



# Befria din kod!

***Om att släppa en produkt som fri källkod***

Markus Härnvi

## Inledning

Har du och ditt företag funderat på att släppa en programvara ni utvecklat under en fri licens? Den här artikeln försöker förklara vad man behöver tänka på för att lyckas skapa en *community* runt ett mjukvaruprojekt. Hur förklarar man för företagsledningen vad öppen källkod är? Hur hanterar man licensdjungeln? Hur får man människor att bidra till projektet? Vad finns det för risker?

## Vad är öppen källkod?

*Open Source*, *öppen källkod*, *fri mjukvara*. Kärt barn har många namn. Termen ”Open Source” lanserades på bred front av *Eric S. Raymond*. Den term som fram till dess mest använts om fenomenet var *Free Software*. Raymond menade att det var för många som misstog sig på ordet ”Free”. Det fanns en förutfattad mening att det inte gick att skapa affärer på basen av fri mjukvara. Raymond ville visa företagsvärlden att öppen källkod inte var en fiende till företagande och vinst.

Termen ”Free Software” förknippades starkt med organisationen *Free Software Foundation* (FSF) och dess grundare *Richard M. Stallman*. Han startade organisationen 1985 med målet att all mjukvara skulle vara fri. Med *fri* menade han inte nödvändigtvis *gratis* utan fri som i ”frihet”. Stallman talar om de *fyra friheterna*:

- Frihet att köra programmet för alla syften
- Frihet studera hur programmet fungerar och ändra det efter behov
- Frihet att distribuera kopior av programmet till hjälp för andra
- Frihet att distribuera ändringar så att alla får del av dem

Mot slutet av 90-talet ville Eric S. Raymond och andra visa att fri mjukvara inte behöver grundas i Stallmans filosofiska utgångspunkter om att all källkod skall vara fri. Enligt Raymond går det inte att få företagsvärlden att anamma konceptet utifrån filosofiska argument. Raymond såg att Internets framväxt till stor del baserades på fri mjukvara. Hela infrastrukturen bakom DNS, epost, WWW baserades på fria programvaror som *bind*, *sendmail* och *Apache*. Ett annat exempel var Linux som växte lavinartat, även i serverhallarna hos de stora företagen. Raymond menade att det finns rent företagsekonomiska fördelar med fri mjukvara. Fri mjukvara innebär ett annat sätt att utveckla lösningar och det kan bli både billigare och snabbare än traditionella, slutna modeller. Kort sagt: man kan tjäna pengar på fri mjukvara. Även om Richard Stallman och FSF inte var emot möjligheten att tjäna pengar på mjukvara, fanns det en aura av idealitet och filosofi som skrämde bort många mjukvaruföretag. För Raymond var det nyttoaspekterna som styrde och han såg inga problem med att blanda fri mjukvara och proprietära (stängda, hemliga) lösningar.

Raymond startade *Open Source Initiative* (OSI) 1998 och började formulera en mer företagsvänlig, pragmatisk definition av vad fri mjukvara är. Han var en nyckelspelare när Netscape belutade sig för att släppa källkoden till sin webbläsare fri 1998. Mozillaprojektet startades och har nu, långt senare, fått en rejäl framgång med webbläsaren *Firefox* och epostprogrammet *Thunderbird*.

Så här beskriver Raymond grunden för Open Source i boken *The Cathedral and the Bazaar*:

*"The basic idea behind open source is very simple: When programmers can read, redistribute, and modify the source code for a piece of software, the software evolves. People improve it, people adapt it, people fix bugs. And this can happen at a speed that, if one is used to the slow pace of conventional software development, seems astonishing."*

## Definition

Open Source Initiative har skrivit en definition av vad som är öppen källkod. Det räcker inte bara att skicka med källkoden till sin produkt för att den skall kallas "Open Source" enligt OSI:s definition.

Här är en juridisk lekmans sammanfattning av de viktigaste punkterna som skall finnas i villkoren för att en mjukvarulicens skall kunna kallas "Open Source":

1. Fri vidaredistribution.  
Man får inte hindra en köpare att vidare distribuera (med eller utan monetär ersättning) mjukvaran/källkoden till tredje part.
2. Tillgänglig källkod  
Antingen skall källkoden skickas med produkten (på CD/DVD), eller finnas tillgänglig för nedladdning t.ex. på en webbplats.
3. Tillåta härledda verk  
Man får inte hindra att andra bygger vidare på källkoden och distribuerar dessa ändringar.
4. Ingen diskriminering  
Man får inte med licens hindra användning i vissa brancher, delar av världen eller grupper av människor.

Inom öppen källkods rörelsen finns, som jag redan antytt, många olika målsättningar och ideologier. Det har lett till en hel flora av olika licenser som ställer olika krav på den som distribuerar fria program. Men all programvara som kallas Open Source lyder under de ovanstående fyra principerna. Annars är det inte fri källkod.

## Varför släpper ett företag sin produkt fri?

Varför vill ett företag ge bort det som de kanske lagt hundratusentals eller miljoner kronor på att utveckla? Vem är så korkad? "Har jag närt en kommunist vid min barm?"

Vad finns det att tjäna på öppen källkod?

## Marknadsföring

### Gratis PR

Genom att släppa sin produkt fri kan det skapas positiva vågor. Många utvecklare och användare kommer att ladda ner produkten, testa den, visa den för andra. Kanske kommer artiklar att skrivas om den i tidningar och på webbplatser. En artikel om en stängd programvara kan oftast upplevas som säljsnack och negligeras, medan en artikel om en fri programvara lockar till läsning. Redan samma dag kan utvecklaren ha läst om produkten, laddat ner och testat den. Insatsen att lära sig om produkten kommer att kunna betala sig. Den kommer att

Copyright 2007 Markus Härnvi. Artikeln får användas enligt *Creative Commons Erkännande-Dela Lika 2.5 Sverige Licens*, <http://creativecommons.org/licenses/by-sa/2.5/se/>

finnas tillgänglig i utvecklarens verktygslåda, utan att hon först måste betala licenskostnader för varje projekt hon får nytta av den i.

Man skall inte underskatta styrkan i utvecklarens inflytande vid val av produkter inom ett företag. Om "hackern" gillar produkten kommer han/hon att rekommendera den för IT-chefen.

## Trygghet för kunden

Om det finns en levande community kring produkten kommer den att leva vidare även om ett företag går i konkurs. Någon annan kan ta upp projektet och få det att leva vidare. I värsta fall har man källkoden och kan hyra in konsult hjälp för att rätta buggar och vidareutveckla. Detta kan ge en trygghet för kunden som kan verka positivt vid införsäljning.

## Minskade utvecklingskostnader

Om man lyckas med sin Open Source-lansering kommer man att ha möjlighet att få hjälp från fler utvecklare än man betalar lön till varje månad. Detta bygger på att man har en bra produkt som många vill använda, samt att man lyckas skapa en community. Företag som lyckats med detta har mångdubblat sin utvecklargrupp. Från att ha varit ett litet företag i ett land, har projektet plötsligt utvecklare i flera tidszoner.

Det kan gå till så att en utomstående utvecklare anpassar koden för sina syften och skickar in sina ändringar så att de kan integreras med huvudprodukten. En annan utvecklare rättar en bugg. Någon bidrar med en artikel eller en installationsguide på tyska.

Genom feedback som ges på forum och mailinglistor får företaget hjälp att förfina produkten, göra den mer generell så att den kan appliceras i nya sammanhang. Kanske kan produkten komma att användas i helt nya branscher än vad den utvecklats för.

Men kommer verkligen utvecklare att bidra med sina utökningar och buggfixar? Kommer de inte att tjuvhålla på dem? Nej, de vill ha in sina utökningar och rättningar i "basen", i den gemensamma källkoden. Annars måste de själva applicera dem för varje ny version av "basen" som släpps. De sparar tid och pengar på att det integreras i baspaketet.

## Varför hjälper de oss?

Hur kan det komma sig att utvecklare lägger sin sparsamma fritid på att skriva kod för olika fria projekt? Orsakerna är många. För många utvecklare handlar programmering om hantverk och skapande. Man är stolt över att ens kod tas emot och integreras. Det finns ett tävlingsmoment där man vill erbjuda den bästa lösningen på ett problem. Ett annat skäl är att man får mycket värdefulla kontakter som kan hjälpa i yrkeslivet. *Linus Torvalds* behöver aldrig mer aktivt söka jobb, för att ta det mest extrema exemplet. En del är fristående konsulter som försörjer sig på att konsultera kring öppen källkodsprogram. De har ett egenintresse i att projekten lyckas - de bygger sitt eget företag på det.

## Gratis infrastruktur och hjälpmedel

Det finns en del företag som erbjuder mycket kvalificerade hjälpmedel för programutveckling och tar riktigt bra betalt för dem. En del av dessa företag låter Open Source-projekt använda produkterna gratis, t.ex bughanteringsverktyget *Jira*, wikin *Confluence* och kodgranskningsverktyget *Fisheye*.

Man kan också placera sitt projekt hos t.ex. Sourceforge.net eller java.net. De erbjuder bandbredd (ett populärt projekts nya version kan sänka vilket normalt webbhotell som helst), versionshantering, epostlistor, filserverutrymme, forum, bugghantering. Allt administreras av dem och projekt med öppen källkod får använda tjänsterna gratis. Att placera sitt projekt på en av dessa siter hjälper också till med marknadsföringen.

## **Hur tjänar man pengar på öppen källkod?**

Open Source betyder inte alltid gratis. "The word 'free' in 'Free Software' means free as in 'free speech', not 'free beer'". Alla fria licenser jag känner till låter dig sälja din produkt för det pris du vill. Däremot föreskriver de att du måste ge vissa rättigheter till din källkod vidare till den du säljer programvaran till.

Det finns många olika sätt att göra pengar på/med öppen källkod:

- Paketera snyggt och användarvänligt  
*Redhat* är ett exempel. All programvara de säljer har öppen källkod. Du kan ladda hem nästan allt i *Redhat Enterprise Linux* och kompilera det själv. Eller också kan du betala till *Redhat* och dessutom få support, garantier och snabba uppdateringar.
- Vässa produkten  
Genom ökad spridning och genom kod från externa utvecklare blir produkten allt bättre och når nya marknader. Här finns pengar att tjäna för de som associeras mest med produkten.
- Konsulta runt produkterna  
Det finns oräkneliga exempel konsultbolag som till stor del använder fria produkter i sina konsultuppdrag. *IBM* t.ex. använder mycket Open Source.
- Ta öppen källkodsprodukter och addera egna "stängda" lösningar ovanpå  
*Apple* gör så med *MacOS X*, *Nokias* brandväggar hade i alla fall tidigare *BSD Unix* i botten. *IBM*, *SUN*, *Bea* m.fl. använder öppen källkod i sina applikationsservrar, men lägger in egen stängd kod ovanpå.
- Utbilda, skriva böcker  
*O'Reilly* och andra förlag. Alla IT-utbildningsföretag har kurser som berör öppen källkod.
- Skapa webbplatser om och för öppen källkodsprojekt och få reklamintäkter  
T.ex. *Slashdot*, *Freshmeat*
- "Dual-license"  
Distribuera produkten under flera licenser samtidigt. Mer om det senare.

## Några olika licensmodeller

Det finns för många licenser i omlopp. Men här beskriver jag de som jag tycker är mest användbara och speglar de flesta huvudgrupperna.

### Apache License

Den mest liberala av de jag presenterar. Du kan göra vad du vill med källkoden och behöver inte skicka med din egen källkod om du använder komponenter som är licensierade med Apache-licensen. IBM kan t.ex. ta *Tomcat* (som är släppt under denna licens) och baka in den i sin *WebSphere Application Server* utan att behöva ge andra källkoden till vare sig Tomcat eller WebSphere. De måste bara nämna i sin dokumentation att produkten innehåller källkod från Apache-projektet. De är inte skyldiga att bidra tillbaka med källkod (förbättringar, buggfixar) till Tomcat om de inte vill. En annan vanlig licens i samma familj är *BSD*.

### GPL

*GNU Public License* är FSF:s licens och präglas starkt av Richard Stallmans idealistiska tänkande. Om du använder GPL-licensierad kod i din produkt måste du släppa källkoden till hela din produkt fritt. GPL ”smittar” din kod och gör den automatiskt fri. Om någon modifierar Linuxkärnan är han skyldig att publicera dessa ändringar så fort han vill distribuera den till en annan part. Observera att du kan använda GPL-programvara ”in-house” utan att behöva dela med dig av din källkod. Om du t.ex. använder GPL-programvara på din webbplats räknas det inte som distribution. Men om du säljer vidare din webbpubliceringslösning till andra så klickar GPL-villkoren in och du måste skicka med källkoden.

Idag används GPL version 2, men inom kort kommer version 3 att släppas. I stort gäller samma villkor, men man har infört en hel del förtydliganden.

Sun släppte nyligen Java under GPL. För Sun är detta ett sätt att behålla kontrollen. Ingen kan addera ”hemlig” kod till sin variant av Java. Utvecklingen måste ske i det öppna och Sun är garanterad att kunna ta tillvara alla förbättringar som görs.

### LGPL

*Library GNU Public License* är en mildare form av GPL. Används för komponenter. En komponent med LGPL-licens kan du ”väva” in i din produkt utan att ”smittas” av LGPL. Om du däremot gör ändringar i själva komponenten måste  *dessa* förändringar göras tillgängliga för andra.

### CDDL, MPL

*Common Development and Distribution License* och *Mozilla Public License*. Med dessa licenser är källkoden fri och ändringar måste distribueras. Men du får lägga till egna proprietära delar, förpacka och sälja under en annan licens. Netscape distribuerar *Mozilla* med egna utökningar och under sitt eget namn. Sun säljer sitt operativsystem *Solaris* med egna licenvillkor, men du kan också använda *OpenSolaris* under CDDL.

Skillnaden mellan dem är att MPL är knuten till Mozillaprojektet. I licensen hänvisas till Mozilla överallt. Sun ville ha en licens i samma stil och skrev då om MPL så att den blev generell. Alla kan direkt använda CDDL utan att behöva skriva om den.

## ”Dual licensing” - släppa en produkt under flera licenser

Det viktigaste exemplet på detta är *MySQL*. MySQL släpper sin databas under GPL-licensen. Du får ladda ner, använda, ändra och vidare distribuera MySQL, men är bunden av GPL:s villkor: alla förändringar och all källkod i den produkt som använder MySQL måste också släppas fri. Även databasdrivrutiner för C och Java är under GPL. Förenklat betyder det att om du använder MySQL får du inte sälja din produkt utan att skicka med källkoden till *din* produkt under GPL. MySQL ”smittar” din produkt med GPL:s regler.

Vill du sälja din produkt och inte öppna den för världen, måste du kontakta MySQL AB och köpa MySQL under en *annan* licens som medger användning utan GPL:s krav.

MySQL AB släpper alltså sin databas under flera licenser: en fri, GPL och en egen stängd. På detta sätt kan de få ”cred” för att vara öppen källkod. Deras databas finns med som standard i alla Linuxdistributioner. Där igenom har de en enorm spridning. Pengar flyter in genom att mjukvaruleverantörer bygger lösningar som integrerar mot MySQL. De måste köpa licenser för att få distribuera MySQL med sina lösningar.

En nyckel till detta är att MySQL från början hade total kontroll över vem som skrivit källkoden till produkten. De kan alltså själva bestämma vilken licens som skall gälla för koden. När det gäller Linuxkärnan så är varje deltagande programmerare rättighetsinnehavare för sin lilla del. Det måste röra sig om många hundra personer. Linux Torvalds skulle alltså inte kunna ändra Linux licens till, säg, Apachelicensen, utan att få godkännande från alla dessa hundratals människor.

Men då kan ju inte MySQL få bidrag från andra utvecklare utanför sin organisation? Jo, de kan de. Dels kan de utan vidare ta emot ”patchar” som är för små för att uppnå verkshöjd. Några källkodsradar kan räcka för att fixa en bugg. Om någon bidrar med större kodmängder, måste de först skriva på ett papper där de delar sin upphovsrätt till materialet med MySQL AB.

### Nackdelar

En stor nackdel med dual-licensing är att man skapar en osäkerhet. Är företaget verkligen intresserade av att bygga en community? Är det inte bara ett förtäckt, stängt projekt som alla andra?

För att denna affärsmodell skall lyckas, måste den fri licens man använder vara av det ”smittande” slaget, vanligen GPL.

### Avtal om delad upphovsrätt

Om man vill använda någon av de strängare licenserna (GPL, LGPL, MPL, CDDL) behöver man fundera om de som bidrar skall skriva på ett avtal i stil med det som MySQL använder. Annars kan det bli i det närmaste omöjligt att byta licens senare (man måste kontakta alla utvecklare som någonsin bidragit med kod för att få deras godkännande). Med ett avtal om delad upphovsrätt behåller man friheten för framtiden.

## **Hur gör man?**

### **Vill nån ha det?**

För att lyckas med öppen källkod måste man planera noga. Den första frågan man måste ställa sig är: Vill någon använda det vi har?

- Finns det redan etablerade fria projekt som erbjuder samma funktionalitet? Kommer produkten att kunna tävla om ”mind share” med de andra i den fria mjukvaruvärlden?
- Är produkten tillräckligt generell så att andra kan ha nytta av den? Finns det en marknad för andra företag att vidareutveckla och tjäna pengar på produkten?
- Är koden i bra skick? Ingen vill ladda ner ett projekt och se spagettikod. Om bara företagets anställda ser källkoden är de bara de som mår illa, men nu skall den exponeras för världen. Krävs det en arbetsinsats att städa upp först?
- Är det lätt att kompilera och installera? Utvecklare är rastlösa. Går det inte snabbt att få igång produkten, utan att man behöver installera andra komponenter och sätta miljövariabler i en evighet, kommer ingen att orka testa.

### **Upphovsrätt**

Man måste också ha koll på upphovsrätten till det man släpper:

- Är projektet beroende av komponenter som inte kan släppas fria?
- Är projektet beroende av fria komponenter, men som inte är kompatibla med den licens som planeras att användas. Alla licenser är tyvärr inte kompatibla med varandra. Tumregeln är att liberalare licenser kan inkluderas i striktare projekt, men inte tvärtom.
- Har vi upphovsrätten till alla ”våra” delar? Äger vi all källkod i projektet?

Om dessa initiala frågetecken är utträtade så finns möjligheten att släppa projektet fritt. Nu måste det till en plan för hur man skall skapa en community runt produkten. Utan community – ingen vinst. Att bara släppa en produkt fri ger måhända en tillfredsställelse för företagets ”hackers”, men det ger inget mervärde till produkten. För att kunna dra nytta av Open Source behövs volym. Många behöver ladda ner produkten, använda den, rapportera och rätta buggar, implementera nya funktioner.

## Infrastruktur och engagemang

Det kommer att krävas en insats av företaget. Det behövs infrastruktur och personligt engagemang;

- Publik webbplats  
En attraktiv och överblickbar webbplats är viktigt. Det handlar om marknadsföring.
- Versionshantering  
Källkoden måste versionhanteras. Utvecklare måste kunna bidra med kod från olika delar av världen. Medlemmarna i projektet behöver kunna spåra ändringar enkelt via webbverktyg. Man måste anonymt kunna ladda ner de senaste uppdateringarna.
- Bughantering  
Ett publikt bughanteringssystem behöver finnas. Medlemmar och användare skall kunna mata in buggrapporter och förändringsförslag och kunna följa sina ärenden. Man skall kunna fördela uppgifter till olika utvecklare i projektet.
- Diskussionforum, chatt och mailinglistor  
Kommunikation är oerhört viktigt i ett öppen källkod-projekt. Utvecklarna är spridda över stora ytor och kanske flera tidszoner. Anställda i vårt företag behöver avsätta tid att kommunicera och besvara frågor.
- Wiki eller annan ”knowledge base”  
För att det skall vara lätt för användare och utvecklare att bidra med dokumentation behövs ett enkelt system för det.

## ”People skills”

För att bygga en community krävs det ledare som har god hand med olika sorters människor. Det kommer att bli konflikter mellan utvecklare, externa och interna. Det finns en stor risk att de utanför företaget börjar tänka i ”vi och dem”. Det gäller att behålla öppenheten, att jobba för konsensus och att reda ut konflikter öppet och ärligt. Även om någon ibland behöver sätta ner foten bör diskussionen ske öppet.

Alla är överrens om att Linux aldrig hade blivit något stort om det inte var för Linus Torvalds förmåga att med lagom strama tyglar och sunt förnuft hållit samman projektet. Som en jämförelse kan man ta Theo de Raadt som leder OpenBSD. Han har ett rykte om sig att vara extremt hetlevrad och hamnar ofta i konflikter. OpenBSD är i jämförelse med Linux ett litet projekt (fast Theo har förmodligen ingenting emot att inte vara massornas man).

## Marknadsföring

Det finns tiotusentals projekt med öppen källkod. De flesta är små hobbyprojekt som aldrig blir något stort. Ofta därför att det är en smal produkt, eller en dålig implementation. Men en sak som utmärker projekt som lyckas är att de har en tydlig marknadsföring. De skär genom bruset och skapar ”buzz”.

- Webbplatsen skall vara effektivt indexerad på Google
- När man går ut officiellt skall infrastrukturen ovan vara så gott som klar  
Det skall finnas en FAQ, wikin skall innehålla bra info, forumet skall innehålla diskussionstrådar (använd det internt innan lansering).

- Överväg att köpa annonsutrymme på webbplatser, men designa annonserna så att det inbjuder till medverkan – inte till köp.
- Skriv några tekniska artiklar som beskriver den smarta idén och försök få dem publicerade på bra webbsiter som behandlar produktens tekniska sfär.
- Berätta om det på relevanta mailinglistor och forum  
Men gör det smart - ingen spam! Att bli associerat med spam sänker ett projekt direkt.
- Bli listade på freshmeat.net och andra listor över fri mjukvara
- Använder projektet komponenter som har egna projektwebbplatser? Se om de har en ”powered by”-sida där man kan få lägga en länk till ert projekt.

## **Risker**

Det finns förstås många risker med att släppa ifrån sig sin investering under en fri licens. Det troligaste riskscenariet är att man har spenderat resurser och släppt sin produkt fri utan att få någonting tillbaka. Men även om man lyckas med lansering och byggande av community finns det risker.

## **Konkurrens**

Konkurrenter kan ladda ner källkoden och använda den i egna lösningar, eller bara få kunskap om hur nya funktioner implementerats.

Det finns en risk att vi förlorar affärer. Men hur stor är marknaden egentligen? Finns det rum för flera? Om vår produkt blir den allmänt accepterade lösningen, är det inte då vi som har mest att vinna på det, även om andra också drar nytta av vår kod? Vem går man till om man vill ha originalet?

Använder konkurrenter koden utanför licensvillkoren (ponera att produkten släppts under en sträng licens, men konkurrenten inte delar med sig av källkoden) kan man gå till domstol. Det finns en del prejudikat, men framför allt finns det flera exempel på att den uppståndelse brottet lett till i media har fått företagen att lyda villkoren. Underskatta inte kraften hos några tusen arga hackers. Free Software Foundation har advokater som driver rättsfall där företag stulit fri programvara.

Ibland kan det fungera att släppa en del av produkten fri, men behålla vissa delar proprietärt. Det kan vara en farlig väg då det ger ett intryck av att man inte menar allvar. I så fall skall man vara ärlig och tydligt skilja mellan det fria projektet och på det proprietära produkten. De bör ha olika namn. Så gör t.ex. svenska företaget *Primekey* med sin certifikatserver. *EJBCA* är ett fritt projekt, medan *PrimeCA* är den förpackade, utökade produkten.

## **Utvecklare lämnar företaget och startar eget**

Ledande utvecklare kan ta källkoden och starta ett eget konkurrerande företag. Händer detta har dock företaget förmodligen andra problem. Hur tar man hand om sina begåvningar? Risken att förlora nyckelpersoner lever man med ständigt. Kanske kan öppen källkod göra det extra intressant att stanna kvar inom företaget än det annars skulle vara.

Värdet i företaget är större än källkoden. Även om de upproriska anställda skulle kunna få med sig stora delar av källkoden, har de inte avtal med kunder, säljkontakter, presentationsmaterial. Ge utvecklarna bra betalt, optioner i företaget och låt dem hålla på med intressanta arbetsuppgifter så är denna risk minimerad.

## **Externa utvecklare klantar till det**

Vill man ha effektiv hjälp från externa utvecklare, måste man förr eller senare ge dem tillgång att själva skicka in sina ändringar i versionshanteringen och göra andra mer operativa delar i projektet. Då har man människor som man inte har något anställningsavtal med väldigt nära kärnverksamheten. Det kan hända att de gör stora misstag med projektets resurser. Misstag som drabbar befintliga kunder eller stjälar resurser när de skall åtgärdas.

Man kan minska den här risken genom att inte vara för frikostig med incheckningsrättigheter. Låt utvecklare visa vad de går för först. De kan skicka in ”patchar” som en anställd granskar och applicerar på koden i början. Automatiska tester av källkoden och automatisk kontroll att kodningsriktlinjer följs kan också minska risken för misstag. Man kan ordna träffar för deltagare i projektet där man kan bygga förtroende öga mot öga med nya ansikten.

## **Sammanfattning**

Öppen källkod är passar inte för alla projekt och produkter. Alla företag klarar inte av processen att gå från en liten, men välkontrollerad, utveckling, till det kreativa vimmel som ett fritt projekt kan skapa. Alla har inte uthålligheten att skapa en community och hålla den gående. De legala, infrastrukturella, ekonomiska och tekniska problemen kan verka för stora.

Men för de som vill och kan finns det oerhörda möjligheter, både ekonomiskt och tekniskt, med att öppna sin källkod för världen.

### **Länkar:**

<http://www.opensource.org/>

Open Source Initiative

<http://www.oreilly.com/catalog/producingoss/>

Bok: Producing Open Source Software

[http://opensource.mit.edu/online\\_papers.php](http://opensource.mit.edu/online_papers.php)

Uppsatser och avhandlingar om bl.a. företagsekonomiska perspektiv på öppen källkod

## Innehåll

Befria din kod!.....	1
Om att släppa en produkt som fri källkod.....	1
Inledning.....	2
Vad är öppen källkod?.....	2
Definition.....	3
Varför släpper ett företag sin produkt fri?.....	3
Marknadsföring.....	3
Minskade utvecklingskostnader.....	4
Gratis infrastruktur och hjälpmedel.....	4
Hur tjänar man pengar på öppen källkod?.....	5
Några olika licensmodeller.....	6
Apache License.....	6
GPL.....	6
LGPL.....	6
CDDL, MPL.....	6
”Dual licensing” - släppa en produkt under flera licenser.....	7
Avtal om delad upphovsrätt.....	7
Hur gör man?.....	8
Vill nån ha det?.....	8
Upphovsrätt.....	8
Infrastruktur och engagemang.....	9
”People skills”.....	9
Marknadsföring.....	9
Risker.....	10
Konkurrens.....	10
Utvecklare lämnar företaget och startar eget.....	10
Externa utvecklare klantar till det.....	11
Sammanfattning.....	11