. Naraščajoče število tistih, ki iz skupnega fonda samo črpajo, resda povečuje delež neprimernih vprašanj na dopisnih listah ali v novičarskih skupinah, kar pa lahko odpravimo tako, da sestavimo spisek pogosto zastavljanih vprašanj z odgovori in ignoriramo tiste, ki so vprašali, ne da bi ga prebrali. Pozitivni prispevek prvih je torej linearen, (negativni) prispevek drugih pa konstanten.

Kaj varujemo z zaprto kodo?

Denimo, da imamo napisan računovodski programski paket. V računovodstvu ni nobenih posebnih algoritmov, ki bi jih kazalo skrivati. Možna odgovora, kaj varujemo z zaprto kodo, sta torej dva: s tem varujemo njegovo tržno vrednost, ali pa ne želimo, da pride izvorna koda v roke konkurenci.

Prvi argument je veljaven za tisti manjšinski delež programov, ki so v resnici namenjeni trgu, ne pa za programe, ki jih uporabljamo sami.

Drugi argument je težji. Konkurenca res dobi izvorno kodo našega programa, namesto da bi jo morala razviti sama, vendar pa lahko na ta način pridobimo tudi nove zunanje sodelavce, ki nam bodo morda pomagali s popravki in dopolnitvami. Premisliti moramo torej, ali dobre plati odtehtajo slabe.

Kot tretji razlog mnogi navajajo tudi varnost, posebej če gre za kriptografske metode. Razmišljanje "obskurnost pomeni varnost" je tiščanje glave v pesek. Če kje, potem je prav pri kriptografskih metodah pomemben neodvisen zunanji strokovni pregled uporabljenega postopka, če naj se ne slepimo z utvaro o varnosti.

Posredna tržna vrednost odprtega programja

Raymond navaja nekaj zgledov, kako lahko s programi z dostopno izvorno kodo ustvarjamo posredno tržno vrednost.

- *Ustvarjanje ali držanje tržne niše.* Če nastopamo v dveh soodvisnih tržnih segmentih, lahko zapustimo manj donosnega in se osredotočimo na bolj donosnega. Če na primer prodajamo brskalnike in strežnike, odpremo brskalnike in se osredotočimo na donosnejši del: strežnike, oglaševanje na internetni vstopni točki ali naročnino. Za tak korak se je spomladi 1998 odločila družba Netscape Communications.
- *Gonilniki.* Nujno zlo za vse proizvajalce naprav je pisanje gonilnikov. Trg zahteva, da jih pišejo in vzdržujejo, kljub temu pa to ni vir dohodka, saj živijo od prodaje strojne opreme. Z odprtjem izvorne kode gonilnikov lahko proizvajalec pridobi širšo bazo sodelavcev, kar pomeni hitrejše prilagajanje, zaradi neodvisnega zunanjega pregleda pa tudi večjo zanesljivost. Zadržek, da s tem izdamo pomembne informacije o zgradbi svoje strojne opreme, je s skrajšanjem razvojnih ciklov strojne opreme tudi odpadel. Konkurenca lahko iz izvorne kode morda res razbere zgradbo opreme, vendar pa gre čas, ko so se ukvarjali z analizo našega izdelka, na račun razvoja njihovega. Plagiatorstvo je past. Glede na kratko življenjsko dobo strojne opreme je model dostopne izvorne kode zanjo še posebej primeren. Ko izdelovalec napravo opusti, so kupci prepuščeni sami sebi. Če so gonilniki dostopni, je nekaj več možnosti za odpravo napak tudi potem, ko proizvajalec izdelek preneha vzdrževati, kar bo morda pritegnilo kakega kupca ... Tako odločitev je marca 1999 sprejela družba Apple Computer.
- *Objavimo recept, odpremo restavracijo.* V tem primeru odprto programje drži odprto nišo za storitve. Tako obratujejo hiše, kot so Red Hat, Caldera ali SuSE, ki garnirajo odprto programje v enostavno uporabne pakete.
- *Potrebščine.* Ena spremljevalnih dobičkonosnih dejavnosti ob odprtem programju je tudi prodaja potrebščin: od majic, vrčkov za kavo in kap do profesionalno urejene dokumentacije. Primer slednje je založba O'Reilly.
- *Sprostimo prihodnost, prodajamo sedanjost.* Program izdamo skupaj z izvorno kodo pod zaprto licenco, vendar časovno omejeno, denimo takšno, ki dopušča le nepridobitno izrabo, a obenem zagotavlja, da program zapade pod GPL eno leto po izidu, ali pa, če ga proizvajalec opusti. V tem primeru so stranke lahko prepričane, da je program karseda prilagodljiv njihovim potrebam, saj imajo izvorno kodo. Izdelek nosi tudi določena zagotovila glede

prihodnosti. Slednje lahko vliva strankam več zaupanja v izdelek, kot bi ga, če bi bil ta izdan pod zaprto licenco, odprtost izvorne kode pa seveda prinese tudi neodvisni zunanji pregled in s tem večjo zanesljivost. Tak pristop uporablja Aladdin Enterprises pri programu Ghostscript.

- *Sprostimo programje, tržimo blagovno znamko.* Model je zaenkrat špekulativen. V njem odpremo program, vendar zadržimo testni nabor in uporabnikom na njihovo željo prodamo pravico do uporabe blagovne znamke, ki zagotavlja, da je njihov izdelek združljiv z vsemi, ki uporabljajo isto blagovno znamko. Na ta način bi Sun Microsystems lahko tržil Javo in Jini.
- *Sprostimo programje, tržimo vsebino.* Nepreverjeni model, primeren za večje ponudnike internetskih storitev, kot je na primer AOL, ki je razvil tudi lastni spletni brskalnik. Pri njem vrednost ni v kodi brskalnikov ali strežnikov, temveč v vsebini. Kodo brskalnika bi lahko odprli in prodajali izključno naročnino na vsebino. Z neodvisnim prenosom brskalnika v novo strojno okolje seveda samo pridobimo, ker se razširi krog potencialnih strank.

Kaj pri programju sploh prodajamo?

V modelu zaprtega programja je stvar jasna. Zavestno se odrečemo neodvisnemu zunanjemu vpogledu v kodo in pobiramo dividendo na algoritme in ideje, skrite v programu. Do pred nekaj leti je prevladovalo mnenje, da je to tudi edini način dobička v industriji programske opreme. Vzpon odprtega programja nas sili v to, da začnemo razmišljati tudi o bolj subtilnem in posrednem donosu odprtega programja.

Model odprtega programja prek postopka neodvisnega zunanjega pregleda izvorne kode zagotavlja visoko raven kakovosti in zanesljivosti ter se dobro prilagaja velikosti. Če je izpad dohodka iz trženja programskih skrivnosti mogoče nadomestiti s prihodkom od storitev, dodatkov in postranskih trgov, vezanih na programje, je verjetno prehod na model odprtega programja pametna odločitev. Odprto programje je tudi eden od načinov za podporo neodvisnim odprtim standardom, kot je na primer TCP/IP. Hitro rast interneta je brez dvoma omogočilo tudi dejstvo, da je odprt, ne pa lastniški standard. Vsekakor pa je dogajanje, ki je sledilo odprtju brskalnika Netscape spomladi 1998, ovrglo mnenje, da od odprtega programja ni moč pričakovati nobenega donosa.

V prid modelu odprtega programja govorijo naslednji dejavniki:

- Zanesljivost/stabilnost/skalabilnost so kritični parametri.
- Pravilnosti zasnove in izvedbe ni mogoče enostavno verificirati drugače kot z neodvisno recenzijo. Vsak količkaj zahteven program zadošča temu kriteriju.
- Programje je bistveno za nadzor nad uporabnikovim poslom. Ali drugače, uporabnik si ne more privoščiti popolne odvisnosti od proizvajalca programske opreme.
- Programje vzpostavlja ali omogoča enotno računsko ali komunikacijsko infrastrukturo.
- Ključni postopki ali njihovi funkcionalni ekvivalenti so del splošnega inženirskega znanja.

Linux ali Apache sta vzorčna primera, ki zadoščata vsem petim kriterijem. Poučen je tudi razvoj rabe omrežnih protokolov. Po nekaj neuspešnih poskusih izgradnje imperija s protokoli DECNET, XNS, IPX in podobnimi je sredi devetdesetih prevladal najstarejši med njimi – TCP/IP – ki edini temelji na odprti kodi.

Kot nasprotje temu navaja Raymond primer družbe, ki trži program za lesne obrate – optimizacija razreza hlodov. Ta primer približno zadošča le tretjemu kriteriju, zato ni verjetno, da bi z odprtjem bistveno pridobil.

Če naj verjamemo današnjim trendom, kaže, da bo eden večjih izzivov projektnega vodenja v naslednjih letih in desetletjih primerna izbira trenutka, ko se odločimo za odprtje kode. Če to storimo prezgodaj, zavržemo del prihodka, ki bi ga imeli s trženjem programskih skrivnosti. Če to storimo prepozno, je lahko prav tako slabo. Če nas pri tem v naši tržni niši z odprtjem kode prehiti drug proizvajalec, bo na svojo stran pritegnil najboljše razvijalce in z njimi tudi večino prednosti, ki jih prinaša model odprtega programja.

Nevarnosti, ki grozijo odprtemu programju

Oglejmo si še nekaj potencialnih nevarnosti, ki grozijo modelu odprtega programja. Večino smo ovrgli že sproti, kot denimo strah pred fragmentacijo Linuxa. Dve nevarnosti pa vendarle obstajata. V svojem poročilu jih je navedel Valloppillil (Valloppillil 1998) kot možnosti, s katerimi si lahko monopolisti zagotovijo svoj položaj še naprej.

- *Prisvajanje protokolov.* Linux in internet sta bila tako uspešna, ker temeljita na javno dostopnih protokolih (TCP/IP, SMTP, HTTP, POP3, IMAP, NFS

idr.). Tu so interesi monopolistov in skupnosti, ki podpira odprto programje (pa tudi vseh manjših proizvajalcev programske opreme, ki niso v monopolnem položaju), v konfliktu. V interesu monopolista je zamenjava javnih protokolov z lastniškimi, nad katerimi bdi sam namesto IETF.

- *Patenti nad programi.* Evropska unija za razliko od ZDA ne priznava patentov nad programi. Ti so celo v ZDA predmet žolčnih polemik, saj varujejo izključno največje proizvajalce, ki si lahko privoščijo vzajemno licenciranje množice patentov. Celo nekateri večji ameriški proizvajalci, kot sta Adobe in Oracle, so izrazili mnenje, da patenti na programsko opremo inovativnosti bolj škodijo kot koristijo (Freepatents.org). Kljub temu skupine, kot je denimo BSE, izvajajo na Evropsko patentno organizacijo (EPO) velik pritisk, da se evropska zakonodaja uskladi z ameriško in spremeni člen 52.2 Evropske patentne konvencije (EPC), ki računalniške programe izvzema iz skupine izdelkov, ki jih je moč patentirati. Oktobra 2001 je tako EPO že delno popustila in sprejela določilo, s katerim je dovolila odobritev patentov, ki izrecno navajajo programsko kodo ali podatkovne strukture.

Življenje po revoluciji

Ideja odprtega programja ima pridih revolucionarnega in uporniškega. Ali pa lahko ideja preživi tudi na daljši rok? Ni razlogov, da ne. Obenem pa seveda tudi ne smemo pričakovati odprtja čisto vsega programja.

Programe lahko razvrstimo v nekaj skupin. V prvi je obče programje, ki vzpostavlja osnovno tehnično infrastrukturo (operacijski sistemi, omrežne komunikacije) in uporablja javno dostopne protokole. Pričakujemo lahko, da bo ta skupina programja ostala in še v večji meri postala odprto programje, katere razvoj bodo vodili konzorciji uporabnikov in/ali založniško/storitvena podjetja, kot je na primer Red Hat.

Antipod tej skupini so namenski programi, npr. urejevalniki besedil, ki delujejo v svetu neobstoječih ali šibkih tehničnih standardov. V tej skupini je največ programov, v kateri dodatna zanesljivost, pridobljena z neodvisnim zunanjim vpogledom v kodo, morda ne bo odtehtala vrednosti tajnih algoritmov ali tehnik.

Značilen predstavnik vmesne skupine so programi za obdelavo podatkovnih zbirk potem, ko je uporaba SQL odlepila vmesnike od strežniškega dela. Odprtje tega segmenta je odvisno od cene nezanesljivosti. Višja ko je, večji bo pritisk trga po odprtju.

Prehod na odprto programje seveda ne pomeni konca industrije programske opreme, kot jo poznamo danes, pomeni pa njeno prestrukturiranje in zasedbo novih niš, ki jih prinaša sprememba. S stališča uporabnika programja pa je seveda tak razvoj ugoden, saj s tem, ko bo vrsta programov živela naprej, namesto da bi bila opuščena, omogoča večjo izbiro.

Literatura

Axelrod, R. (1985): The Evolution of Cooperation, Basic Books, New York.

Brooks, F. P. Jr. (1975): The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley, Reading.

Free Software Foundation (1991): GNU General Public License, 2. izdaja. http://www.gnu.org/copyleft/gpl.html.

Freepatents.org – Protecting Competition Against the Abuse of Software Patents, http://www.freepatents.org/.

Himanen, P. (2001): The Hacker Ethics and the Spirit of the Information Age, Secker & Warburg, London.

Levy, S. (2001): Hackers: Heroes of the Computer Revolution, 2. izdaja, Penguin USA, New York.

Open Source Initiative (1998): The Open Source Page, http://www.opensource.org/.

Raymond, E. S., ur. (1996): The New Hacker's Dictionary, 3. izdaja, MIT Press, Cambridge.

Raymond, E. S. (1997): The Cathedral and the Bazaar, http://www.tuxedo.org/~esr/writings/cathedral-bazaar/.

Raymond, E. S. (1998): Homesteading the Noosphere, http://www.tuxedo.org/~esr/writings/homesteading/.

Raymond, E. S. (1999): The Magic Cauldron, http://www.tuxedo.org/~esr/writings/magic-cauldron.

Raymond, E. S. (2001): The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, 2. izdaja, O'Reilly & Associates, Sebastopol.

Valloppillil, V. (1998): Open Source Software – A (New?) Development Methodology", http://www.opensource.org/halloween/halloween1.html.

Williams, S. (2002): Free as in Freedom: Richard Stallman's Crusade for Free Software, O'Reilly & Associates, Sebastopol. http://www.oreilly.com/openbook/freedom/.

Primož Peterlin je med uporabniki odprtega programja znan predvsem kot avtor navodil za prilagoditev operacijskega sistema Linux slovenskemu jezikovnemu okolju. Sam ali s sodelavci je v domačih in tujih revijah objavil več člankov o Linuxu, skupaj s sodelavci pa tudi dva priročnika za delo z njim. Po izobrazbi je doktor fizike in se ukvarja z biofizikalnimi raziskavami mehanskih lastnosti celičnih membran.