FOSS

Aspekter på fri programvara och öppen källkod i en fri marknadsekonomi

Magisteruppsats, 10 poäng, i informatik

Framlagd: Januari, 2005

Författare: Claes Jönsson

Per Böckman

Handledare: Hans Lundin

FOSS

Aspekter på fri programvara och öppen källkod i en fri marknadsekonomi

© Claes Jönsson

Per Böckman

Magisteruppsats framlagd januari, 2005

Omfång: 130 sidor

Handledare: Hans Lundin

Resumé

Med denna uppsats vill vi ge läsaren en uppfattning om vad Free Software och Open Source (FOSS) handlar om och mer specifikt analyseras huruvida FOSS kan fungera i en kapitalistisk värld eller inte. Den grundläggande idén med Free Software och Open Source, eller fri programvara och öppen källkod, är att ge varje användare av ett datorprogram rätten att fritt köra, undersöka, ändra och vidaredistribuera programmet.

Författarna ställer i uppsatsen frågan ”Hur kan pengar tjänas på FOSS”. Uppsatsen syftar till att försöka bena ut denna frågeställning. Frågeställningen ger upphov till två perspektiv; dels i meningen ”på vilka sätt kan man tjäna pengar på FOSS” och dels i betydelsen ”hur kommer det sig att det går att tjäna pengar på FOSS då det är gratis”. Uppsatsen knyter an till litteratur kring FOSS samt ekonomisk teori och är främst teoretisk till sin natur.

I uppsatsen visas att gamla beprövade affärsmodeller inte kan användas rakt av för företag som använder eller utvecklar produkter baserade på Open Source. Uppsatsen visar också att det finns stora skillnader mellan olika typer av fri programvara och öppen källkod. Dessa skillnader ger upphov till såväl intressanta samarbets- som konkurrensmöjligheter. Slutligen redovisas exempel på vilka möjligheter som finns att generera intäkter för företag som arbetar med ”Open Source”-produkter och de problem och utmaningar dessa företag ställs inför.

Nyckelord: FOSS, Open Source, Free Software, ekonomi, affärsmodeller

Förord

Vi vill passa på att tacka vissa personer som hjälpt oss lite extra med denna uppsats. Vår handledare Hans Lundin har varit mycket behjälplig med kommentarer under arbetets gång. Tack också till Mårten Berggren för tips och intressanta länkar. Slutligen vill vi rikta ett tack våra familjer för hjälp med korrekturläsning och framförallt för ert stöd och er uppmuntran under arbetets gång.

Innehållsförteckning

1 Inledning

"Anyone who says you can have a lot of widely dispersed people hack away on a complicated piece of code and avoid total anarchy has never managed a software project." (Andrew Tanenbaum, i ett inlägg i nyhetsgruppen comp.os.minix den 5:e februari 1992, angående Linus Torvalds nystartade arbete med Linux.)

Med denna uppsats vill vi ge läsaren en uppfattning om vad Free Software och Open Source handlar om. Speciellt försöker vi analysera huruvida Free Software och Open Source kan fungera i en kapitalistisk värld.

I detta inledande avsnitt ämnar vi att teckna bakgrunden till och ytligt presentera de områden som senare kommer att avhandlas i uppsatsen. Vidare kommer vi att beskriva det tillvägagångssätt som vi har valt att använda oss av i själva uppsatsarbetet.

Låt oss redan nu ge en idé om vad Free Software och Open Source handlar om. En tanke är exempelvis att ge varje användare av ett datorprogram rätten att fritt köra, undersöka, ändra och vidaredistribuera programmet. De exempel som nyss omnämndes är hämtade från organisationen Free Software Foundation FSF (FSF, 2004a) som vi kommer att stifta närmare bekantskap med senare i uppsatsen. Gemensamt för Free Software och Open Source är att källkoden till ett program ska vara tillgänglig för användaren så att användaren kan idka de rättigheter som vi nämnt ovan. För Free Software gäller att användaren av ett program ska ge samma rättigheter till nästa mottagare medan detta inte är tvingande för alla typer av Open Source (Olofsson, 2003a). Olofsson noterar vidare att *"Free Software och Open Source bygger i grund och botten på det upphovsrättsliga skyddet för datorprogram"* (Olofsson 2003a, sid. 8). Vi kommer att få anledning att återkomma till detta lite längre fram, men först ska vi avhandla lite formalia.

1.1 Formalia

Innan vi skyndar vidare och angriper ämnet för uppsatsen vill vi passa på att redovisa några formaliteter.

Vi har valt att primärt använda de engelska termerna Free Software samt Open Source för att ge namn åt det objekt vi ämnar studera. Termen Free Software är närbesläktad med och används ibland helt synonymt med Open Source. Vissa författare, som exempelvis Wong och Sayo (2004a) samt Coleman (2004a), har valt att sammanföra orden till akronymet FOSS som ska utläsas som Free/Open Source

Software (eller fri programvara och öppen källkod på svenska). Andra talar om FLOSS som i ”Free Libre Open Source Software” (fri som i frihet öppen källkods mjukvara för att ta en lite krystad svensk översättning). Exempel på den senare benämningen återfinns bland annat i International Institute of Infonomics et al (2002a).

Vi håller oss dock mestadels till FOSS. Sammansatta ord med Open Source som del sätts inom citattecken som till exempel i ordet ”Open Source”-förespråkare. Svenskans ”fri programvara och öppen källkod” är annars en utmärkt översättning och vi använder denna term synonymt med FOSS. Vid de tillfällen det finns skäl att göra en distinktion mellan Free Software och Open Source använder vi de begreppen enskilt istället för sammanslaget i förkortningen FOSS. Vi har även valt att behålla vissa andra engelska termer istället för att översätta dem och hoppas att läsaren finner sig till freds med våra val och prioriteringar avseende detta. Det är inte vår avsikt att konsolidera angliseringen av det svenska språket.

Det kan också vara på sin plats att göra några noteringar kring citat. Citat är alltid omgivna av citationstecken. De citat där vi själva har översatt från ursprungsspråket, oftast engelska, är texten direkt efter det återgivna stycket märkt med ”vår översättning” (utan citationstecken). Text återges i sitt ursprungliga format om inget annat anges. I de fall där vi återger ett resonemang eller en slutsats med egna ord eller i starkt förkortade drag ger vi enbart en referens till källan.

För andra formalia har vi använt oss av instutitionen för informatiks riktlinjer gällande rapportskrivande (Olerup & Steen, 2004a) samt av Bjerstedt (1997). I appendix C återfinns en fullständig källförteckning.

1.2 Bakgrund

Vi anser att fri programvara och öppen källkod (FOSS), vid tidpunkten för uppsatsens skrivande, vara ett ytterst aktuellt ämne. FOSS är enligt vårt förmenande dock inte någon helt ny företeelse.

Författarna har kommit i kontakt med FOSS på olika nivåer och därigenom kommit att intressera sig för olika frågor och problemområden kring fri programvara och öppen källkod. Därför ser vi det som naturligt att göra en studie av de frågor som intresserar oss mest inom ämnet.

1.3 Syfte

Den fråga som starkast har fångat vår uppmärksamhet och vårt intresse är: ”Hur kan pengar tjänas på FOSS”, eller med andra ord, ”Hur kan man tjäna pengar på något som är gratis?”.

Uppsatsen syftar till att försöka bena ut denna frågeställning. Ordet ”hur” i vår frågeställning bearbetar vi på två huvudsakliga sätt då vi väljer att se ordet dels i meningen ”på vilka sätt” och dels i betydelsen ”hur kommer det sig att”. I det första perspektivet knyter vi an till ekonomisk teori. Mer specifikt vill vi i detta perspektiv även undersöka om vi kan finna nya vägar att göra fri programvara och öppen källkod mer tilltalande för dagens IT-entreprenörer. För att förstå det senare perspektivet kommer vi bland annat presentera olika angreppssätt för att licensiera programvara och specifikt fri programvara och öppen källkod. Vi ämnar även konkretisera olika licenstyper genom ett antal exempel. De båda perspektiven på vår frågeställning har självfallet ett otal knytpunkter, men vi har likväl valt att lyfta fram dem som två delvis separata perspektiv med förhoppningen att vi på så sätt kan underlätta förståelsen för vår läsare.

För att kunna ge svar på de frågor vi ställde upp behövs emellertid en teoretisk grund att vila våra analytiska resonemang på, varför vi även kommer at ge en allmän översikt av fenomenet FOSS och likaså koppla samman fri programvara och öppen källkod till befintlig ekonomisk teori. Detta innefattar även andra aspekter som är intressanta för att förstå företeelsen FOSS.

Vi kommer sålunda att undersöka fenomenet med fri programvara och öppen källkod ur perspektiv som om de utgör ett hot mot kommersiell programvara, om det går att tjäna pengar på FOSS och i samband med detta kommer vi även att beröra frågan om för vem det passar att utnyttja denna typ av programvaror och licensieringsformer. Vi kommer även att ge exempel på program som erbjuds som FOSS och de olika villkor som FOSS kan föra med sig.

1.4 Målgrupp

Vi riktar oss med denna uppsats främst till en publik med intresse för såväl teknik som ekonomi och med funderingar kring hur FOSS faktiskt kan fungera i en kommersiell värld. Läsaren antas åtminstone ha en grundläggande insikt i tekniska begrepp såsom exempelvis källkod, kompilering, objektkod och länkning.

1.5 Avgränsning

För att vi ska kunna nå i mål med denna uppsats föresats är det nödvändigt att dra vissa gränser för vår undersökning.

Frågor som hur ett FOSS-projekt drivs rent praktiskt eller de mer ideologiska tankarna bakom fri programvara och öppen källkod berörs inte alls eller bara kortfattat. Vi kommer inte heller att försöka ställa upp jämförelser mellan enstaka program som är sluten programvara respektive Open Source eller Free Software. Exempelvis avser vi inte att jämföra Linux med Microsoft Windows eller ställa

Apache mot Internet Information Server (IIS) eller Mozilla Firefox mot Internet Explorer. Vi analyserar inte i detalj hyllvara (engelska COTS Software, Commercial of the Shelf Software") såsom exempelvis kontorsprogrampaketet Microsoft Office.

För utvecklingsmodeller beskrivande mjukvaruutveckling redogörs endast ytligt och med utgångspunkt i litteratur kring resonemang rörande öppen och stängd mjukvara.

Inom ramen för vårt ämne kommer vi att beakta och till stor del använda oss av ekonomisk litteratur och teori, något vi även använder för att dra slutsatser om företeelsen Open Source och för att svara på våra frågeställningar, men vår utgångspunkt är inte strikt företagsekonomisk. Sålunda intresserar oss inte frågor som om det är möjligt att spara licenspengar på att gå över till FOSS eller räkneexempel med Total Cost of Ownership (TCO) och liknande speciellt mycket och sådana jämförelser har redan gjorts, exempelvis i Wheeler (2004a) och i Cybersource (2002). Vi gör inga anspråk på att göra en fullständig redovisning av ekonomisk litteratur och ekonomiska modeller utan berör de områden som vi anser vara intressanta för oss.

Utöver jämförelser mellan kostnader för olika typer av licenser kommer vi heller inte att försöka utreda lagliga aspekter för användning och patentering av datorprogram vad gäller programlicenser och dylikt, vi nöjer oss med att definiera vad en licens är och vilken betydelse den har för användaren.

1.6 Relaterade arbeten

Vi har undersökt tidigare publikationer och uppsatser kring ämnet fri programvara och öppen källkod. Vi har funnit tre tidigare uppsatser framlagda vid instutitionen för Informatik vid Lunds Universitet som helt eller delvis berör ämnet FOSS.

Innehållsmässigt fann vi emellertid att ingen av uppsatserna direkt interfererade med vår frågeställning. De olika uppsatserna innehåller dock samtliga delar som vi tangerar i vår uppsats, dock på lite olika plan. Metodmässigt finns inte någon reell likhet med vårt arbete då samtliga dessa tre uppsatser använt sig av enkätundersökning som undersökningsmetod.

Anderberg, Martin och Dahlberg, Magnus (1999): Säkerhet i Linuxbaserade system, Magisteruppsats

Uppsatsen "Säkerhet i Linuxbaserade system" (Anderberg & Dahlberg, 1999) handlar om säkerhet i allmänhet och mer specifik säkerhet i Linuxbaserade system. Metoden som författarna använder sig av kretsar kring en enkätundersökning och tre intervjuer som sedan leder till en lista över saker som en systemadministration behöver göra för att säkra ett Linuxsystem. Denna uppsats berör endast delvis vår frågeställning och då i meningen som resonemang kring för– och nackdelar med

FOSS. Säkerhetsfrågor som sådant relaterat till olika program eller operativsystem intresserar vi oss inte för i vår uppsats vilket framgår av vår avgränsning ovan.

Nordström, Rickard, Schmidt, Peder (2000): Open-source: En studie av metoden ur ett kvalitetsperspektiv, Lunds Universitet, Institutionen för Informatik, Magisteruppsats

Nordstöm och Schmidt (2000) fokuserar i sin magisteruppsats ”Open-source: En studie av metoden ur ett kvalitetsperspektiv” på själva utvecklingsmodellen och särskilt hur den påverkar kvalitén på mjukvaran som produceras. Frågan som författarna ställer sig är: *”Hur står sig decentraliserad och distribuerad mjukvara kvalitetsmässigt gentemot traditionellt utvecklad?”* (Nordström & Schmidt, 2000 sid. 2). Frågan som författarna inriktar sig på är i och för sig besläktad med vår forskningsfråga, men fokus skiljer sig skarpt mellan våra arbeten. Nordström och Schmidt har vidare använt sig av enkätundersökningar för att hämta in data som de senare analyserar. Även här skiljer sig våra arbeten åt då vi har valt att genomföra en teoretisk studie av ämnet.

Nilsson, Kristoffer (1998): Operativsystem, Lunds Universitet, Institutionen för Informatik, Kandidatuppsats

Nilsson (1998) försöker i sin uppsats titulerad ”Operativsystem” utröna *”vilka faktorer företag har i beaktande när de väljer operativsystem”* (Nilsson, 1998 sid. 1). I uppsatsen använder sig Nilsson bland annat av enkätundersökningar och intervjuer. En av författarens frågeställningar kretsar kring vad företag anser om ”freewareoperativsystem”. Med freewareoperativsystem avser Nilsson (1998) främst Linux. Vi skulle istället ha använt termen Open Source eller Free Software då freeware egentligen bara avser ett program som kan användas utan kostnad. Hursomhelst så är Nilsson centrerad på att behandla de rent tekniska meriterna hos olika operativsystem. I dagsläget känns en sådan jämförelse gammal speciellt avsnitten kring Windows 95, MacOS och OS/2.

De empiriska slutsatser som Nilsson kommer fram till vad gäller faktorer som styr företags val av operativsystem är: *”Funktionalitet, Lätt använt/administrerat. Kompabilitet/Integrering med befintliga system”* (Nilsson, 1998 sid. 48). När uppsatsen skrevs innebar dessa faktorer att Windows var det dominerande valet för företag.

Vi har också funnit en licensiatavhandling framlagd vid Chalmers tekniska högskola, ”Networks of Innovators and the Private-Collective Innovation Model: Why Do Firms Engage in Open Source Software?” författad av Linus Dahlander (2004). Denna avhandling behandlar till stor del vårt ämne men har en lite annorlunda vinkling. Dahlander fokuserar på de processer som styr företags agerande i en marknad som påverkas av förekomsten av FOSS.

Dahlander (2004) undersöker bland annat olika företag och hur de använt sig av FOSS samt i förlängningen hur de gjort för att generera intäkter med hjälp av FOSS.

Avhandlingen består egentligen av tre olika delar med lite olika forskningsvinkling. Dahlander kommer fram till ett antal sätt med vilka intäkter kan genereras men han kommer också fram till att det inte är helt problemfritt. Världen är ju sällan enkel att förstå, så inte heller marknaden för FOSS/sluten källkod. I gränslandet mellan dessa finns många problem att resonera kring. Dahlander lyfter fram dessa och föreslår vidare forskning kring vissa av dem, exempelvis problematiken att andra kan dra nytta av FOSS genom att med hjälp av olika licensieringsformer och lite eget arbete stänga källkod och på så sätt ta betalt för arbete utfört av andra. Vi skall återkomma till dessa problem senare i denna uppsats.

1.6.1 Sammanfattning

Inget av de arbeten som vi har bekantat oss med rör direkt samma frågeställning som vi har ställt upp, närmast ligger då avhandlingen av Dahlander.

Vidare bygger samtliga uppsatsarbeten och avhandlingen som vi har bekantat oss med på empiriska undersökningar främst baserade på enkäter. Vårt arbete har både en annorlunda inriktning samt bygger på en rent analytisk metod, låt vara med inslag av empiri från tredje part.

1.7 Metodbeskrivning

I nedanstående avsnitt kommer vi att resonera kring vårt val av metod och problem där kring.

1.7.1 Problemformulering

När vi bestämt vad vi ville undersöka ställdes vi inför nästa problem: *Hur* skulle vi genomföra vår undersökning och med vilka metoder?

Vi funderade då över de olika alternativ som stod till buds. Till exempel skulle vi kunna ställa upp en enkät med för ämnet relevanta frågor och skicka den till för vårt syfte intressanta företag, exempelvis både företag som jobbar med FOSS-verktyg och företag som inte gör det. En annan möjlighet vore att först skicka en enkät med lösa frågor till intressanta företag och sedan bland svaren välja ut de mest intressanta för en djupare intervju eller mer detaljerad enkät. Det finns otaliga sätt att kombinera olika undersökningsmetoder, det gäller att välja metod för varje enskild undersökning baserat på förutsättningarna i det enskilda fallet. Denscombe beskriver bland annat detta i sin bok ”Forskningshandboken” (Denscombe, 2000). Han skriver att

forskaren har vissa möjligheter vad gäller metodanvändningen, men att *"Forskarna bör fråga sig själva vilken metod som är bäst lämpad för uppgiften..."* (Denscombe 2000, sid. 101). Vi kommer i detta kapitel att förklara vilken metod vi har valt att använda och varför vi tycker att denna metod lämpar sig väl för vår uppgift.

1.7.2 Olika metoder

När det gäller metodval måste vi även definiera vilken metod vi menar i uppsatsen. Bjurwill (2001) beskriver två olika typer av metod; En metod för arbetet med att samla in data (som Bjurwill kallar sökmetod) och en metod för att analysera denna insamlade data (kallad analysmetod). Metoden för datainsamling definierar alltså hur vi samlar in data medan metoden för analys av dessa data definierar hur vi gör när vi tolkar den data vi har samlat in. Den första metod vi måste definiera rör då insamlandet av data eller med andra ord sökmetoden.

1.7.2.1 Sökmetod

Denscombe(2000) identifierar fyra forskningsmetoder för detta ändamål:

- Frågeformulär
- Intervjuer
- Observation
- Skriftliga källor

Dessa metoder menar han kan användas var för sig eller tillsammans, men eftersom de är olika till sin natur passar de olika bra i olika sammanhang. Patel och Davidson (1994) tar även upp självrapportering i form av dagböcker som en metod för att samla in data.

Eftersom vi hade preciserat frågan vi ville ha svar på funderade vi över hur vi bäst skulle få svar på denna. Eftersom vi redan innan arbetet med denna uppsats startade studerat ämnet fri programvara och öppen källkod i litteratur och dagspress samt till stor del också själv använt FOSS-verktyg fann vi det naturligt att använda oss i viss mån av egna erfarenheter. Huvuddelen av vårt informationssökande var vi dock tvungna att söka oss på annat håll för att få fram, eftersom vi inte själv har den breda erfarenhet av ämnet som krävs för att kunna ge en vetenskapligt täckande bild av ämnet. Vi vände oss då i tur och ordning till de olika alternativ vi hade enligt forskningsmetoderna ovan.

Tid och utrymme blev då en viktig faktor för vårt resonemang kring detta, samt naturligtvis det förväntade utfallet av metoderna.

Frågeformulär funderade vi sålunda kring först. Vid upprättandet av frågeformulär gäller det att noggrant utforma frågorna för att få relevanta och reliabla svar från dem som besvarar frågorna, respondenterna. Urvalet av respondenter är naturligtvis också viktigt, och frågan om kvantitativ eller kvalitativ undersökning blir då också automatiskt intressant. För att utreda om frågeformulär var något för oss granskade vi därför bland annat en magisteruppsats skriven vid Lunds Universitet i ämnet informatik (Nordström & Schmidt, 2000) och fann där en stark anledning till att inte använda frågeformulär för vår undersökning: Respondenterna (som ställs frågor kring användandet av Open Source som metod) visade lågt intresse av att svara. Detta i kombination med den tid som krävs för att utforma frågorna gjorde att vi kände osäkerhet inför vad vi skulle komma att få ut av denna metod. Vi kan naturligtvis inte vara helt säkra på att de personer vi valt ut för att besvara frågeformulären skulle ha lika låg svarsfrekvens, det är ju en fråga om personer och attityder lika väl som tidsaspekter. Denscombe (2000) skriver bland annat om detta att forskaren för att bedöma uppnådd svarsfrekvens kan resonera kring om svarsfrekvensen är rimlig och i överensstämmelse med jämförbara undersökningar. Det är också mycket viktigt att ställa sig frågan hur de som inte svarat skiljer sig från gruppen som svarat på undersökningen (Denscombe, 2000). Inom ramen för denna uppsats såg vi då en risk att alltför mycket tid skulle gå åt till att analysera och jämföra olika personers inställningar till ämnet och varför de inte besvarat vår undersökning i stället för att kunna koncentrera oss på vår kärnfråga.

Ett annat som vi ser det tungt skäl till att inte välja denna metod är också att eftersom FOSS-frågan i sig själv ställer två världsbilder mot varandra var risken stor att vi skulle få en typ av svar från FOSS-förespråkare och en annan typ av svar från motståndare/skeptiker till fri programvara och öppen källkod. Dahlander (2004) gör en poäng av att av de respondenter han använt sig av för att undersöka operativsystemet Linux hade endast ungefär hälften använt sig av Linux regelbundet (utgångspunkten var att respondenterna hade minst Magisterexamen i datavetenskap). Med detta i åtanke såg vi en risk att vi skulle få starkt vinklade svar som vi skulle kunna få svårt att tolka på ett för uppsatsen och vårt syfte givande sätt, då vi inte ville gräva ned oss för djupt i resonemang och argumentation av typen för- och emot FOSS.

Med detta sagt har vi också i stort sett förklarat varför vi valt bort intervjun som metod. Även om man i intervjun som forskare har lite mer utrymme för att vara flexibel (strukturerad, semi-strukturerad intervju) och därmed få fram intressanta synpunkter och kommentarer från initierade personer måste ändå en hel del tid på förberedelser läggas ned (Patel & Davidson, 1994). Dessutom blir forskaren beroende av att de personer man vill intervjua ställer upp. Vi ville därför hellre lägga vår tid på något som vi själva trodde kunde ge solida resultat än på något med hög osäkerhetsfaktor.

Vad gäller observation har vi inte använt oss av denna metod sett ur ett strikt vetenskapligt perspektiv. Vi har i vissa fall i vårt yrkesliv gjort intressanta observationer som vi försöker använda oss av, men det som exempelvis Denscombe (2000) tar upp (två olika typer av observationer; systematisk och deltagande observation) kan vi inte påstå oss ha använt oss av som metod. Systematisk observation innebär mätning av frekvens eller löptid för olika händelser medan

deltagande observation innebär att undersöka och observera människor eller situationer i det dagliga livet. Systematisk observation ger alltså i första hand kvantitativ data medan deltagande observation ger kvalitativ data (Denscombe, 2000).

Det är dock inte heller våra egna erfarenheter och observationer vi valt att lägga tyngdpunkten för datainsamling på. Vi har gjort våra observationer/erfarenheter som en del av vårt normala arbete och använder dessa som ett komplement till övrig litteratur vi använt oss av för att beskriva fenomenet FOSS och ge ett svar på den frågeställning vi försöker besvara med denna uppsats. Författarna har vid uppsatsens skrivande sammantaget 15 års arbetslivserfarenhet inom IT-sektorn, med roller som bland annat konsult, programmerare, projektledare och utbildare. Under vår yrkesutövning har vi stundtals använt oss av fri programvara och öppen källkod. Då vi har valt att företa en litteraturstudie kommer emellertid våra egna empiriska inslag endast att fungera som ett komplement till den litteratur vi använt oss av. Rent metodiskt har vi alltså inte använt oss av självrapportering i form av dagböcker eller liknande från vårt yrkesliv, de erfarenheter vi använder oss av i denna uppsats är baserade på vår samlade erfarenhet och våra samlade intryck under vårt hittillsvarande yrkesliv.

Då återstår metoden att använda skriftliga källor. Vår undersökning går till stor del ut på att finna specifika egenskaper i fri programvara och öppen källkod, försöka förstå och i slutändan svara på frågan om- och i så fall hur de kan fungera i en marknad. Då undersökningen till stora delar är uppbyggd på rent teoretiska resonemang kommer studier av teoretiska modeller vara en essentiell ingrediens. För att nå i mål med vår föresats anser vi därför med stöd av ovanstående resonemang att enkäter, intervjuer och observationer helt enkelt inte fungerar som adekvata verktyg, utan vårt primära redskap är studier av skriftliga källor.

Någonstans var vi tvungna att börja gräva och således anskaffade vi ett par böcker i ämnet och vi sökte även efter information på Internet. De källor vi använt är sådana vi har ansett är relevanta för vår ansats med uppsatsen. Bland den tryckta litteratur som vi har använt oss av märks bland annat Rosenberg (2000) och Raymond (2001a) som behandlar flera aspekter av företeelserna fri programvara och öppen källkod.

Eftersom undersökningsområdet utvecklas i en snabb takt så har Internet varit en viktig, om än stundtals osäker, källa att inhämta aktuell information från.

1.7.2.2 Analysmetod

Upplägget för uppsatsen är således att vi valt att göra en litteraturstudie av ämnet. Vi har följaktligen valt att inte primärt ta ett introspektivt grepp och forska i våra egna erfarenheter av att använda FOSS eller dra slutsatser från när vi själva har involverat oss i FOSS-projekt. För att besvara de frågeställningar vi ställt upp i början av denna uppsats måste vi emellertid ha en metod också för att analysera vårt material och komma fram till relevanta svar. Det räcker inte med att samla in och sammanställa fakta från olika källor, utan Bell (1995) menar att insamling av fakta bara är en del av en forskares uppgift. Forskaren ska efter insamling organisera och klassificera informationen i ett sammanhängande mönster.

Vi beslöt oss för att försöka penetrera frågan genom en teoretisk analys och många diskussioner sinsemellan, ett slags dialektiskt grepp. Hamlyn (1994) skriver att *"… den dialektiska teorin försöker arrangera begrepp i termer av test, antites och syntes"* (Hamlyn, 1994 sid. 263). Vår ansats är inte att nå fram till vad Hamlyn (1994) kallar en "absolut syntes", men vi hoppas likväl att vi genom att ställa olika argument för- och emot FOSS bättre kan förstå dess natur och möjligheter att utvecklas i en kommersiell värld. Vi försöker dock inte på något sätt inta någon fundamentalistisk position och helt ignorera empirin. Hansson (1992) menar att en extremt rationalistisk kunskapsteori kan lätt förefalla absurd, eftersom vi har en så rik erfarenhet av direkta kontakter med omvärlden. I det faktum att vi i viss mån tar hänsyn till våra tidigare erfarenheter parallellt med ett teoretiskt resonemang antar vi en tämligen moderat ställning.

Vi har med upplägget för vår undersökning valt att använda en kvalitativ metod för vår databearbetning. När det gäller analys av kvalitativ data finns det vissa utmaningar den som forskar ställs inför. Det finns i vår undersökning inga kvantifierbara mätvärden i form av exempelvis svar på enkätfrågor eller liknande att analysera. De möjliga mätvärden vi kan tänka oss skulle kunna vara rent ekonomiska mätetal vid jämförelser mellan olika typer av projekt eller programvaror, men eftersom vi avgränsat oss i denna undersökning till att inte undersöka den typen av frågor faller detta alternativ bort, vi försöker i stället påvisa vissa faktorer som vi kan använda oss av för att göra jämförelser och dra slutsatser. Exempel på sådana faktorer är ekonomiska modeller och olika former av licenstyper inom och utom FOSS-världen. Vi använder dessa faktorer blandat med exempel från marknaden och våra egna erfarenheter.

När det gäller tolkningen av insamlad data presenterar Miles och Huberman (1994) ett antal koncept för att lyckas med detta på bästa sätt. Eftersom vi genomför en teoretisk studie utan några intervjuer eller enkätundersökningar är inte alla metoder direkt applicerbara för oss. Vad vi i huvudsak inriktar oss på är vad Miles och Huberman kallar att uppfatta relationer mellan variabler och bygga logiska beviskedjor.

Miles och Huberman påvisar när det gäller kvalitativa undersökningar att när det arbetas med text eller mindre organiserat material är ett sätt att dra slutsatser att forskaren noterar återkommande mönster som plötsligt blir synliga för forskaren. Detta är vårt huvudsakliga sätt att dra slutsatser, vi noterar mönster och trender i litteratur och verkliga exempel. Vi försöker, eftersom vi gör en litteraturstudie med endast begränsade inslag av empiri, dra slutsatser på detta sätt genom korsreferenser mellan exempel, teori och egen erfarenhet. Speciellt effektiv anser Miles och Huberman att denna metod är när mängden data eller undersökningsobjekt är mycket stor, vilket vi anser passar väl in på vår undersökning.

Vidare kommer vi bland annat att använda oss av matriser i vårt analyskapitel. När matriser ställs upp menar Miles och Huberman att det är viktigt att fundera över hur många nivåer man lägger in i matrisen och hur man grupperar dem. För att behålla överblicken bör man inte heller göra matrisen större än att den får plats på ett papper. Matriser kan enligt Miles och Huberman (1994) användas på flera sätt och

med flera syften, den största fördelen med användandet av dem är att de går förhållandevis snabbt att göra och ger gott resultat.

Miles och Huberman (1994) beskriver också hur forskaren lätt kan hänfalla till att anta saker som verkar uppenbara, vilket kan vara fallet även för vår undersökning. Det behöver dock inte enligt författarna vara något negativt utan kan tvärtom bidra till att föra forskningen på rätt spår, så länge forskaren kan visa och motivera det grundläggande antagandet.

1.7.2.3 Reflektioner kring val av metod

Den fråga vilken vi försöker besvara är skenbart enkelt formulerad och rymmer vi en närmare studie många bottnar och när vi dök ner i frågeställningen såg vi många nya aspekter reflekteras. Våra egna slutsatser baseras på tidigare litteratur och observationer från marknaden och den modell vi presenterar i analyskapitlet skall närmast ses som ett redskap för ökad förståelse av ämnet och en skulptur av marknaden för fri programvara och öppen källkod för att ha något att senare kunna argumentera kring. Enligt Hansson (1992) ställer en av de ledande vetenskapsteoretikerna nämligen Karl Popper följande kriterium för en teori: *"en vetenskaplig teori är falsifierbar, om den har testbara konsekvenser (åtminstone i teorin)"* (Bertil Mårtensson, i Hansson, 1992 sid. 23-24). Att som Popper ställa principiell falsifierbarhet som ett demarkationskriterium låter sig inte enkelt göras för eventuell teori i vårt ämne, ja i strikt mening är det svårt utanför logiken och möjligen fysiken. I Hansson (1992) beskrivs även en annan extrem i Feyerabend, vars "*viktigaste poäng är att det inte existerat något unikt avgörande kriterium – crucial experiments – eller dylikt, med vars hjälp man skulle kunna fälla avgörandet mellan konkurrerande teorier"* och från detta intar Feyerabend en i det närmaste anarkistisk inställning och argumenterar för en vittgående tolerans (Svante Nordin, i Hansson, 1992 sid. 55). Vår ambition är att lägga oss någonstans mitt emellan de två motsatta poler som kortfattat redogjorts för ovan.

1.8 Käll- och metodkritik

Valet av undersökningsmetod, där vi fäster stor vikt på andras observationer snarare än egen empiri gör det om möjligt än viktigare med källkritik. Vad gäller de källor som vi har använt oss av finns det en klar bias, då huvuddelen av källorna är pro FOSS. Judith Bell skriver att *"Det är viktigt att noggrant undersöka källmaterialet för att avgöra om en eventuellt politisk övertygelse kan ha påverkat stilen eller fokus i en beskrivning och för att kunna komma fram till en slutsats som har en verklighetsförankrad grund."* (Bell, 1995 sid. 69), och fortsätter lite senare, *"En författare klargör sällan sina utgångspunkter eller grundläggande värderingar och därför får det bli forskarens uppgift att visa på dem – om detta nu är möjligt att göra"* (Bell, 1995 sid. 69). Att klargöra de olika författarnas utgångspunkter och värderingar görs kanske enklast genom att se för vilket företag de arbetar och om vilka områden de skriver.

Överlag har vi till stor del hämtat våra källor från Internet. Detta är nästan ett krav med det ämne vi behandlar då mycket av den skrivna texten om FOSS finns på Internet och inte i tryckt bokform. Detta är både bra och dåligt för forskaren. Bra i den meningen att informationen är lättillgänglig för alla (vilket ju är en av grundtankarna med FOSS i sig) men dåligt i den meningen att det blir svårare att validera innehållet i den skrivna texten. Bell (1995) skriver att oavsett vilken metod som har använts för informationsinsamlingen så måste informationen kritiskt granskas ur olika perspektiv. Den första aspekten författaren behandlar är reliabilitet (eller tillförlitlighet) vilken beskrivs som *"ett mått på i vilken utsträckning ett instrument eller tillvägagångssätt ger samma resultat vid olika tillfällen under i övrigt lika omständigheter"* (Bell, 1995 sid. 62). Ett sätt att satisfiera reliabiliteten i vårt fall är att försöka samköra uppgifter från olika källor. Speciellt viktigt är detta för webbreferenser av "tveksamt" ursprung, där innehållet ibland snarare är åsikter än data baserat på gedigen forskning, eller subjektivt snarare än objektivt om man så vill.

Validitet (eller giltighet) kan ses som *"... ett mått på om en viss fråga mäter eller beskriver vad man vill att den ska mäta eller beskriva"* (Bell, 1995 sid. 63). Reliabilitet är en förutsättning, men ingalunda någon garanti för, validitet. För vårt vidkommande rör det sig kanske främst om att vara försiktiga i att dra slutsatser från ett område till ett annat.

Även de slutsatser vi drar från litteratur och egna exempel kan naturligtvis diskuteras i form av begrepp som validitet och reliabilitet. Våra tidigare erfarenheter menar vi påverkar undersökningens reliabilitet och validitet på två olika sätt. Positivt i meningen att våra erfarenheter kan hjälpa oss att värdera såväl trovärdigheten hos våra källor samt de slutsatser vi drar på ett positivt sätt (exempelvis kan direkt felaktiga eller missvisande källor eller felaktiga orsakssamband genom mönster i litteraturen elimineras) men negativt i meningen att vår objektivitet riskerar att påverkas. Vi har under arbetets gång, samt med vår egen bakgrund, haft indikationer på att FOSS har goda egenskaper och det har också påverkat hur vi analyserat den teori vi tagit upp i våra teorikapitel. Vi har under arbetets gång försökt vara medvetna om vår egen och materialet vi använts bias och hantera det genom att jämföra med andra källor med avvikande uppfattning i den mån det har varit möjligt.

De debatter som förs på Internet har i många fall varit hetsiga och förts med överdrivna och direkt felaktiga uppgifter och mest kommit att handla om smutskastning. Vi har i möjligaste mån försökt undvika att ta med sådana debatter som underlag till uppsatsen, men vi kommer ändå att beröra dem i de fall vi finner att de kan bidra med förståelsen för ämnet.

Metodvalet för denna litteraturstudie kan naturligtvis diskuteras. Med ett ökad inslag av empiriska studier hade vi möjligen kunnat få en djupare insikt i hur företag på marknaden aktivt arbetar med produkter baserade på FOSS relaterat till möjligheterna till ekonomisk vinst. De val som gjorts i denna litteraturstudie har vi dock försökt motivera. Vi anser också, baserat på våra yrkeserfarenheter, att det finns en utbredd osäkerhet hos allmänheten såväl som hos yrkesverksamma inom alla branscher kring vad FOSS egentligen innebär. I någon mån har vi också riktat in vårt metodval på detta, att från en bredare bas klargöra hur grundtankarna bakom FOSS egentligen ser ut och vad det innebär för utvecklare och användare.

1.9 Uppsatsens struktur

Uppsatsen kan kategoriseras i tre huvudsakliga delar.

Del I. Inledning (Kapitel 1)

I detta avsnitt har vi försökt sammanfatta vår frågeställning och våra angreppssätt.

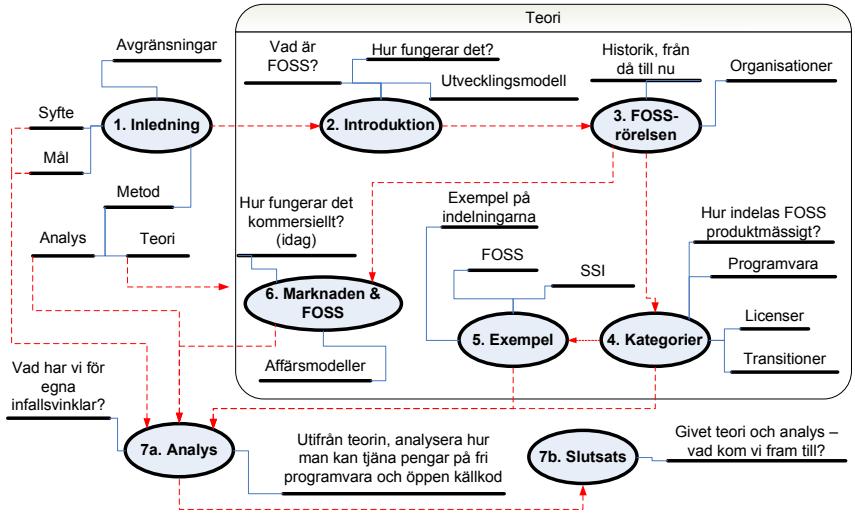
Del II. Teori och frågeställningar (Kapitel 2-6)

I den här delen beskrivs FOSS och rationaliteten bakom fenomenen. För att underlätta den vidare avhandlingen av ämnet så kommer vi först att gå igenom lite datorhistorik för att lägga grunden för en resa genom historien som har lett fram till begreppet FOSS. Vi kommer även att stifta bekantskap med olika typer av mjukvarulicenser och studera ett antal exempel på program med såväl öppen som stängd källkod. Avslutningsvis kommer vi i detta kapitel beskriva ekonomiska teorier och modeller vilka är relevanta för en analys av förutsättningarna för fri programvara och öppen källkod i en marknadsekonomi.

Del III. Analys – Analys, diskussioner och slutsatser (Kapitel 7)

Avslutningsvis för vi en diskussion där olika analytiska grepp tillämpas för att försöka förstå fenomenet FOSS och ge ett svar på om och i så fall hur samt varför fri programvara och öppen källkod kan fungera i en kapitalistisk värld.

Vi har valt att grafiskt illustrera uppsatsens struktur i en strukturkarta som återges i figur 1.1 nedan.



Figur 1.1: Karta över uppsatsens struktur

I figur 1.1 ovan ser vi hur två kopplingar utgår från teorin till analysen. Dessa kopplingar motsvarar de två huvudperspektiv vi har spänt upp mot forskningsfrågan ”Hur kan pengar tjänas på FOSS?”. Perspektivet ”på vilka sätt” knyter an till kapitlet ”Marknaden och FOSS” i teorin och perspektivet ”hur kommer det sig att” knyter närmast an till kapitlen kring ”Kategorier av licenser” och ”Exempel på programvara ur olika kategorier” i teorin.

Uppsatsen kan läsas på olika sätt beroende på var läsarens huvudsakliga intresse närmast ligger. För en läsare som främst är intresserad av de ekonomiska områdena av uppsatsen kan kapitel 3, 4 och 5 läsas översiktligt. Läsaren vars intresse snarare ligger kring tekniska och historiska detaljer kan rikta in sig på dessa kapitel istället för kapitel 6.

Utöver genomgången och analyserna i kapitel ett till sju, innehåller uppsatsen även en allmän referenslista där den intresserade läsaren kan finna länkar till mer information (appendix A) och en ordlista som förklarar de för ämnet centrala begrepp och förkortningar som vi använder oss av i uppsatsen (appendix B). En källförteckning med den litteratur som refereras till i uppsatsen återfinns slutligen i appendix C.

Vi har en lång väg att vandra innan området kring fri programvara och öppen källkod är avhandlat, så låt oss skrida till verket och börja med att ta tag i den teoretiska genomgången!

2 Introduktion till FOSS

I detta kapitel kommer vi att förklara vad fri programvara och öppen källkod (Free Software and Open Source Software, FOSS) innebär samt även beskriva dess motsats stängd proprietär programvara i syfte att teckna en inledande bakgrund till vidare resonemang.

Fri programvara och öppen källkod är emellertid inga nya företeelser. Genom att bekanta oss med dess historia kan förhoppningsvis en ökad förståelse för företeelsens nuvarande utseende och position nås. Vi inleder därför med en översiktlig genomgång av historiken bakom FOSS.

2.1 Historik

Låt oss göra en summarisk historisk tillbakablick och kort beskriva hur datorer och programvara till dessa har utvecklats sedan några årtionden tillbaka.

1960-1970

De första datorprogrammen som utvecklades var starkt knutna till datortillverkarna. När ett system såldes skeppades såväl hård- som mjukvara i ett ”paket”. Vid den här tidpunkten gjordes inga försök att skilja på hård- respektive mjukvara ur ett licensierings- och distributionsperspektiv. Låt oss ta Digital Equipment Corporation (DEC) som exempel. DEC hade ingen copyright på sitt operativsystem och de gav dessutom sina kunder en samling andra program för en kostnad av fem dollar som i princip enbart skulle täcka kostnaden för distributionsmediet (ett band) (Rosenberg, 2000).

UNIX och C

På Bell Labs i USA utvecklade Kenneth Thompson operativsystemet UNIX år 1969. Några år senare skrev Dennis Ritchie programmeringsspråket C och Thompson kom att skriva om UNIX i C (Bell-labs, 2004a, 2004b).

En grundtanke med UNIX var att skriva ett operativsystem som skulle fungera på flera olika hårdvaruplattformar.

För hårdvarutillverkare var mjukvaran till att börja med mest något som var tvungna att skeppa med för att kunna sälja sina maskiner, men tillverkarna började också inse

att mjukvaran faktiskt kunde binda kunden till deras maskiner eftersom program inte kunde flyttas från en plattform till en annan (Rosenberg, 2000). UNIX utgjorde ur denna aspekt ett hot mot hårdvarutillverkarna. UNIX spreds först i universitetsvärlden och dess källkod kunde studeras. Detta ledde till att buggrättningar och förbättringar strömmade in till Bell från användare.

När AT & T delades i slutet av 1970-talet sågs UNIX som en kommersiell tillgång och källkoden blev avgiftsbelagd. UNIX började dock mer och mer sprida sig till nya hårdvaruföretag som behövde ett operativsystem till sina plattformar och som därför licensierade UNIX. Istället för att bygga vidare på UNIX grundläggande egenskaper att vara portabelt och interoperabelt valde de nya UNIX-företagen att låsa upp sina kunder genom att göra egna specifika modifikationer av UNIX (Kenwood, 2001; Rosenberg, 2000). Ur ett valfrihetsperspektiv var kunderna på väg tillbaks mot ruta ett.

The University of California at Berkeley var ett starkt fäste för UNIX-utveckling och de distribuerade sin egen version av UNIX kallad BSD. BSD UNIX var i många avseende bättre än Bells ”original” och då BSD var fritt att ändra och vidaredistribuera kom även många kommersiella UNIX-dialekter, däribland Sun Microsystems tidiga operativsystem, att bygga vidare på BSD.

Efterhand och bland annat på grund av upphovsrättsliga tvister kom BSD UNIX att skrivas om så att dialekten inte längre innehöll kod från Bells UNIX (Rosenberg, 2000).

Persondatorn

Under senare år har ett antal dominerande hårdvaruplattformar utkristalliseras och då framförallt persondatorn (PC:n) som så småningom kom att slå igenom stort med IBM:s PC-serie från år 1981 med PC-DOS (MS-DOS) som ett alternativ. PC-datorer, eller mikrodatorer, hade dock redan funnits sedan mitten av 1970-talet (Lunell, 1991). Lowendpc (2004) menar att termen PC bytt betydelse efter IBM:s lansering: “Ever since IBM, the term PC has taken on a second meaning. Although it retains the original meaning of "personal computer," the IBM architecture has so dominated the industry that it soon came to mean IBM compatible computers to the exclusion of other machines.” Lowendpc (2004). En anledning till PC:ns spridning var att andra tillverkare kopierade IBM:s beskrivning av PC:n och saluförde så kallade PC-kompatibler. Program skrivna för en IBM PC fungerade även på en PC-kompatibel maskin (Lunell, 1991).

Microsoft som hade hand om MS-DOS träffade 1981 ett avtal med IBM som gav dem rätten att sälja MS-DOS till andra hårdvarutillverkare. Detta banade vägen för en hel industri med PC-kloner, vilket i så småningom marginaliserade IBM:s inflytande och Microsoft lade grunden till sitt monopol (Lunell, 1991; Lowendpc, 2004).

Tankarna bakom fri programvara och öppen källkod kommer som en slags reaktion mot monopolism och instängdhet numera ofta exemplifierat med företaget

Microsoft. Återstoden av kapitlet kommer vi att ägna åt tankarna bakom FOSS och kontrastera med motsatsen Closed Source.

2.2 Vad är FOSS?

FOSS ger som tidigare nämnts var och en rätten att fritt köra, undersöka, ändra och vidaredistribuera ett program (FSF, 2004a). För Free Software gäller också att all mjukvara som distribueras med öppen källkod måste lämnas öppen även om den distribueras vidare, men FOSS är mycket mer än så:

”The verdict of history seems to be that free-market capitalism is the globally optimal way to cooperate for economic efficiency; perhaps, in a similar way, the reputation-game gift culture is the globally optimal way to cooperate for generating (and checking!) high-quality creative work” (Raymond, 2001a sid. 107).

Detta citat är hämtat ur Eric S. Raymonds bok ”The Cathedral and the Bazaar” (Raymond, 2001a). Raymond är en av ”Open Source”-rörelsens centralfigurer och ett av dess mer kända ansikten utåt och vi har därför valt att till stora delar använda hans resonemang för att presentera vad Open Source handlar om. Citatet ovan visar på kärnan i Raymonds resonemang: Det har visats att marknadskapitalism är det optimala sättet att nå ekonomiskt välstånd, är det inte så att gåvokultur är det globalt optimala sättet att generera kvalitetsmjukvara? Raymond för resonemang på flera plan om fördelarna med Open Source. Han jämför traditionell mjukvaruutveckling med modellen av en katedral och utveckling enligt Open Source med en basar.

En annan liknande beskrivning av Free Software och Open Source görs av Olofsson (2003a) som menar att FOSS kan ses som ett sätt att utveckla programvara som är en protest mot den rådande situationen på marknaden för datorprogram och författaren konstaterar att idén med FOSS är *”att ge användare av datorprogram en möjlighet att själv ändra och förbättra sina egna program, vilket ska resultera i att det skapas fler datorprogram av bättre kvalité”* (Olofsson, 2003a, sid. 8).

Det kan vara på sin plats att poängtera att ”fri programvara” (Free Software) främst syftar på fri som i ”yttrandefrihet” och inte som primärt i meningen ”gratis”. Ur ett juridiskt perspektiv är fri programvara *”ett licensieringssystem genom vilket upphovsmannens ekonomiska förfoganderätt används för att ge licenstagaren större frihet att använda programvara”* (Olofsson, 2003a, sid. 15-16). Systemet som möjliggör detta brukar kallas för copyleft och beskrivs närmare i kapitel 4.

Om vi ser till företeelsen ”öppen källkod” (Open Source) så är det en slags vidareutveckling och utvidgning av ”fri programvara” (Free Software). Kortfattat så har begreppet öppen källkod tillkommit för att tillgodose kommersiella intressen samt för att definiera regler för vad som är ”öppen källkod”. Reglerna sammanfattas i dokumentet OSD, det vill säga Open Source Definition (OSI, 2004b). Den främsta skillnaden mot Free Software är det faktum att reglerna i OSD inte ställer som krav

att ”bearbetningar av ett verk måste licensieras enligt samma licens som orginalprogrammet” (Olofsson, 2003a, sid. 19). Kontentan är att Free Software är en striktare delmängd av Open Source. Definitionen av Open Source tillåter exempelvis licenser som gör det möjligt för en användare att distribuera en modifierad version av ett program som hon mottagit med öppen källkod vidare till en annan part utan att hon gör källkoden till den modifierade versionen öppen för mottagaren. Detta hade inte varit tillåtet för ett program under en ”Free Software”-licens såsom GNU GPL. Innebörden GNU GPL och andra licensvillkor kommer att avhandlas mer ingående i senare kapitel.

Vi skall nu närmast bekanta oss med de modeller som Raymond (2001a) ställer upp och också förklara begreppet ”gåvokultur”. Vidare skall vi undersöka andra sätt att se på FOSS och dess motsatser.

2.2.1 Basaren

Vi börjar med att titta på basarmodellen och förklara den eftersom denna symboliserar Raymonds syn på Open Source.

Med basarmodellen menar Raymond (2001a) att ett stort antal programmerare tillåts utveckla programvaran, men all ny kod ”filtreras” genom en liten kärna av utvecklare som ansvarar för programvaran. I den typ av projekt som Raymond beskriver finns inga licenskostnader, alla användare (som också är potentiella utvecklare) får rätt att använda och förändra koden enligt kriterier för den licensmodell som används för projektet. Vi kommer i senare kapitel att stifta närmare bekantskap med olika licensmodeller.

Basarens utvecklingsmodell drivs av ett antal användare/utvecklare som för programutvecklingen vidare. Eftersom all källkod är öppen får dessa utvecklare som regel inte betalt för sitt arbete utan delar med sig av sitt arbete helt gratis. Det är detta som Raymond (2001a) kallar för gåvokultur. Denna gåvokultur hindrar naturligtvis inte företag från att ha anställda (och därmed avlönade personer) som arbetar med utveckling av fri programvara eller öppen källkod, men deras lön betalas då av det enskilda företaget och inte av FOSS-projektet i sig. Detta har också noterats av Görling (2003a) samt Lerner och Tirole (2000), som exemplifierar med företag som Netscape, Sun och O´Reilly. Görling kommer i sin undersökning fram till att *”våra mest berömda Open Source projekt inte är framtagna av datornördar som jobbar gratis, utan av professionella utvecklare som är anställda av kommersiella företag för att bidra till (Open Source) projekt”* (vår översättning) (Görling, 2003a, sid. 2). Vi får anledning att återkomma till dessa ekonomiska inslag senare i kapitlet som beskriver marknadens förhållande till FOSS.

Vad gäller projektgenomförande menar Raymond (2001a) att en av basarmodellens styrkor ligger i att utvecklarna dels är motiverade av intresse för utvecklingsarbetet (eller med andra ord helt enkelt tycker de att arbetet är roligt) samt dels oftast tillhör den duktigare delen av programmerarkåren (Raymond nämner till och med en siffra på topp-5-procenten av alla utvecklare). Tidigare forskning (Hertel et al, 2003;

Dahlander, 2004) visar också att ett fåtal utvecklare står för de flesta modifieringarna av källkoden. Förvisso är denna forskning gjord endast på operativsystemet Linux och dess kärna, men Linux torde vara ett av de största ”Open Source”-projekten och därför i högsta grad relevant för ämnet.

2.3 … Och vad är inte FOSS

När vi nu har tittat på vad fri programvara och öppen källkod innebär kan det vara på sin plats att beskriva vad som inte är FOSS.

2.3.1 Katedralen

Det finns en uppsjö av teorier kring utvecklingsmodeller men vi fördjupar oss inte i detta utan nöjer oss med att kategorisera till Open Source eller inte. I det vi då kallar den traditionella utvecklingsprocessen inom mjukvaruindustrin, starkt förenklat utveckling enligt Raymonds katedralmodell, är det ett företag som utvecklar en produkt (underleverantörer kan användas) och därefter säljer den vidare till slutanvändaren, antingen själv eller via en eller flera distributörer (Raymond, 2001a).

Vi väljer att inte här i detalj beskriva de ekonomiska incitamenten för utvecklingsmodellen, detta ämne behandlar vi noggrannare i kapitlet om marknaden och FOSS. Kortfattat kan emellertid denna form av programutveckling sägas fungera på så sätt att en kärntrupp av utvecklare tar fram en programvara eller ny version av en programvara, testar den och gör sedan denna programvara eller version tillgänglig för en kundkrets. Denna kundkrets kan då köpa rätten att använda programvaran exempelvis genom att betala en licenskostnad.

För utvecklingen tas en kravspecifikation fram och en utvecklingstidplan sätts som det utvecklande företaget har fördelen att ha full kontroll över. Det är dock svårt att snabba upp processen utan att allokera extra resurser, exempelvis genom att ta folk från andra delar av organisationen eller hyra in konsulter (Raymond, 2001a).

Grundtanken är alltså att en investering i form av tid och ekonomisk ersättning till utvecklare leder till en färdig produkt eller mjukvara som på något sätt (genom exempelvis försäljning eller uthyrning eller liknande) kan generera intäkter som är tänkta att överstiga investeringskostnaden (Lerner & Tirole, 2000).

Ett problem eller en begränsande faktor för katedralen som utvecklingsmodell är vad Raymond (2001a) beskriver som ”Brooks’s Law”. Denna lag stipulerar att flera programmerare/utvecklare i ett projekt medför att komplexitet och kostnader för intern kommunikation i ett projekt ökar kvadratiskt medan vinsten i kortad programmeringstid som bäst endast ökar linjärt. Görling (2003a) hävdar å andra sidan att Brook’s lag även gäller för ”Open Source”-projekt, men att effekterna inte alltid visar sig eftersom många ”Open Source”-projekt är små eller organiserade på

ett sådant sätt att de negativa effekterna med att addera resurser kan minskas, till exempel genom en hierarkisk struktur.

2.4 Katedralen vs Basaren – viktiga skillnader

Katedralen och basaren är alltså två helt skilda sätt att se på programutveckling. Raymond (2001a) gör också vissa intressanta jämförelser mellan de två metoderna som förtjänar att uppmärksammas.

Förutom de mer uppenbara skillnaderna i utvecklingsmodellerna som beskrivits ovan jämför Raymond (2001a) även saker som projektledning och projektgenomförande mellan katedralen och basaren. Vid utveckling enligt katedralmodellen tas en kravspecifikation fram och en utvecklingstidplan sätts som det utvecklande företaget har fördelen att ha full kontroll över. Projekt utvecklade enligt basarmodellen däremot bygger på ett genuint personligt intresse hos både den utvecklande och användande parten (som ofta kan vara samma personer).

Ytterligare en väsentlig skillnad Raymond identifierar mellan katedral- och basar-modellen är att arbetet inom katedralen hålls slutet och resultatet (det körbara programmet) visas sent för användaren. Raymond (2001a) menar att det överskuggande målet är att användarna skall se så få buggar som möjligt. Kontrasten är basaren där buggar ses som ett ytligt fenomen och med flera som kan leta efter fel så kommer felen att upptäckas och åtgärdas tidigare, eller med Raymonds ord *"Given enough eyeballs, all bugs are shallow"* (Raymond, 2001a sid. 30). Det bör kanske påpekas att detta är ett mycket förenklat sätt att se på utveckling enligt katedralmodellen och beskriver endast ett sätt att utveckla mjukvara enligt denna modell. Utvecklingsmodeller för stängd mjukvara med hög grad av användarmedverkan kan presentera bilder eller prototyper av ett system under utveckling för användaren i ett tidigt stadium av utvecklingen.

2.5 För- och nackdelar med Katedralen och basaren

Om alla hade tyckt likadant hade världen tett sig ganska enformig och tråkig. Det finns naturligtvis en åsikt per individ i varje enskild fråga, men ett antal för- och nackdelar med dessa utvecklingsmodeller låter sig ändå påvisas vid ett försök att objektivt betrakta dem var för sig. För att användandet av någon av dessa modeller ska fungera i en kommersiell verklighet behövs även tilltalande marknadsekonomiskt motiverbara egenskaper, de djuplodande frågeställningarna kring dessa överlåter vi emellertid till att behandla i kapitlet som berör marknaden och FOSS.

Vi ser två olika perspektiv på mjukvaruutveckling, ett utvecklings- (för den enskilde utvecklaren) och ett användningsperspektiv (inbegriper även den som betalar för konsulttjänster/licenser och därmed projektgenomförande som helhet).

Här beskrivs i första hand för– och nackdelar sett ur ett användningsperspektiv. Utvecklingsperspektivet och olika belöningsformer för den enskilde utvecklaren tar vi upp i det ekonomiska kapitlet då vi ser utvecklaren som en del av marknaden och det ekonomiska system som omgärdar utvecklingen av mjukvara.

För att FOSS ska användas i en kapitalistisk värld måste det finnas incitament för företag och myndigheter att satsa på olika former av öppen källkod. Huruvida sådana incitament verkligen finns kan undersökas på olika sätt. Det går naturligtvis att undersöka företag och olika institutioner och inventera deras mjukvarubestånd för att se vad som används i dag. Ett annat sätt är att ta del av publikationer i ämnet som redovisar just denna typ av undersökning, vilket följer det metodval vi har valt att begagna oss av i denna uppsats.

Statskontoret har i sin rapport 2003:8 (Statskontoret, 2003a) undersökt förutsättningarna för användning av öppen källkod i den offentliga förvaltningen i Sverige. Denna rapport baseras bland annat på en rapport lämnad av det så kallade ”FLOSS”-projektet (Free/Libre Open Source Software), en utredning finansierad av IST (Information Society Technology) med syfte att tillgodose behovet av information kring öppen källkod (Statskontoret, 2003a; International Institute of Infonomics et al, 2002a).

”FLOSS”-projektet är ett stort projekt som har undersökt bland annat hur företag ser på öppen källkod och om det finns förutsättningar för företag och myndigheter att satsa på öppen källkod. Undersökningen kommer fram till att sådana förutsättningar finns, bland annat framkom att av 1452 undersökta företag och institutioner i Tyskland, Sverige och Storbritannien med minst 100 anställda använde eller planerade 395 av dessa att använda öppen källkod. Denna undersökning, som gjordes från februari till maj 2002, visar alltså att drygt vart fjärde företag av de undersökta är väl förtroget med Open Source (International Institute of Infonomics et al, 2002a).

Statskontorets rapport 2003:8 (Statskontoret, 2003a) redovisar vilka för- och nackdelar utredningen kan se med öppen källkod och den förväntade effekt Open Source kan få på den offentliga förvaltningen i Sverige.

I rapporten konstateras att man kan finna ett antal fördelar med öppen källkod, bland annat:

- *att produkterna är stabila*
- *hög säkerhet*
- *förenklad licenshantering*
- *möjlighet att modifiera källkoden*

- *bra tillgång till IT-specialister*
- *oberoendet av stora programvaruföretag*

Men det är inte så enkelt att det enbart finns fördelar. Ett antal negativa effekter identifieras också i rapporten:

- *kan kräva ett omfattande migrationsarbete*
- *kan leda till ökade krav på egen kompetens och underhåll inom myndigheten*
- *kan vara svårt att hitta rätt produkt*
- *man kan få interoperabilitetsproblem med proprietära program*
- *för närvarande mindre utbud på marknaden av konsult- och supporttjänster*
- *psykologiskt motstånd bland beslutsfattare*

Dessa för- och nackdelar som identifierats gör det möjligt att göra vissa intressanta iakttagelser. Utredningen visar att det finns fördelar med att använda produkter baserade på FOSS, utmaningen är för företag att eliminera eller åtminstone minimera nackdelarna. Även om utredningen på intet sätt kan sägas vara en bekräftelse på hur verkligheten ser ut ger den ändå indikationer på hur fri programvara och öppen källkod skulle kunna fungera i den svenska offentliga förvaltningen.

Utredningen tar dock inte upp vissa aspekter som exempelvis själva projektgenomförandet för att ta fram den önskade mjukvaran. Om företaget eller myndigheten beslutar sig för att använda FOSS kan ju modifieringar behöva göras av källkoden. Enligt Raymond(2001a) ses projektledningsbiten eller möjligheten att definiera klara mål och följa upp dem som en styrka av förespråkare för katedralens utvecklingsprocess. Detta bemöter basarens utvecklingsmodell och Raymond med att peka på att många utvecklingsprojekt (60-75 %) enligt katedralmodellen aldrig färdigställs eller förkastas av slutanvändaren efter mjukvarans färdigställande. Raymond menar att enda gången dessa projekt misslyckas och dör ut är när intresset svalnar hos de inblandade själva och visar på GNU emacs som ett talande exempel; projektet har haft otaliga medarbetare under 15 år och ändå lyckats hålla en klar utvecklingsmässig linje och en kontinuerlig utvecklingsfas under hela denna tid. Kort sagt menar alltså Raymond att basarprojekt har en bättre och mer motiverad projektgrupp än katedralprojekt och därför lyckas i högre utsträckning.

Det är dock svårt att ge en siffra på hur många projekt baserade på basarmodellen eller med FOSS-mjukvara som misslyckas då dessa dör ut bland annat just på grund av bristande intresse. Om endast ett fåtal personer satsat tid och/eller resurser på ett FOSS-projekt och det dör ut är det ju inte så många personer som uppmärksammar detta. Det är ju då mycket lättare att visa på de projekt som faktiskt lyckas. Siffran på

60-75 % misslyckade ”katedralprojekt” kanske därför skall tas med en nypa salt, åtminstone vid en jämförelse med FOSS-projekt.

Görling (2003a) undersökte mer än 34000 “Open Source”-projekt på SourceForge.net, som är en webbplats som fritt tillhandahåller resurser såsom lagringsutrymme, diskussionsforum och webbsidor för projekt som utvecklar program med fri och öppen källkod. En av slutsatserna i Görlings undersökning är att de flesta ”Open Source”-projekt bör ses som misslyckade*: “Project often dies at early stages, before releasing any stable products, without building a sustaining community of developers and users.”* (Görling, 2003a sid. 38).

Vad gäller de punkter utredningen kommer fram till som styrkor med mjukvara baserad på Open Source är stabilitet, säkerhet och ekonomiska besparingar nyckelord som stämmer väl överens med FOSS-förespråkarnas argument för sin sak. Det är ju också viktiga saker för företag likväl som för offentlig förvaltning.

Stabilitet och säkerhet har på senare år flitigt debatterats och kommit att bli ett argument i den alltmer hetsiga debatten mellan proprietär stängd programvara typiskt exemplifierar med Microsoft och företrädare för FOSS (kanske främst med syfte på Linux). Denna debatt väljer vi dock att inte ge oss in i utan nöjer oss med att konstatera att frågorna är viktiga för slutanvändare när de skall välja programvara.

Den ekonomiska aspekten är ytterst viktig då många investeringsbeslut baseras på investeringskalkyler. Vi ger oss inte heller i enlighet med vår avgränsning för denna uppsats in i några djupare resonemang kring vilka program som enligt sådana kalkyler blir ekonomiskt fördelaktiga.

De saker som identifieras som negativa av utredningen är dock viktiga att inte glömma bort, de tenderar annars att lätt drunkna i debatten mellan FOSS och proprietär programvara, ofta Microsoft (se exempelvis Wheeler, 2004a; OSI, 2004f). Här finns möjligheter för företag inom FOSS att hjälpa till att bistå andra företag och myndigheter vilket vi skall titta närmare på i kapitlet om marknaden och FOSS samt i vårt analyskapitel.

Rapport 2003:8 från Statskontoret (Statskontoret, 2003a) är ett exempel på den aktualitet ämnet öppen källkod har idag. Det blir emellertid ämne för en framtida studie att se vilka effekter denna rapport verkligen fått för användningen av öppen källkod inom den offentliga förvaltningen i Sverige.

De för- och nackdelar som redovisas ter sig emellertid intressanta inte bara för offentlig förvaltning, utan de torde högsta grad även vara aktuella för den privata sektorn. Även bortsett från grad av arbetsinsats och allt annat bortskalat inte minst eftersom ekonomisk hänsyn är en realitet för såväl den offentliga som privata sektorn.

2.6 Andra synsätt på fri programvara och öppen källkod

Vi har tidigare beskrivit Raymonds syn på Open Source och dess utvecklingsmodell. Det finns naturligtvis olika synsätt, även om Raymond är bland de mest framträdande figurerna i diskussionen kring företeelsen Open Source.

I grund och botten är dock de flesta som diskuterar fri programvara och öppen källkod överens om vad företeelsen handlar om: Att skapa högkvalitativ mjukvara i ett nätverk av gemensamma utvecklare och användare där vem som helst är fri att både använda och förändra källkoden, så länge man delar med sig till nätverket av sina förbättringar (Raymond, 2001a; SUN, 2004a).

Men det finns även de som ser faror med Open Source. Exempelvis Jones (2004) ser en fara med ”Open Source”-mjukvarans växande popularitet baserat på att med så många utvecklare/användare torde risken för att någon använder en produkt för att medvetet få in dålig kod/ kod med undermålig säkerhet i produkten öka. Det skulle därefter vara möjligt att använda produkten för andra syften än de tänkta. För företag och myndigheter som använder ”Open Source”-programvara borde detta vara en varningssignal enligt Jones (2004).

2.6.1 Vidare utveckling för FOSS

Raymond (2001a) refererar till undersökningar där det sägs att bonus till programmerare tar bort kreativiteten hos dem, det är alltså bättre att hålla lön och prestation åtskilda och i stället låta programmerarna välja sina projekt för att uppnå maximal prestationsnivå. Detta i sin tur leder fram till en av Raymond framförd hypotes att ”industriell” mjukvarutillverkning gick mot sitt öde i det ögonblick kapitalismen ledde till att programmerare tjänat tillräckligt för att kunna ägna mer av sin fria tid till att syssla med det de helst vill göra i stället för det de tvingas till av sitt arbete. Med andra ord, låt programmerare göra det de vill och glöm allt vad tidplaner heter för att få den bästa koden utvecklad! Raymonds poäng här är att detta var just den metod som användes av ”Open Source”-rörelsen när den inleddes.

När tillräckligt många använder ”Open Source”-metoden för att utveckla program skriver Raymond (2001a) vidare att problem kan uppstå med att integrera den växande massan av individer med existerande projekt och grupper. Han anknyter därför till människans historia där ökade befolkningsgrupper ledde till gemensamma lagar som först fördes vidare från stamledare till stamledare och sedan genom historien kom att skrivas ned. Alltså kan vi behöva definiera regler för lösningar på de problem som kan uppstå vid utveckling av ”Open Source”-projekt. Raymond har själv påbörjat ett sådant regelverk, ”the Malvern Protocol” (Raymond, 2001a) .

Ett ytterligare problem som bland annat Raymond (2001a) samt Lerner och Tirole (2000) noterar är risken för att utvecklingen av ett program enligt ”Open Source”-metoden skall delas upp i olika sidospår. Detta kallas på engelska för ”forking” (från engelska ordet för gaffel) och är en inom branschen vedertagen term (även i Sverige)

för detta fenomen, varför vi i fortsättningen även använder oss av denna term. Denna ”forking” innebär alltså att olika varianter av samma program kan uppstå och kan bero på till exempel dispyter kring designen av programmet mellan ledande utvecklare (Lerner & Tirole, 2000, exemplifierat med Berkeley Unix). Raymond menar att detta fenomen är ganska sällsynt och hänvisar till att det finns ett starkt sociologiskt grupptryck inom ”Open Source”-allmänheten att inte bidra till att flera parallella versioner av ett program sprids.

Vidare gör Lerner och Tirole (2000) en jämförelse mellan vad man kallar ”traditionell” mjukvaruutveckling och ”Open Source”-utveckling och konstaterar då att ”Open Source”-utveckling ofta är riktad mot den avancerade skaran av utvecklare (jämför med exempelvis Görling, 2003a, som också noterar att den största delen av utvecklingen görs av sofistikerade användare) medan exempelvis Microsoft (som får representera ”traditionell” utveckling) i stället ofta riktar sina produkter i huvudsak till de minst avancerade användarna. Lerner och Tirole menar att detta kan vara en orsak till att användare av FOSS ofta i högre utsträckning accepterar brister i exempelvis dokumentation, användarvänlighet eller bakåtkompabilitet än användare av programvara utvecklad på traditionellt sätt (enligt katedralmodellen).

2.6.2 Från Closed Source till Shared Source

Trots vår ovilja att hamna i debatten mellan ”Open Source”-rörelsen och Microsoft anser vi dock oss behöva i korta drag beskriva Microsofts inställning till fri och öppen källkod eftersom företaget har en relativt övriga konkurrenter dominerande ställning på marknaden för mjukvara för persondatorer. Traditionellt har ett företag som Microsoft arbetat efter ”katedralprincipen” enligt beskrivning ovan. Denna modell kallas av Microsoft för CSD, ”Commercial Software Development” (Microsoft, 2004a). I takt med de senaste årens framgångar för mjukvara utvecklad enligt ”basarprincipen” (av Microsoft kallad OSS, ”Open Source Software”) har Microsoft dock valt att närma sig denna inriktning.

Microsofts sätt att bemöta och ta vara på fördelarna med ”OpenSource”-utveckling är det som kallas för SSI, ”Shared Source Initiative”. Detta är ett sätt att ge utvecklare tillgång till källkod från Microsoft och kunna göra modifieringar i och utveckling av koden. Genom licensvillkoren (där speciella licensvillkor finns för exempelvis studerande samt utbildningsinstitutioner som universitet och högskolor) har den som modifierar koden dock inte rätt att distribuera den, utan ägandeskapet av och rätten till källkoden ligger fortfarande hos Microsoft. Detta ligger i linje med Microsofts uttalande om att ”närma sig mitten” (Microsoft, 2004a). Med detta menas en önskan att ta vara på fördelarna med ”OpenSource”-utveckling (att ha fler utvecklare som testar och rättar koden) samtidigt som licensintäkter och konkurrensfördelar med att äga källkoden kan bevaras, helt enkelt förena det bästa av två världar.

2.7 Sammanfattning

I detta kapitel har vi försökt ge en introduktion till vad fri programvara och öppen källkod (FOSS) är och innebär.

Det har även försökts påvisas att FOSS inte alls är någon ny företeelse. Hårdvaruleverantörer såsom Digital Equipment Corporation (DEC) tog initialt inte separat betalt för mjukvaran då de levererade sina system. En möjlig anledning till DEC:s beteende är att de program som DEC skickade med sina system enbart kunde köras på DEC:s maskiner så någon risk för utbredd ”piratkopiering” torde inte direkt föreligga. DEC:s kärnverksamhet och värld var genomsyrad av hårdvara och tanken att man skulle tjäna stora pengar på mjukvara var vid tidpunkten inte speciellt utvecklad (Rosenberg, 2000). Likväl kan det konstateras att tanken att tjäna pengar på program och förekomsten av rena mjukvaruföretag var ett senare fenomen och att system- och hårdvaruleverantörerna fick se sig omsprungna av företag såsom Microsoft (Lunell, 1991; Rosenberg, 2000).

Vi har även visat på några skillnader mellan Free Software och Open Source. Vidare har vi beskrivit motsatsen till FOSS och även försökt förklara utvecklingsmodellen för Open Source och Free Software för läsaren. Utvecklingsmodellen för FOSS, basaren, har ställts i kontrast till katedralen som är det namn som Raymond (2001a) ger för utvecklingsmodellen för proprietär stängd programvara. Synsättet med en strikt uppdelning i basarmodellen och katedralmodellen är en förenkling och det kan mycket väl finnas organisationer som blandar element ur de båda.

I viss utsträckning har vi försökt ställa Open Source och dess motsats mot varandra och med hjälp av litteratur i ämnet försökt visa hur de förhåller sig till varandra.

3 FOSS-rörelsen

Vi vill i detta kapitel ge bakgrunden till och kortfattat beskriva några tongivande rörelser kring fri programvara och öppen källkod.

3.1 Inledning

Som tidigare har visats så fanns fenomenet med fri programvara och öppen källkod redan i datorernas barndom då forskare delade med sig av programvara och så att säga hjälptes åt. När datorprogram mer och mer började användas kommersiellt blev många programmerare på universitet anställda och de ”…befann sig i en situation där de som tidigare varit kollegor plötsligt blev konkurrenter. Som anställda fick de inte längre dela med sig av sina framsteg…” (Olofsson, 2003a sid. 11-12). Så småningom kom en motreaktion och mer organiserade rörelser pro FOSS växte fram.

Det finns ett otal personer världen över som bidragit till framgångarna och framväxten av fri programvara och tanken om öppen källkod. Det är ju en av poängerna med tanken i sig, att människor världen över tillsammans skall hjälpas åt för att generera mjukvara med hög kvalitet. Det finns dock några namn som skiljer sig ur mängden genom att de exempelvis lagt grunden till vissa företeelser eller organisationer eller på annat sätt bidragit lite extra till att sprida idéer eller tankar kring Free Software och Open Source Software (FOSS). Vi kommer här att ge en översikt över några av dessa personer och de organisationer som de närmast förknippas med.

3.2 Free Software Foundation

Vi inleder med organisationen “Free Software Foundation”, en organisation som startades i mitten av 1980-talet.

3.2.1 Organisationen

Stiftelsen ”Free Software Foundation” (i fortsättningen FSF) är en organisation som förespråkar fri programvara, främst i meningen fri att använda och modifiera, men också fri från licenskostnader eller andra avgifter. Denna organisation har sitt

ursprung under 80-talet med Richard Stallmans protest mot att han inte fick tillgång till källkoden till den mjukvara han använde på universitetet Massachusetts Institute of Technology (MIT) där han var verksam. På MIT fanns ett utvecklingscenter för programmerare, "Artificial Intelligence lab". Utvecklingen av programvara var fri och drevs där av att alla delade med sig av de program de skrev och man använde inga lösenord utan alla var fria att ändra och förbättra varandras programkod (GNU, 2004f; Moody, 2001). En del av Stallmans kollegor blev anställda på eller startade egna kommersiella företag där de inte längre delade med sig av kod utan istället konkurrerade med sluten kod (Olofsson, 2003a). Stallman kunde inte acceptera detta utan sa helt sonika upp sig och startade arbetet med att ta fram ett manifest och en licens med fri programvara som ideologisk kärnpunkt. Eftersom UNIX inte längre distribuerade fritt med källkod påbörjade Stallman sin mission med att ta fram ett nytt fritt UNIX-liknande operativsystem (Moody, 2001). Detta arbete ledde så småningom fram till licensvillkoret GNU General Public License som vanligen är känd under förkortningen GNU GPL eller bara GPL (Williams, 2002; GNU, 2004f).

3.2.2 Företrädare för Free Software Foundation

Det finns många personer som aktivt arbetat för FSF, vi beskriver här de mest framträdande.

3.2.2.1 Richard M. Stallman

Richard M Stallman har en akademisk examen i fysik från Harvard, examensår 1974. Han refererar gärna till sig själv med sina initialer "RMS". Stallman arbetade redan under sina tidiga studieår vid ovan nämnda MIT artificiell intelligenslaboratorium där han lärde sig att skriva kod för operativsystem genom att göra det i praktiken. Vid MIT skrev han också den första Emacs-texteditorn 1975.

Richard Stallman avslutade sin tjänst vid MIT 1984 när han grundade det så kallade "GNU-projektet" för att utveckla det fria operativsystemet GNU (som står för "GNU's Not Unix", vilket är ett rekursivt akronym). GNU är fri mjukvara i betydelsen att vem som helst fritt får kopiera och/eller ändra i koden.

Utöver Emacs har Stallman också bland annat varit en av huvudmännen bakom kompilatorn GCC (GNU Compiler Collection), en kompilator som designats för att fungera i olika miljöer och med olika programspråk (Stallman, 2004a; Williams, 2002).

3.2.2.2 Bradley M Kuhn

Bradley M Kuhn har en akademisk examen i datavetenskap från universitetet i Cincinatti. Han inledde sitt arbete med FSF under mitten av 90-talet som frivillig mjukvaruutvecklare till GNU-projektet. År 2001 anställdes han på heltid av FSF och är idag dess verkställande chef. Vid sidan av detta arbete bidrar han fortfarande till utveckling av mjukvara inom ramen för GNU-projektet (FSF, 2004b).

3.3 Open Source Initiative

Låt oss så göra en påhälsning hos organisationen Open Source Initiative (OSI).

3.3.1 Organisationen

Organisationen ”Open Source Initiative” (eller OSI som namnet brukar förkortas) är ett icke-vinstdrivande företag som arbetar för att generellt främja användningen av Open Source och specifikt göra Open Source tilltalande i kommersiella sammanhang (OSI, 2004e). OSI bildades som ett resultat av ett möte mellan företrädare för fri programvara som kom till direkt efter det att Netscape i februari 1998 hade meddelat att de ämnade släppa källkoden till sin webbläsare Netscape Navigator fri (Olofsson, 2003a; OSI 2004a).

OSI har tagit fram kriterier som programvara måste uppfylla för att få räknas som Open Source, kallade ”Open Source Definition” (i fortsättningen OSD). Dessa kriterier omfattar i huvudsak rätten att fritt använda, distribuera och ändra i programkod (OSI, 2004b). I de regler som utgör OSD ingår, till skillnad för de licenser som FSF förespråkar och tillhandahåller, inget krav på att rättigheterna, exempelvis tillgång till källkod, tvunget förs över till mottagaren av ett program när det distribueras vidare. OSI tycks mena att ett sådant krav inte alltid passar i kommersiella sammanhang och att Free Software därför kan verka avskräckande för mjukvaruutvecklande företag. Dock finns det inget i OSD som motsäger exempelvis GNU GPL utan tvärtom så finns GNU GPL med på listan över godkända och rekommenderade ”Open Source”-licenser.

3.3.2 Personer

Inom sfären kring Open Source Initative återfinner vi en av dagens massmediala förgrundsfigurer för debatten om öppen källkod, Eric S Raymond, men även exempelvis mannen bakom operativsystemet Linux, Linus Torvalds.

3.3.2.1 Eric S.Raymond

Eric Steven Raymond är en av förgrundsgestalterna inom ”Open Source”-rörelsen och precis som Stallman så refererar även Raymond ibland till sig själv med sina initialer, det vill säga "ESR". Han är en av grundarna till organisationen ”Open Source Initiative” och är även "Open Source"-rörelsens huvudsakliga ansikte utåt mot medier och allmänheten. Eric ligger bakom flera "Open Source"-projekt och har själv ansvarat för utvecklingen av programmet ”Fetchmail”, ett program för att hämta e-post över Internet. Raymond har en bred erfarenhet av programmering och programutveckling på olika nivåer och är idag en aktiv talare och föredragshållare (Raymond, 2001a; Williams, 2002).

3.3.2.2 Linus Torvalds

Linus Torvalds är också ett av de stora namnen inom Open Source. Han har skapat och gett namn åt operativsystemet Linux och var drivkraften bakom dess utbredning och succé. Linus har studerat vid Helsingfors Universitet och tagit examen i datavetenskap där. Han bor och arbetar i USA (Williams, 2002; Moody, 2001).

3.3.2.3 Bruce Perens

Bruce Perens är en av frontgestalterna inom OSI och skapade förlagan till Open Source Definition (OSD) med sitt dokument ”The Debian Free Software Guidelines”.

Med hjälp av kommentarer från Debian-utvecklare förfinades dokumentet och så småningom tog Perens bort alla referenser till Debian i dokumentet och skapade på så sätt OSD.

Debian var ett projekt för att utveckla användarvänligheten och programutbudet för användare av Linux och finansierades till en början av FSF (Debian, 2004a; Williams, 2002; Raymond, 2001a).

3.4 Open Source Initiative/Free Software Foundation

Det finns alltså flera organisationer som kämpar för öppen källkod. Vi har tittat närmare på de två viktigaste, FSF och OSI. Dessa organisationer kan tyckas vara lika, men det finns vissa skillnader. OSI tillåter exempelvis vissa former av licensiering som inte FSF gör. Vi skall studera förhållandet mellan dessa två organisationer lite närmare.

Skillnaden mellan Open Source och Free Software kan kortfattat sägas vara att Open Source kan acceptera att fri, öppen kod används i programvaror med sluten kod medan FSF aldrig accepterar sluten källkod. Skillnaderna kan ibland tyckas vara hårfina, men blir märkbara när man tittar på några praktiska exempel, något som vi ämnar göra i kommande kapitel. Det gemensamma mellan rörelserna är givetvis att båda strävar efter öppen källkod, skillnaden ligger alltså i hur långt man kan sträcka sig för att uppnå detta mål. Både OSI och FSF vänder sig dock mot användandet av termen ”fri” mjukvara som något man inte behöver betala för. Fri skall i stället ses som i betydelsen fri spridning av information och frihet för individen eller yttrandefrihet om man så vill. Ett exempel för att visa på felaktigheten i den vanliga missuppfattningen att fri mjukvara är lika med gratis mjukvara är Microsofts webbläsare Internet Explorer. Detta program är numera gratis, men källkoden är absolut inte fri (Wheeler, 2004b).

De båda rörelserna är inte fientligt inställda till varandra men har likväl en särpräglad hållning i förhållande till varandra något som framgår av såväl GNU (2004e) som OSI (2004c).

3.5 Sammanfattning

I detta avsnitt har de dominerande rörelserna för fri programvara och öppen källkod det vill säga organisationerna Free Software Foundation (FSF) och Open Source Initiative (OSI) presenterats. Bekantskap har även stiftats med några tongivande förespråkare från dessa båda läger. De grundläggande tankarna och vurmandet för öppen källkod delas mellan organisationerna. De skillnader som framträder rör i mångt och mycket inställning och ideologi. OSI är något mer pragmatiskt lagda och är uttalat inriktade på att sälja in konceptet Open Source till den kommersiella världen medan FSF främst drivs av sin ideologiska övertygelse och siktar på att ta fram alternativ så att all typ av programvara ska finnas tillgänglig som öppen källkod eller snarare vad de kallar ”Free or Semi-Free software”, något som vi får anledning att återkomma till i senare kapitel (FSF, 2004c).

4 Kategorier av programvara

–"The only thing the copyright forbids (and I feel this is eminently reasonable) is that other people start making money off it, and don't make source available etc... This may not be a question of logic, but I'd feel very bad if someone could just sell my work for money, when I made it available expressly so that people could play around with a personal project. I think most people see my point."- Inlägg av Linus Benedict Torvalds, den 6:e februari 1992 i nyhetsgruppen comp.os.minix (citerad i Perens, 1999)

I detta kapitel kommer vi att undersöka olika kategorier av mjukvara eller närmare bestämt olika typer av licenser för mjukvara.

I citatet ovan kommenterar Linus Torvalds, skaparen av Linux, de regler som han ställde upp för de allra första versionerna av Linux som han delade med sig av. Copyrighten ovan är faktiskt avsevärt mer restriktiv och om man så vill ”marknadsfientlig” än GNU GPL (GNU General Public License) som idag kanske är den mest kända typen av ”Open Source”-licens och som från version 0.12 även är den licens under vilken Linux distribueras.

Innan vi går vidare och stiftar närmare bekantskap med GNU GPL och andra typer av licenser kan det vara på sin plats att gå igenom och definiera några grundläggande termer.

4.1 Fundamentala begrepp

Mjukvara omfattas av immaterialrätt, eller Intellectual Property Rights (IPR) för att ta den engelska termen. Immaterialrätt handlar om (ägande)rätten till saker som det inte direkt går att ta på, det vill säga saker som saknar fysisk form eller annorlunda uttryckt immateriella ting. Några exempel på immaterialrättsområden är patent, copyright, varumärken och affärshemligheter. (Microsoft, 2003a; IPR-Helpdesk, 2004a). I Sverige har datorprogram lagstadgat skydd som litterärt verk enligt första paragrafen i upphovsrättslagen (Olofsson, 2003a).

Den grundläggande tanken med immateriella rättigheter är att skydda uppfinningar, namn, kreativitet och uppfinningsrikedom. Europeiska kommissionen (2004) delar upp immaterialrätten i två huvudgrenar: industriell egendom och upphovsrätt. Industriell egendom rör främst skydd av uppfinningar (patent), varumärken och industridesign. Europeiska kommissionen (2004) menar att den andra delen *”brukar traditionellt beskrivas som de rättigheter som gäller litterära och konstnärliga verk, det vill säga*

nyskapande verk inom litteratur och konst". De områden som omfattas av dessa rättigheter är emellertid större än så och inkluderar *"tryckta medier, konst, musik, ljudinspelningar och filmer men också etermedier, datorprogram, databaser och andra former av multimedieverk."* (Europeiska kommissionen, 2004). Mjukvara hamnar således med denna indelning under upphovsrätten. Det finns även fall då mjukvara faller under patentlagstiftningen och detta är ett ämne för kontroverser (Tysver, 2000; FFII, 2004a). Patentlicenser kan enligt Olofsson (2003a) medföra problem speciellt för FOSS. Frågeställningarna kring mjukvara och dess patenterbarhet ligger dock utanför denna uppsats skop.

För att få nyttja ett program behöver användaren ha tillstånd för det eller annorlunda uttryckt ha en korrekt licens för det. Bonnier (1996) förklarar att ordet licens kan spåras till latinets *licentia* som närmast betyder just tillåtelse och fortsätter sedan att definiera licens som: *"tillstånd att (mot ersättning) utnyttja ett patent, mönster, varumärke etc."* (Bonnier, 1996 sid. 220). Det förekommer också att upphovsrätten helt överlåts till exempel om ett företag utvecklar ett system specifikt för mottagaren.

Eftersom en licens är slags avtal kan det finnas oklarheter huruvida motparten (användaren) fått eller rimligen kunnat skaffa sig kännedom om villkoren i avtalet för att avtalet ska betraktas ha trätt i kraft. Detta problem är dock inte specifikt för FOSS, men då fri programvara och öppen källkod nästan alltid enkelt finns att ladda ner från olika källor på Internet blir problematiken inte mindre. Med tanke på att fri programvara och öppen källkod är transnationell, finns likaledes frågor gällande lagval, exempelvis i vilken domstol eventuella tvister ska lösas, samt frågetecken gällande olika licensers förenlighet med olika länders lagstiftning (Olofsson, 2003a). Vidare undersökningar och resonemang kring dessa frågeområden lämnas utanför uppsatsens omfång.

Vi kommer inom kort att bekanta oss med ett antal licenser för fri programvara och öppen källkod i avsnitt 4.3 (Licensieringsmodeller för FOSS). Låt oss dock börja med att försöka bena ut vilka olika kategorier av mjukvara som existerar.

4.2 Kategorier av mjukvara

Det finns många tänkbara sätt att kategorisera mjukvara. Vi har valt att utgå från en klassificering gjord av Free Software Foundation (FSF) i "Categories of Free and Non-Free Software" (GNU, 2004a) samt från ett antal kriterier fastställda i en definition av Open Source gjord av Open Source Initiative kallad "The Open Source Definition" (OSI, 2004b).

När vi ska jämföra olika mjukvarulicenser har vi valt att beakta följande aspekter baserat på GNU (2004a) och OSI (2004b):

- Rätt till vidaredistribution

  - Får användaren distribuera programmet vidare utan kostnad eller inskränkningar i exempelvis antal kopior?

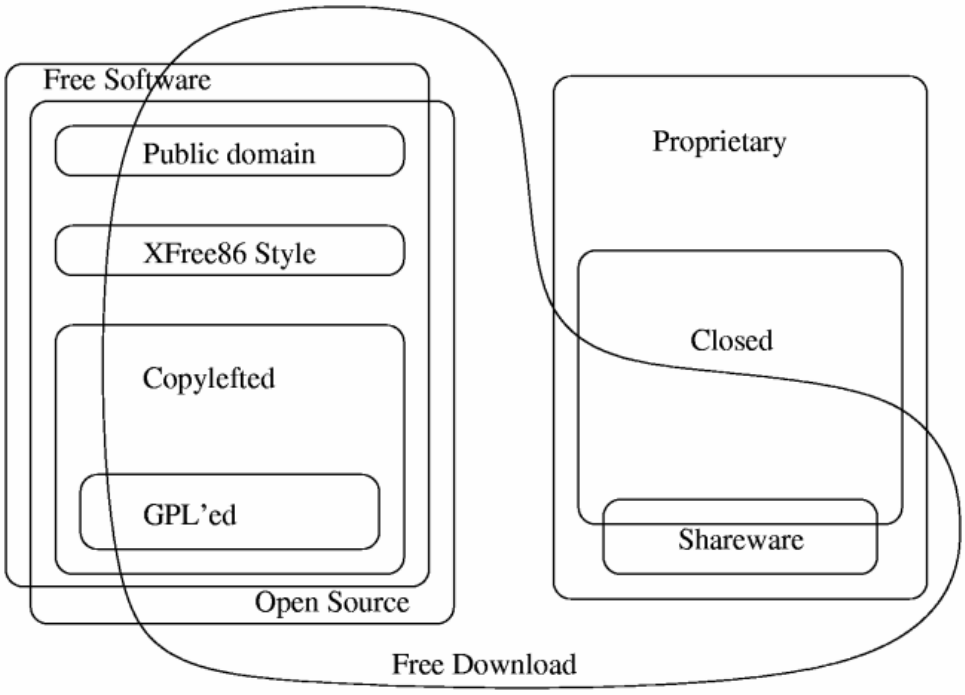
- Tillgång till källkod
  - Finns källkoden tillgänglig utan begränsningar?

- Rätt till modifikationer
  - Är det tillåtet att modifiera programmet och sedan vidaredistribuera modifikationen?

- Begränsningar
  - Görs någon diskriminering av personer, grupper eller användningsområden? Finns rätten att använda ett program för kommersiellt bruk?

- Åtkomst
  - Är programmet fritt att införskaffa, exempelvis genom att det går att ladda ner från upphovsmannen via Internet?

För att beskriva olika kategorier av mjukvara har vi valt att utgå ifrån en figur skapad av Chao-Kuei i GNU (2004a). Bilden är reproducerad i figur 4.1 nedan.



Figur 4.1: *Kategorier av Fri- och ickefri mjukvara (Från "Categories of Free and Non-Free Software",* GNU, 2004a*).*

Figur 4.1 visar bland annat att Free Software och Open Source överlappar varandra. Vidare kan vi utläsa att fri nerladdning eller gratis inte alls är entydigt med öppen källkod utan att även olika typer av stängd eller proprietär mjukvara kan vara gratis att hämta hem. Den huvudsakliga skiljelinjen går mellan fri och olika former av icke-fri mjukvara. Vi kommer att börja studera olika typer av fri mjukvara för att och senare bekanta oss med kategorier av icke-fri mjukvara.

Det bör noteras att en upphovsman kan välja att ge ut ett program under flera, i sig inkompatibla, licenser. Låt oss gå igenom de olika kategorierna ovan och kortfattat definiera vad de innebär. Varje kategori exemplifieras med konkreta exempel på programvara. Eftersom kategorierna i figur 4.1 överlappar så kommer vi återge samma exempel på flera ställen. Vi har i huvudsak valt att behålla de engelska termerna i modellen ovan istället för att försöka uppfinna nya svenska termer.

4.2.1 Fri mjukvara

Nedan undersöker vi olika kategorier av fri mjukvara.

4.2.1.1 Free Software

Andemeningen med ordet ”Fri” i Free Software (Fri mjukvara) är ”frihet” snarare än ”gratis”. För att ett program ska vara Free Software ska var och en ges rätten att använda, kopiera, distribuerar orginalprogrammet och modifikationer därav. För att rättigheterna ovan ska kunna uppfyllas måste källkoden vara fritt tillgänglig för att ett program ska kunna klassificeras som Free Software. Free Software förespråkas främst av Free Software Foundation (FSF) och GNU-projektet. Notera att Free Software inte ska blandas ihop med Freeware (se nedan).

Exempel:

- GPL-mjukvara såsom:
  - Linux (http://www.kernel.org)
  - GCC (http://gcc.gnu.org/)

4.2.1.2 Open Source

Kategorin Open Source är närbesläktat med Free Software och Free Software kan ses som en delmängd av Open Source. Detta innebär att Open Source är ett lite mindre strikt begrepp även som omfattar licenser som inte är Free Software enligt FSF:s synsätt (GNU, 2004a). Notera att Figur 4.1 inte tydligt återspeglar detta förhållande.

Open Source kan också karaktäriseras som mer kommersiell- eller företagsvänlig. Termen Open Source gjordes populär av bland annat Eric Raymond och Bruce Perens när de startade organisationen Open Source Initiative OSI (Slackware, 2004a; OSI, 2004a).

Exempel:

- MPL (Mozilla Public License) – (Mozilla, 2004a)
- EPL (Eclipse Public License) – (Eclipse Foundation, 2004a)

4.2.1.3 Public Domain

Copyright används som ett instrument för att skydda upphovsmannens rätt till sina verk. Om ett verk är i Public Domain har verket ingen känd upphovsman eller så har

dess upphovsman helt enkelt avsagt sig rätten till sitt verk och låter allmänheten använda och reproducera verket utan vare sig kostnad eller begränsningar (Olofsson, 2003a).

Programvara som kategoriseras som Public Domain är inte copyright-skyddad, det vill säga dess upphovsman donerar programmet till allmänheten eller så saknar programmet upphovsrättsägare (Williams, 2002). Ingen äger rätten till ett arbete, exempelvis mjukvara, som är Public Domain och ”om ett arbete verkligen är i allmänhetens domäner, så finns det ingen med äganderätt till arbetet”, vår översättning (UCAR, 2004). Detta ger upphov till en prekär situation då vilken person som helst kan ta ett program som är Public Domain och distribuera det vidare under sin egen copyright och därigenom ta kontroll över sina kopior av programmet. Låt oss ta ett exempel som inte rör mjukvara, men vars principer kan översättas rakt av även till mjukvaruvärlden: *“Modifications to a public domain work may be protected by copyright and cannot be used without permission. A famous example used in many copyright classes is of the artist who paints an elaborate hat and mustache on the Mona Lisa. Even though anyone is free to copy the Mona Lisa image, the modified image (with mustache and hat) is protected under the artist 's copyright.”* (Stanford, 2003a).

Public Domain var ganska vanligt förekommande förr för applikationer för person- och hemdatorer. Ett program som är i Public Domain kan ha antingen källkod, binärkod eller båda i Public Domain.

Exempel:

- Fred Fisk Disk collection (Lysator, 2004a)

4.2.1.4 Copylefted Software

Copylefted Software är en lek med ordet copyright. Copyleften bygger vidare på copyright och fungerar som en garant för rättigheterna för Free Software så att ingen part ska kunna inskränka dem. Organisationen Free Software Foundation beskriver begreppet Copyleft: *”Copyleft is a general method for making a program free software and requiring all modified and extended versions of the program to be free software as well.”* (GNU, 2004b). Vi får anledning att återkomma med mer information om Copyleft i samband med avsnittet om GPL senare i detta kapitel.

Exempel:

- Linux och annan GPL-mjukvara

4.2.1.5 GPL-covered software

GPL-covered software är helt enkelt programvara som täcks av GPL-licensen. GPL är i sin tur ett exempel på Copylefted Software (se ovan).

Exempel:

- Linux (http://www.kernel.org/)
- Emacs (http://www.gnu.org/software/emacs/emacs.html)

4.2.1.6 Non-copylefted free software

Med kategorin Non-copylefted Free Software (vilken visas som ”Xfree86 Style” i Figur 4.1 ovan) avses program där upphovsmannen ger rätt att distribuera programmet vidare, rätt att modifiera programmet och rätt att lägga till nya begränsningar på kopior av programmet. Detta innebär exempelvis att modifikationer av programmet inte behöver vara öppna, varken avseende tillgång till källkoden eller det körbara programmet. Notera dock att licenserna inte på något vis hindrar ändringar som görs släpps öppna. Det är bara så att den tvingande regeln som återfinns för copyleftade program inte finns.

Exempel:

- X Windows (http://www.x.org)
- Apache Server (http://httpd.apache.org/)

4.2.2 Proprietär mjukvara

Proprietär mjukvara kommer i olika former. Notera att begreppet proprietär ursprungligen betyder att något har en ägare eller innehavare. I datorsammanhang brukar termen proprietär, eller ”proprietary” på engelska, användas för *”programvara som ägs och kontrolleras av rättighetsinnehavaren och ställs i motsatsförhållande till fri programvara”* (Olofsson, 2003a sid 122). Vi väljer att fortsatta använda order proprietär som något som står i motsats FOSS. Nedan låter vi fästa blicken vid ett antal kategorier av proprietär programvara.

4.2.2.1 Proprietary software

Proprietary software (proprietär mjukvara) är program som varken kan kategoriseras som Free Software eller Semi-free software. Användning, vidaredistribution samt modifiering är begränsad och eller helt förbjuden. Även installation av proprietär programvara kan vara begränsad exempelvis som i Microsoft Office EULA (End User License Agreement) (Microsoft, 2004b).

Exempel:

- Windows (http://www.microsoft.com/windows/default.mspx)
- Microsoft Office (http://office.microsoft.com/en-us/default.aspx)

4.2.2.2 Shareware

Ett program som är Shareware kommer med rätten att distribuera det vidare till andra. Shareware är gratis att ladda ner och testa för en begränsad period, men efter en viss tid, typiskt ett par veckor, ombeds användaren att betala en summa till upphovsmannen eller införskaffa en permanent licens exempelvis så som utvärderingslicensen för programmet Winzip stipulerar (Winzip, 2004a). Shareware är varken Free Software eller Semi-free Software och oftast finns källkoden till ett Sharewareprogram ej fritt tillgängligt.

Exempel:

- UltraEdit (http://www.ultraedit.com/downloads/index.html)

4.2.3 Fler kategorier

Vår taxonomi har flera kategorier som inte rakt av kan återfinnas i Figur 4.1 ovan men vilka likväl är av intresse och som också omnämns i GNU (2004a).

4.2.3.1 Semi-free software

Med semi-free software avses program som ej är Free Software, men där programmet ändå kommer med rätten för individer att använda, kopiera, distribuera och modifiera. Rättigheterna ovan är för den här kategorin av program dock begränsade i att de exempelvis inte får användas i vinstdrivande sammanhang eller ibland uttryck som att programmet enbart får användas för privat bruk.

Exempel:

- PGP (http://www.pgpi.org/)

4.2.3.2 Freeware

Begreppet Freeware har ibland lite olika mening men oftast är program som kallas Freeware gratis att ladda ner och fritt att använda, men oftast är källkoden till programmet inte fritt tillgänglig. Ibland är det inte fritt att distribuera programmet vidare. Kategorin är lite problematisk och termen används ibland helt felaktigt synonymt med Free Software.

Exempel:

- Context Editor (http://www.context.cx/features.html)

4.2.3.3 Kommersiell mjukvara

Kommersiell mjukvara är mjukvara som utvecklas av en affärsdrivande enhet med syfte att tjäna pengar på programvaran. Oftast innebär kommersiell programvara ”Proprietary”, men det finns även kommersiell fri programvara.

Exempel:

- Microsoft SQL Server, är en kommersiell proprietär databashanterare (http://www.microsoft.com/sql/default.asp)

- MySQL, är en kommersiell fri databashanterare (http://www.mysql.com/)

4.3 Licensieringsmodeller för FOSS

Vi har bekantat oss med ett antal kategorier av mjukvara. Låt oss nu närmare undersöka ett antal licensieringsmodeller specifika för fri programvara och öppen källkod.

4.3.1 GNU GPL/LGPL - Free Software

GNU GPL (GNU General Public License) är en licensieringsmodell för så kallad Free Software. Ordet ”Free” i ”Free Software” syftar till fyra friheter som vi har valt att återge nedan direkt från källan utan försök till översättning (GNU, 2004c):

- *“The freedom to run the program, for any purpose (freedom 0).*
- *The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.*
- *The freedom to redistribute copies so you can help your neighbor (freedom 2).*
- *The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. (freedom 3). Access to the source code is a precondition for this.“*

En sidnotering som kan låta sig göras angående ovanstående lista är att numreringen av ”friheterna” börjar på noll och inte ett. Förklaringen till detta är med i det närmaste visshet att det är människor med bakgrund som programmerare och mer specifikt C- och C++ programmerare som har inspirerat till och tagit fram kriterierna ovan, mest noterbar är kanske Stallman (Williams, 2002).

GNU GPL har tagits fram av Free Software Foundation (FSF). Det är värt att notera att Free Software inte syftar till att gratis (som i ”gratis öl” för att ta ett ofta citerat exempel). ”Free” syftar snarare på fri som i ”frihet”. Friheterna som GPL medför, sammantaget med att GPL är avsett som ett (legalt) bindande licensieringsavtal, kan ses som en intrikat kullerbytta där de normalt inskränkande licensierings- och copyrightlagarna vänds ut-och-in på. GNU GPL tvingar programvaran att vara fri, men hindrar samtidigt en enskild part från att kunna monopolisera eller stänga inne mjukvaran. Eftersom GPL kan sägas vända på copyrighten brukar den kallas för copyleft. Ordet copyleft är en lek med ord som inte kan översättas. Perens (1999) skriver att Stallman valde ordet copyleft eftersom: *”it leaves the right to copy left”*.

GPL innebär att all programvara som baseras på GPL-kod själv kommer att omfattas av GPL. Exempel på programvara som distribueras under GPL-licens:

- GNU/Linux (http://www.linux.org, http://www.debian.org)

Att GNU GPL är smittsam (eng. ”viral”) till sin natur, i meningen att licensen smittar vidare till annan programvara som baseras på GPL, kan paradoxalt nog inverka bromsande för spridningen av fri- och öppen programvara då kommersiella produkter som utvecklarna vill distribuera utan att nödvändigtvis behöva distribuera ”sin” programkod ej kan baseras på GPL-licensierad kod (Microsoft, 2004c).

Lösningen på detta ”problem” är GNU LGPL (GNU Lesser General Public License, eller som den ursprungligen benämndes GNU Library General Public License). LGPL innebär i korthet att programvara baserad på LGPL-kod ej smittas vidare av LGPL utan kan distribueras under godtyckliga licensvillkor. Dock innebär LGPL att den kod som ligger under LGPL-kod alltid ska vara fri programvara och att alla modifikationer av LGPL-koden måste distribueras vidare under LGPL (GNU, 2004g; GNU, 2004h). Exempel på programvara som använder LGPL:

- Glibc (C-bibliotek) – (http://www.gnu.org/software/libc/libc.html)
- Jboss (J2EE-server) – (http://www.jboss.org/index.html)

4.3.2 Apache, X- och BSD Style

Apache-licensen och dess släktingar, till exempel X- och BSD-licenserna, skiljer sig radikalt från GPL och LGPL. Den här gruppen av licenser ger dig rätt att göra i stort sett vad som helst med ett program under deras licensvillkor. Perens (1999) skriver att för X- och BSD-licenserna är detta en effekt av att de ursprungligen var sponsrade av den amerikanska staten.

Den stora skillnaden mot GPL och LGPL är att du kan göra ändring i programmen och distribuera programmet vidare i exekverbart format utan att dela med dig av källkoden. Den här kategorin av licenser betraktas ändå som ”Open Source” eftersom definitionen av Open Source gjord av OSI ej ställer som krav att modifikationer av ett program ska ha samma licens som den ursprungliga instansen (OSI, 2004b).

Låt oss kortfattat bekanta oss med två av licenstyperna som vi har nämnt ovan, nämligen Apache samt BSD-liknande licenser

4.3.2.1 Apache license

Apache license är den modell som organisationen Apache Software Foundation (ASF) släpper sin kod under. ASFs programvara har en mycket stor spridning då gruppen bland annat ligger bakom Webbservern Apache Server, eller httpd (Apache, 2004a).

Exempel:

- Apache Server (http://httpd.apache.org)

4.3.2.2 BSD

Vi hämtar det sista exemplet från OpenBSD som använder vad det kallar ”Berkeley copyright”. Målet är att ”Berkeley copyright”-licensen ska göra det möjligt att distribuera "Berkeley" Unix helt fritt (OpenBSD, 2004a).

Exempel:

- OpenBSD (http://www.openbsd.org)

4.3.3 NPL / MPL

Netscape Public License (NPL) och Mozilla Public License (MPL) är de licenser som Netscape och senare efterföljaren Mozilla distribueras och utvecklas under (Mozilla, 2004a; Moody, 2001) . NPL var den licens under vilken Netscape först släpptes när källkoden öppnades upp. Bland annat på grund av avtal med tredjepartsleverantörer så valde Netscape att införa specifika undantag och rättigheter för sig själva (Moody, 2001). MPL kan ses som en mer ”rättvis” licens men ligger ändå en bit ifrån exempelvis GPL. Varken NPL eller MPL är enligt GNU att betrakta som kompatibla med GPL och LGPL och detta gör det exempelvis omöjligt för MPL-baserade program att dra nytta av (L)GPL baserad mjukvara och vice versa. Projektet Mozilla försöker därför ändra sina licenser så att programmen kan användas under tre olika licenser (MPL, GPL och LGPL) (Mozilla, 2004b) .

Exempel:

- Mozilla (http://www.mozilla.org/products/mozilla1.x/)
- Firefox (http://www.mozilla.org/products/firefox/)

4.3.4 Public domain

Vi redan tidigare bekantat oss med Public Domain som en typ av mjukvara. Låt oss dock repetera genom att citera OpenBSD-projektet: ”För att program ska vara Public Domain så ska upphovsmannen frånsäga sig alla rättigheter och erbjuda materialet utan restriktioner” (vår översättning) (OpenBSD, 2004).

Exempel:

- The public-domain Prolog library (http://www.j-paine.org/prolog/library.html)

4.3.5 Jämförelse mellan olika licenser

Vi avslutar avsnittet kring licenser för FOSS med en jämförelse mellan ett antal licenser i tabell 4.1 nedan:

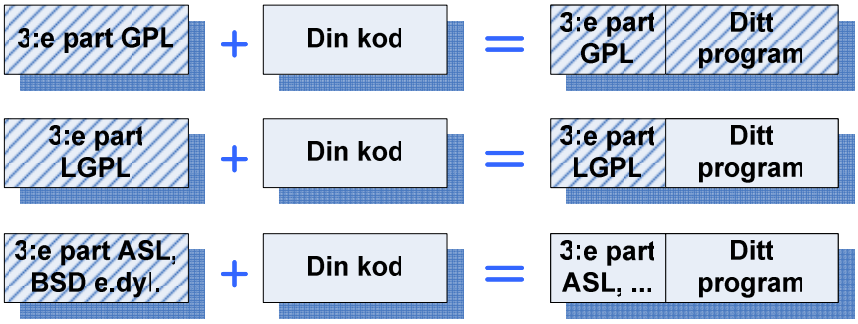
Tabell 4.1: Jämförelse mellan olika licenser

Licens	Kan blandas med icke-fri mjukvara	Modifieringar kan göras utan att de görs publikt tillgängliga	Kan fritt licensieras under annan licens	Har copyright-hållaren speciella rättigheter till dina modifieringar
GPL	-	-	-	-
LGPL	X	-	-	-
BSD	X	X	-	-
NPL	X	X	-	X
MPL	X	X	-	-
Public Domain	X	X	X	-

Tabellen ovan är baserad på ett kapitel av Bruce Perens i "Open Sources: Voices from the Open Source Revolution" (Perens, 1999). Som tabell 4.1 ovan visar så är GPL den mest restriktiva licensformen, men därigenom också den som ger störst frihet eftersom den förhindrar att rättigheterna exempelvis för kopiering och access till källkoden begränsas. I den andra ändan av spektrat ligger Public Domain som genom sin brist på begränsningar och ägandeskap ger var och en rätten att hävda ägandeskap på sina kopior och tillföra godtyckliga begräsningar exempelvis genom att förbjuda tredje part att göra egna kopior av instanser erhållna av en modifierande part. I NPL hade företaget Netscape speciella rättigheter som inte andra

upphovsmän som deltog i utvecklingen av Netscape hade (fram till dess att Netscape införde MPL).

Låt oss även betrakta effekterna av att inkludera några olika typer av fri programvara och öppen källkod som en del av ett program.



Figur 4.2: *Effekter av användning av FOSS i egen programvara.*

I figur 4.2 ovan försöker vi grafiskt att illustrera resultaten av att infoga tredje parts programvara som är fri programvara eller öppen källkod med egenutvecklad kod. GPL smittar över till ditt program om du väljer att distribuera det vidare. LGPL smittar inte, men om du gör ändringar i tredjepartskoden som är licensierade med LGPL måste du göra ändringarna fritt tillgängliga om du väljer att distribuera ditt program vidare till en annan part. Slutligen ser vi effekten av att sammanfoga program med en tillåtande licens som exempelvis Apache Software Licens eller en BSD-liknande licens med din kod. I detta fall kan men behöver du inte dela med dig av vare sig ditt program eller eventuella ändringar i den från tredje part inkluderade koden.

4.4 Sammanfattning

Vi har i kapitlet konstaterat att mjukvara omfattas av immaterialrätt och närmare bestämt för vår diskussion upphovsrätten. Den som äger upphovsrätten till ett program kan ge en annan part rätten att använda programmet genom en så kallad licens eller avsäga sig rätten till programmet och placera det i allmänhetens ägor (Public Domain).

Först då licensmekanismerna bakom såväl fri programvara och öppen källkod som icke-FOSS-programvara förstås kan en meningsfyll analys och diskussion kring affärsmodeller kring fri programvara och öppen källkod föras. Vi har i kapitlet lärt känna ett antal olika licenstyper. Den mest framträdande är kanske copyleftade licenser och det finurliga greppet att applicera copyright till att garantera rätten till spridning och hindra ytterligare begränsningar. Vi har även tydliggjort att FOSS i grund och botten och speciellt för Free Software handlar om fri som i frihet snarare än i betydelsen gratis. Bara för att ett program är gratis så behöver det inte vara fritt.

Friheten kan begränsas genom regler för vidaredistribution men också genom att källkoden till ett program inte är öppen vilket drastiskt hindrar möjligheter till egna modifikationer.

Vidare kan vi notera att enbart för att ett program baserar sig på Open Source så behöver det inte tvunget vidaredistribueras under en ”Open Source”-licens. Exempel på detta Apache- och BSD-liknade licenser som exempelvis inte hindrar användaren från att göra egna ändringar, utan att dela med sig av källkoden och även ta betalt för sin modifierade variant.

Olofsson (2003a) erbjuder en annan indelning av fri programvara och programvara med öppen källkod. Olofsson ställer upp fyra generella modeller för licensiering av fri och öppen källkod. Licensieringsmodellerna enligt denna indelning är: tillåtande licenser, såsom BSD och Apache license, som ger licenstagaren stor frihet; restriktiva licenser, såsom GPL och andra copyleftade licenser, som försäkrar sig om att licenstagaren som bearbetar programmet ger ändringarna tillbaka till upphovsmannen om programmet distribueras vidare; dubbellicensering, såsom MySQL och TrollTech, där licensgivaren ger ut programmet under flera licenser och i fallet MySQL exempelvis både en traditionell proprietär licens och GPL; slutligen finner vi i denna indelning kompromissande licenser, såsom LGPL och MPL, där man försöker förena det bästa ur två världar.

En gemensam tanke om att fri och öppet tillgänglig källkod är bra till trots skiljer sig olika FOSS-licenser vittgående åt. I nästa avsnitt kommer vi att exemplifiera några av de licenstyper som vi har bekantat oss med i detta kapitel.

5 Exempel på programvarukategorier

I detta kapitel presenterar vi ett urval av såväl FOSS som icke-FOSS-projekt och program, vilka vi finner vara representativa och intressanta bland annat vad gäller popularitet och användningsområde. Framställningens tyngdpunkt ligger på FOSS-program.

Låt oss börja med några exempel från världen av fri programvara och öppen källkod.

5.1 Exempel på FOSS

Vi kommer nu att stifta bekantskap med ett antal program som kommer med öppen källkod. Det första exemplet, Linux, är kanske det mest kända.

5.1.1 Linux

För att ge exempel på ”Open Source”-projekt faller det sig naturligt att inleda med Linux. Linux är ett av de mest kända exemplen på ett lyckat ”Open Source”-projekt och blev snabbt kultförklarat inom datorvärlden. Låt oss titta närmare på vad Linux är, bakgrunden till Linux skapande och studera dess utbredning.

Linux är en unix-klon där användaren interagerar med genom kommandon som körs i ett skal (shell). Detta skal kan även vara grafiskt och på så sätt förenkla för användaren att interagera med operativsystemet. Exempel på sådana grafiska användar- eller skrivbordsmiljöer är KDE (KDE, 2004a) och GNOME (GNOME, 2004a). Dessa grafiska skal gör att Linux får ett utseende på ytan väldigt likt Windows eller Macintosh operativsystem för att jämföra även om operativsystemet som sådant väsentligt skiljer sig från Windows. Linux följer POSIX standard för operativsystem, ser ut som Unix på ytan, men med en annan kodbas. Till skillnad från Windows är också Linux helt gratis att installera och använda.

Skaparen av Linux var som tidigare nämnts Linus Torvalds (Raymond, 2001a; Moody 2001). Torvalds skapade Linux av två orsaker: Dels ville han lära sig mer om mikroprocessorn han studerade för tillfället (vilket kunde göras genom att skapa ett operativsystem som kommunicerar med processorn) och dels hade Linus upptäckt att en annan version av Unix operativsystem kallat Minix inte hade den funktionalitet han efterfrågade (Moody, 2001 beskriver förloppet med stor detaljrikedom).

Comment [f1]: Använda Torvalds isf Linus?

I september 1991 släpper så Linus, efter att ha använt Marice J Bach's bok "Design of the Unix Operating System" (Bach, 1986), den första (0.01) versionen av Linux kärna, 22 år gammal. Ungefär en månad senare kommer den första "officiella" versionen av Linux, version 0.02. Denna version av Linux kunde bland annat köra GNU bash (GNU Bourne Again Shell. GNU bash är ett skal, eller annorlunda uttryckt en kommandotolk, i vilken (textbaserade) kommandon kan köras. Den andra versionen av Linux klarade även att köra kompilatorn GCC (GNU Compiler Collection), men inte så mycket mer än så. I januari 1992 släpps version 0.12, där licensvillkoren ändras till GNU GPL (GNU General Public License). Fler versioner släpps löpande, och så har det fortsatt. Yggdrasil (ett företag grundat av Adam Richter) släpper första CD-ROM distributionen 1992. Tack vare Internet börjar Linux också nu bli spritt över världen. Uppgifterna är hämtade från The History of Computing Foundation (2002), Moody (2001) samt Bezroukov (2004).

Det är nu över ett decennium sedan Linus började experimentera men han är fortfarande inblandad i utvecklingen av Linux kärna även om många företag idag hjälper till att distribuera Linux. Operativsystemet har blivit ett av världens mest spridda och används och underhålls av miljoner människor världen över. Underhåll sker genom att vem som helst får lov att koda delar av operativsystemet, något som självfallet ställer som krav att källkoden finns fritt tillgänglig. Dessa delar kan sedan skickas in för godkännande och om sådant ges kan de inkluderas i nästa version av Linux kärna.

Linus säger själv om Linux *"… Linux was and is a major democratic OS and a major democratic social force in computing"* (Bezroukov, 2004) – det är alltså enligt Linus en stark social kraft i datorvärlden, en motpol till licenskostnader och monopolställningar.

Linux är som sagt fritt att använda, men kommersiell support kostar givetvis pengar. Det finns även, bokstavligt talat, miljoner av "supportgivare" över hela världen i form av användare som hjälper andra användare i diskussionsforum etc. Kommersiella versioner av Linux finns i form av olika paketeringar. Återförsäljare av dessa paketeringar (exempelvis Red Hat, Debian, Mandrake, SuSE med flera) ger då ofta support på Linux som operativsystem. Det finns alltså inte en officiell Linux-leverantör och inte ens en officiell version av Linux utan olika distributioner kan skilja sig något åt (Distrowatch, 2004a). En sammanställning av olika Linux-distributioner finns på exempelvis i Distrowatch (2004a) och Linux.org (2004a).

En stor del av vad Linux och Linux-support innebär är "Linux Users Group", vanligen LUG. Lokala användargrupper eller LUG:ar finns över hela världen. Dessa grupper låter användare/nybörjare få hjälp och support av andra användare, som befinner sig geografiskt nära som talar samma språk. Detta kan behövas speciellt för den ovane Linuxanvändaren som kanske mest använt Windows eller Macintosh operativsystem. Linux i sig själv innehåller ingen direkt användarnära funktionalitet utan det är programmen som körs under Linux som är intressanta för slutanvändaren. Detta är vad många distributörer av Linux tar betalt för, att lägga ihop olika program som kompletterar varandra till ett "paket" som lätt kan installeras av användaren (Linux.com, 2004a). Enligt Linux.com (2004a) har också ett tidigare problem varit att inte tillräckliga program och drivrutiner funnits att tillgå. Detta har emellertid förändrats den senaste tiden och fler och bättre program finns nu att tillgå

gratis, något som förutspåtts av många förespråkare för ”Open Source”-tanken. Målet med GNU-projektet var ju att ta fram ett fritt operativsystem med en komplett uppsättning av fria program (FSF, 2004c). Linux har dragit nytta av detta och inkluderar GNU-program och program som använder GNU-licenserna GNU GPL (GNU General Public License) och GNU LGPL (GNU Lesser General Public License). Vissa vill därför hävda att en mer korrekt benämning av Linux är GNU/Linux (Debian, 2004b).

Användandet av Linux har hursomhelst spritt sig från tekniker och ”datorbranschfolk” i Linux inledande historia till att vara mer utbrett bland storföretag inom alla tänkbara branscher och inriktningar och även ”vanliga” datoranvändare, även om det naturligtvis är svårt för att inte säga omöjligt att finna det verkliga antalet Linuxanvändare i världen. (The Linux Counter, 2004).

5.1.2 Perl

Perl -- When the best is good enough. (http://www.perl.org/)

Perl är ett annat intressant och belysande exempel på fenomenet öppen källkod. En man vid namn Larry Wall arbetade 1986 som systemadministratör för ett projekt och behövde ett verktyg bland annat för att hantera (text)filer. Inget av de verktyg som Wall kände till passade riktigt till det som han behövde göra. Därför började Wall själv skriva sitt eget skriptspråk som han ursprungligen kallade för Pearl, men som så småningom av olika orsaker kom att mutera till Perl som kan utläsas som ett akronym för ”Practical Extraction And Report Language” (Wall et al, 2000; Moody, 2001). Verktyget kom att användas av fler applikationer i projektet. Då Wall flyttades till nya arbetsuppgifter tog han med sig Perl och anpassade det till nya behov och utökade språket med ett paket för reguljära uttryck. Arbetet lades därefter ut offentligt på Internet och Wall fick fler och fler förfrågningar om programmet. Han fortsatte att vidareutveckla och ändra verktyget och så småningom strömmade fler användare till och fler personer hjälpte till med utvecklingen av språket. (Perl.org, 2004a).

Till Perl finns massvis med öppna perlprogram och dokumentation. Det mesta finns samlat på CPAN (Comprehensive Perl Archive Network). I november 2004 fanns det 7241 moduler listade på CPAN (CPAN, 2004a).

Perl finns tillgänglig under två stycken licenser: Artistic License och GNU GPL. När man använder Perl kan man själv välja den licens som passar en bäst CPAN (2004b).

5.1.3 Apache

När webben uppfanns blev webbservern (httpd, http daemon), precis som webbläsaren Mosaic, utvecklad och spridd ifrån National Center for Supercomputing

Applications (NSCA) vid University of Illinois. Webbservern, eller om man så vill http demonen var ursprungligen utvecklad av en man vid namn Rob McCool. McCool lämnade dock NSCA år 1994 och utvecklingen av httpd stannade då av. I takt med att webben hade börjat ta fart hade dock flera utökningar och fixar gjort till den ursprungliga httpd-koden av webbmasters över hela världen Apache (2004a).

Genom e-postkontakter mellan en grupp webbmasters beslutade de att samla ihop sina ändringar eller om man så vill ”patchar” till NSCAs httpd. Den första officiella releasen (0.6.2) av Apache gjordes publikt tillgänglig i april 1995 (Apache, 2004a).

Det tog sedan mindre än ett år för Apache att ta platsen som den mest spridda webbservern. Enligt flera undersökningar är den fortfarande dominerande. I en undersökning av Netcraft från november 2004 drev Apache httpd exempelvis mer än 67 procent av webbservrarna på Internet (Netcraft, 2004a).

Gruppen bakom Apache httpd bildade 1999 Apache Software Foundation för att citat: *”tillhandahålla organisatorisk, juridisk, och finansiellt stöd för Apache…”* (Apache, 2004b) (vår översättning).

Under Apache Software Foundation ligger numera ett flertal välkända ”Open Source”-projekt, bland annat XML-tolken Xerces (http://xml.apache.org/), JSP/Servlet-motorn Tomcat (http://jakarta.apache.org/tomcat/index.html) och makeverktyget Ant (http://ant.apache.org) för att bara nämna ett fåtal exempel.

Programmen som Apache Software Foundation står bakom licensieras under licensen Apache License vilken är en tämligen tillåtande licens, till exempel i det att mottagaren av ett program har rätt licensiera programmet vidare under en annan licens om så önskas (Apache, 2004c; Apache, 2004d).

5.1.4 eMule

Emule är ett P2P (peer-to-peer) program för delning av filer. Själva skriver personerna som ligger bakom eMule följande: *”As of today, eMule is one of the biggest and most reliable peer-to-peer file sharing clients around the world. Thanks to it's open source policy many developers are able to contribute to the project, making the network more efficient with each release.”* (E-mule, 2004a). Programmet Emule är alltså fri programvara och licensieras med licensen GNU General Public License. Det finns ett flertal kloner som baserar sig på den gemensamma koden och de protokoll som där används. Klonerna lägger till extra funktionalitet utöver standardprogrammet. När nu funktionalitet och buggrättningar görs i standardversionen inkorporeras de så småningom i klonerna och omvänt kommer ibland buggrättningar och tillägg in i standardversionen från klonerna. Slutanvändarna av eMule är dock typiskt privatpersoner som bara använder applikationen för att ladda ner och dela ut filer till andra. Slutanvändaren laddar då ner Emule eller någon av dess kloner i binär form och använder det utan att själv modifiera koden, men möjligheten finns för den intresserade och kunnige att studera och ändra i koden. Emule-projektet använder sig som många andra ”Open Source”-

projekt av den gratis webbplatsen SourceForge.net och projekt återfinns på http://sourceforge.net/projects/emule/.

5.1.5 Open Office

Open Office är den kanske främste konkurrenten till Microsoft Office ifrån ”Open Source”-hållet. Som de flesta andra program i ”Open Source”-världen är Open Office både en applikation och ett projekt för att ta fram och vidareutveckla applikationen. Detta och det faktum att applikationen ska baseras på öppna standarder och format ges eftertryck för i programförklaringen för OpenOffice: *”To create, as a community, the leading international office suite that will run on all major platforms and provide access to all functionality and data through open-component based APIs and an XML-based file format.”* (Open Office, 2004a). Noterbart är att företaget SUN till stor del sponsrar och driver utvecklingen av Open Office. Den ursprungliga koden för Open Office kommer från det programmet Staroffice vilket är en traditionell produkt som fortfarande säljs och marknadsförs av SUN parallellt med Open Office (SUN, 2004b, 2004c). Open Office licensieras under licenserna GNU General Public License, GNU Lesser General Public License samt Sun Industry Standards Source License (Open Office, 2004b).

5.1.6 MySQL

MySQL är en databas som utvecklas av ett (svenskt) kommersiellt företag (MySQL AB) där MySQL och tjänster omkring MySQL är de enda intäktskällorna. En intressant faktor kring MySQL är att den förutom att vara väl spridd finns under två licenser. En användare kan välja att använda produkten under en “Open Source“-licens eller under en mer traditionell proprietär stängd licens, citat: *”All of our products are available under open source licenses, but we also sell commercial licenses for all of the products so they can be adopted in situations where an open source solution is not appropriate.”* (MySQL, 2004a).

Mårten Mickos som är VD för MySQL AB säger: *”- Våra betalande kunder är tacksamma för att vi för ut källkoden till alla de här högintelligenta unga killarna mest, och några tjejer, som bygger spelsajter eller forskningsprojekt med den. För de vet att den vägen härdas produkten. Vi hittar alla fel och vi fixar dem.”* (Ny Teknik, 2004a). Något som i viss mån skiljer MySQL AB från andra företag som sysslar med Open Source är att de inte tar källkod från användarna för att inte riskera problem med ägandeskap och upphovsrätt (Ny Teknik, 2004a).

Eftersom företaget MySQL AB äger all rätt till sin källkod kan de ge ut den under en traditionell strikt och kommersiell licens parallellt med den fria licensen GNU General Public License (GPL). Då GPL ställer krav att de program som använder kod under GPL även de ska vara GPL (eller LGPL) väljer många företag att köpa en traditionell kommersiell licens av MySQL som inte ställer krav på att den kod som använder MySQL-databasen ska vara Open Source, men som också ställer

begränsningar kring hur instanser av programmet som köpts in under en kommersiell licens kan användas och vidaredistribueras (MySQL, 2004a, 2004b).

5.1.7 JBoss

JBoss är en populär javabaserad applikationsserver som varit i utveckling sedan 1999. Applikationsservern kan sägas tillhandahålla tjänster, såsom transaktioner, persistensmappning och behörighetskontroll till de applikationer (eller om man så vill ”java beans”-komponenter som exekverar däri). JBoss implementerar javastandarden J2EE (Java 2 Enterprise Edition). Bakom programmet JBoss finns företaget JBoss Inc (tidigare sammanslutningen JBoss Group) som tillhandahåller utbildning, support, dokumentation och konsulttjänster kring Jboss.

JBoss var ursprungligen licensierad med licensen GNU GPL, men eftersom det fanns oro för att applikationer som exekverade i JBoss skulle smittas av GNU GPL, inhämtades tillstånd från samtliga utvecklare för att ge ut nya version av Jboss under licensen GNU LGPL. Motiveringen för valet av GNU LGPL är *”Your applications are yours and your intellectual property is secure. Only changes made to the JBoss Source code must be released back to the open source community. The LGPL has two important implications; first, no one can take JBoss, close it and brand for commercial sale as JBoss Enterprise, not even JBoss, Inc.; and second, we cannot revoke the LGPL license we give you.“* (JBoss, 2004b). JBoss (2004b) menar att detta gör att användarens investering i Jboss som infrastruktur för sina applikationer framtidssäker.

Jboss integrerar och kan samverka med andra ”Open Source”-program såsom exempelvis Hibernate, Tomcat och Eclipse (JBoss, 2004c).

5.2 Closed Source

Vi skall också ge exempel på några program som inte är Open Source. För att göra det lätt för oss och läsaren har vi valt att kalla denna grupp av program för Closed Source. Termen Closed Source tidigare använts av bland annat OSI (2004d), men benämningen har också blivit ifrågasatt exempelvis i GNU (2004i). Vi använder även begreppet Closed synonymt med proprietär programvara.

5.2.1 Microsoft Windows

Det mest iögonfallande exemplet på vad som inte är fri programvara eller öppen källkod är operativsystemet Microsoft Windows. Sedan det skapades av företaget Microsoft har strategin med operativsystemet varit att källkoden har varit stängd och inga förändringar i koden fått göras utanför Microsofts väggar. Utvecklingen finansieras av licens- och försäljningsintäkter från operativsystemet. Det finns olika

former av licenser beroende på vem användaren är och hur många licenser användaren behöver (en licens behövs per användare). (Microsoft 2004d)

5.2.2 Microsoft Officepaketet

Även Officepaketet kommer från Microsoft och är säkert välkänt för nästan varje datoranvändare. Licensieringsformerna är liknande de för Windows (Microsoft 2004d).

5.2.3 Andra stängda program

Till denna kategori räknas exempelvis de flesta datorspel, kartprogram, aktieprogram och många andra typer av program. Gemensamt för dessa är att de utvecklas inom ett företag med syfte att säljas på en öppen marknad och intäkterna från programförsäljningen skall förhoppningsvis täcka kostnaderna för utveckling, marknadsföring och distribution av programvaran. Exempel på ett företag som utvecklar program enligt denna modell är SPCS, Scandinavian PC Systems AB (SPCS, 2004a, 2004b). SPCS har fokuserat på administrativa program, exempelvis program för bokföring.

5.3 Transitioner

Låt oss så slutligen bekanta oss med ett par exempel på vad vi har valt att kalla för transitioner, med vilket vi menar program som har gått från att vara Closed Source till att vara Open Source.

5.3.1 Netscape

Det kanske mest lysande exemplet på ett program som har gått från att vara stängt och proprietärt till att vara öppen källkod är Netscapes webbläsare. Företaget Netscape hade startat att utveckla en webbläsare med många av programmerarna från den första framgångsrika grafiska webbläsaren NSCA Mosaic. Netscape hade nått en dominerande ställning på marknaden för webbläsare trots att deras ursprungliga strategi var att tjäna pengar på sina webbserverprodukter. Webbläsaren Netscape Navigator kostade förhållandevis lite och Netscape tog i första hand betalt för kommersiell användning av programmet. Trots detta blev Netscape Navigator en avsevärd intäktskälla (Rosenberg, 2000).

Så småningom blev även Microsoft intresserat av Internet och webben. Företaget beslutade sig därför att utveckla en egen webbläsare, Internet Explorer (baserat på

NSCA Mosaic) och att paketera webbläsaren som en integrerad del av sina operativsystem. Själva webbläsaren var också gratis att ladda ner. Effekten av detta drag från Microsofts sida var att Netscape inte längre kunde ta betalt för sin webbläsare, åtminstone inte för Windowsversionerna. Företaget Netscape tog därför i början av 1998 beslutet att inte bara ge bort webbläsaren gratis utan också öppna upp källkoden. Rosenberg (2000) kommenterar beslutet: *"Sett från ett världsligt perspektiv så var öppnandet av källkoden till Mozilla (webbläsaren, vår anmärkning) inget glamoröst ögonblick utan helt enkelt en desperat handling när en överväldigande förlust (till Microsoft, vår anmärkning) stod klart"* (Rosenberg, 2000 sid. 36, vår översättning).

Netscape tycks ha hoppats att många duktiga programmerare världen över skulle hjälpa till att utveckla webbläsaren med kodnamnet Mozilla nu när källkoden släppts fri. Det visade sig dock inte bli någon succé och en av huvudpersonerna bakom Mozilla hoppade av efter drygt ett år. Lerner och Tirole (2000) noterar exempelvis att endast drygt två dussin externa utvecklare bidrog till utvecklingen inledningsvis. En anledning till att många utvecklare ursprungligen ställde sig tveksamma till Mozilla var dess licensform Netscape Public License (NPL) som exempelvis gav företaget Netscape rätten att ta kod som andra bidragit med och omlicensiera den under andra villkor (Rosenberg, 2000; Raymond, 2001a).

Så småningom kom en nyutvecklad version av själva webbrenderingskomponenten som lystrade till namnet Gecko. Den första generationen av Mozilla fick aldrig någon riktigt stor spridning. Utvecklingen av en ny Geckobaserad webbläsare resulterade i att version 1.0 av Mozilla Firefox släpptes i slutet av 2004. Denna version har mottagits väl av exempelvis Mossberg (2004a). Noterbart är att "Open Source"-licensen för Mozilla-programmen numera har ändrats så Netscape inte längre har några specifika fördelar (Mozilla, 2004b).

Hur gick det då för företaget Netscape? Jo företaget blev uppköpt av AOL och dess webbläsare som numera baserar sig på Mozilla är långt ifrån lika populär som produkten var under sina glansdagar.

Själva Mozilla-projektet kan trots initiala motgångar ses som lyckat då produkten effektivt förnekar Microsoft en monopolställning på webbläsarmarknaden (Raymond, 2001a). Delar av den kod och de tekniker som tagits fram inom ramen för Mozilla-projektet, bland annat Gecko och användargränssnittsbeskrivningsspråket XUL, har även återanvänts i andra sammanhang.

5.3.2 Eclipse

Eclipse är en generell plattform med en komponentbaserad arkitektur i vilken företag och användare kan bidra med funktionalitet. Det finns färdiga paket att ladda ner där Eclipse fungerar som en integrerad utvecklingsmiljö (IDE, Integrated Development Environment) . Eclipse utvecklades från början av IBM som inledningsvis hade 40 personer som jobbade med projektet. Så småningom "donerade" IBM allting genom att ge ut Eclipse under en Open Source licens, Common Public License (CPL) och

sedan omlicensierades under en något modifierad version kallad Eclipse Public Licensen(EPL) (Eclipse Foundation, 2004b).

År 2004 bildade IBM tillsammans med ett flertal partners en icke-vinstdrivande organisation som numera äger kontrollen över Eclipse. Det intressanta med Eclipse är plattformsaspekten. IBM tillhandahåller en kommersiell proprietär version av ett verktyg för att administrera och utveckla för en applikationsserver (Websphere) vilken dock bygger på Eclipse (IBM, 2004a). Även andra applikationer från IBM bygger eller kommer att bygga på Eclipse i framtiden. Ett exempel på detta är nya versioner av IBM Rational Rose för modellering. IBM kan alltså både dela med sig av kakan och äta upp den utan att hamna i en paradox. Om det bara var IBM som utvecklade egna applikationer på Eclipse vore det hela mindre intressant sett ur ett ”Open Source”-perspektiv, eftersom man då inte skulle uppnå fördelen med den gemensamma utvecklande basen av användare. Det är dock så att även andra kommersiella företag ser fördelar med att använda Eclipse. Eclipse blir den gemensamma infrastrukturen och de olika företagen lägger själva till värde genom specialfunktioner. Det kan till exempel röra sig om att paketera en kompilator, debugger och andra utvecklingsverktyg för exempelvis inbyggda system. I exemplet är det kanske ett företag nischat mot ett smalt segment (kompilatortillverkning) men där det krävs saker runt omkring själva kärnprodukten. Genom att bygga vidare på Eclipse slipper företaget börja från ”scratch”. Om företaget utvecklar plugin:er som inte är en del av deras egentliga affärsområde kan de dela med sig av sina alster och hoppas på fler intressenter använder och underhåller komponenterna.

Eclipse är alltså ett exempel på en programvara som började sitt liv som proprietär programvara för att sedan bli öppen källkod. Applikationer som i sin tur bygger på Eclipse kan i sin tur, men behöver inte, vara Closed Source eller om man så vill proprietär programvara (Clayberg & Rubel, 2004).

5.3.3 Doom

En annan mjukvara som övergått från Closed till Open Source är dataspelet Doom, version 1. Detta spel utvecklades av företaget id software och dök upp på marknaden 1993. Spelet blev en succé och många kopior såldes världen över, bland annat på grund av en för tiden enastående grafik. Exempelvis grafiken gav id software ett försprång gentemot sina konkurrenter, ett försprång som var värt att skydda och lönsamt att exploatera genom licensförsäljning (Raymond, 2001a). Kostnaderna för spelet när det väl utvecklats var också blygsamma. När spelet släpptes fanns alltså ingen större motivation att öppna källkoden.

Efter hand som tiden gick hände emellertid saker på marknaden. Konkurrenter började äta upp det försprång som tidigare skilt företagen åt samtidigt som kraven ökade på tilläggsprodukter till Doom och möjlighet att spela flera spelare samtidigt, något som ledde fram till spelformen ”Deathmatch” som snabbt blev mycket populär. Dessa förändringar innebar att id software lade ned många programmeringstimmar på tilläggsfunktioner till det ursprungliga spelet samtidigt som konkurrenterna fortsatte att hämta in på det tekniska försprång företaget tidigare

haft. Tekniskt begåvade och intresserade spelare av Doom började också intressera sig för att göra egna tilläggsmoduler eller nya banor till spelet.

Detta sammantaget gjorde att vinstmöjligheterna med att öppna källkoden ökade. Företaget kunde då gratis få fler programmerare som bidrog till att hålla uppe intresset för spelet och komma med förbättringar samtidigt som de egna programmerarna kunde fortsätta utvecklingen av nästa spel. Till sist blev det ekonomiskt försvarbart att öppna källkoden och i stället fortsätta tjäna pengar på spelet på denna eftermarknad, exempelvis genom att ge ut tilläggsbanor/beskrivningar/dokumentation (Raymond, 2001a). Tilläggas bör dock att id software inte fortsatte med denna linje för sina senare spel. Exempelvis har nyligen spelet Doom, version 3 släppts vilket ju inte är Open Source utan användaren måste betala för rätten att köra detta program och har ingen rättighet att ändra källkoden till programmet.

5.4 Sammanfattning

I detta kapitel har vi lärt känna ett antal program såväl ”öppna” som ”stängda”. Vi har i exemplen visat på olika drivkrafter bakom programmen och val av licenstyp. Ett exempel som vi visade på var webbservern Apache där den ursprungliga drivkraften var ett gemensamt behov och intresse hos ett antal vitt skiljda personer. Trots att de jobbade för olika företag och organisationer och var spridda geografiskt samarbetade de över nätet för att gemensamt förbättra ett program. Detta kan vi jämföra med Raymonds åsikt att varje stycke bra kod härstammar från en programmerares personliga behov, *”Every good work of software starts by scratching a developer’s personal itch”* (Raymond, 2001a).

I kapitlet har vi även kortfattat gett prov på program som inte är Open Source, exempelvis Microsoft Office.

Vi har även visat exempel på hur program har bytt licens av olika anledningar. Eclipse var ett exempel på hur storföretaget IBM gav ut en intern komponent fritt och därigenom skapade en ny miljö som samtidigt ställer krav på både samarbete och konkurrens. Enligt indelningen i Olofsson (2003a) skulle licensen för Eclipse hamna i facket för kompromissande licenser. Det mest kända och uppmärksammade fallet av program som bytt licensmodell från proprietär till öppen källkod torde vara då Nescape gav ut källkoden till sin webbläsare Netscape Navigator som öppen källkod.

6 Marknaden och FOSS

I detta avsnitt kommer vi att introducera ett antal teoretiska begrepp och modeller kring marknaden och då främst marknaden för mjukvara. Vidare kommer vi att relatera dessa begrepp och modeller till fenomenet Open Source. Vi börjar direkt med att bekanta oss med den ”kommersiella världen” och dess spelplan – marknaden.

6.1 Vad är marknaden?

Hur ska marknaden som begrepp definieras? Vi låter boken ”Företag och Marknad – Samarbete och konkurrens” (Ljung et al, 1994) göra ett försök att definiera marknaden åt oss: *”Marknaden kan betraktas som en organisation med en viss arbetsfördelning och vissa samordningsmekanismer”* (Ljung et al, 1994 sid. 44). De varor som produceras för konsumtion passerar innan de är färdiga genom ett antal ”aktivitetssystem”. Produktion och konsumtion skiljs åt av dessa aktivitetssystem. Marknaden får därför sägas inkludera både den producerande och den köpande parten – det måste ju finnas två parter för att ett utbyte av varor eller tjänster skall kunna ske. En liknande definition av marknaden ger Kotler och Trias de Bes*: ”... de personer/företag som köper eller kan köpa varor eller använda tjänster i en viss situation för att täcka ett givet behov”* (Kotler & Trias de Bes 2003 sid. 19).

I dagens komplexa och globala affärsvärld är det dock inte alldeles enkelt att kunna överblicka och konkretisera ”marknaden”. Vi kommer framöver i detta kapitel att redovisa några ekonomiska teorier som alla har sin syn på marknaden som begrepp och dess aktörer. För att sätta ramen för den verksamhet vi intresserar oss för vill vi dock först konkretisera de produkter vi främst intresserar oss för (samt rena tjänster i anslutning till dessa produkter).

6.2 Karaktäristik för mjukvara

Mjukvara har en lite speciell karaktäristik och mjukvaruindustrin kan av den orsaken inte rakt av jämföras med exempelvis traditionell tillverkande industri. Själv utvecklingsfasen inom mjukvaruindustrin kan emellertid jämföras med traditionell ingenjörskonst. Gamma et al (1995) gör med sin bok ”Design Patterns” kopplingar till traditionell arkitektur. Gamma et al (1995) går metodiskt tillväga och kategoriserar olika mönster för lösningar av återkommande problem och designöverväganden.

Boken, som har fått flera efterföljare, exempelvis Schmidt et al (2000) och Fowler (2002), ger ett klart ingenjörsmässigt intryck.

När det kommer till produktionsfasen liknar mjukvaruindustrin exempelvis musikbranschen då material och reproduktionskostnader är extremt låga. Mjukvara levereras nuförtiden ofta på en CD- eller en Dvd-skiva och det är inte alltid som en tryckt manual följer med. Ett annat vanligt leveranssätt är att låta kunden ladda ner mjukvaran från tillverkarens hemsida på Internet. Precis som i fallet med musik- och filmindustrin är detta leveranssätt en fördel inte bara för kunden som får sin vara direkt utan också en besparing för företagen som säljer mjukvara som inte behöver finansiera en distributionskedja för att få ut sina produkter på marknaden, vilket vi kommer närmare att förklara i ekonomikapitlet framöver.

6.3 Ekonomiska modeller

När vi nu presenterat marknaden och de produkter och tjänster vi ämnar undersöka vill vi övergå till att betrakta tidigare litteratur kring hur dessa tillsammans kan passa ihop, ur perspektivet vilka ekonomiska möjligheter som kan finnas för företag som utvecklar produkter eller tjänster med inslag av öppen källkod. Dels kommer vi att beskriva rena affärsmodeller, det vill säga rent krasst idéer om hur företag kan tjäna pengar, och dels kommer vi att beskriva andra ekonomiska modeller som kan användas för att skapa nya affärsmodeller eller i kombination med befintliga affärsmodeller för att förstå och förbättra dessa.

6.3.1 Affärsmodeller

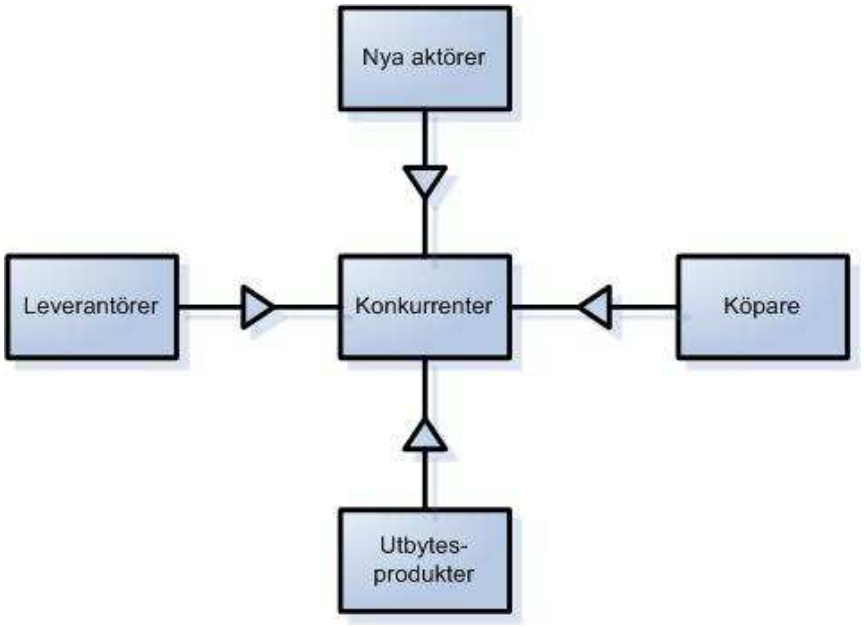
En affärsmodell kan alltså kortfattat sägas vara en modell för hur ett företag skall kunna tjäna pengar. Affärsmodellen talar om detta genom att visa och förtydliga var i ”värdekedjan” produkten eller tjänsten befinner sig (Rappa, 2004a). Värdekedjan hänför sig från Porter (1985) som beskriver begreppet som den totala kedjan av händelser och personer från tillverkning till leverans till slutkund för en produkt eller tjänst.

Det finns mycket tidigare litteratur om marknaden och dess aktörer, vilket så småningom lett fram till teoribildning kring hur en affärsmodell är uppbyggd. Vi gör här ingen ansats att göra en fullständig beskrivning över all denna tidigare litteratur, utan vi väljer ut ett antal som vi anser vara viktigast för vårt syfte och som vi kommer att återknyta till i analyskapitlet. Vi anser dem vara viktigast antingen för att de beskriver en struktur på ett enkelt och lättförståeligt sätt eller för att de tar hänsyn till tidigare litteratur och kan sägas vara representativa för dagens forskning och allmänna uppfattning.

Den tidiga litteraturen innehåller en del exempel på synen på marknaden och speciellt företagen inom denna marknad. En av de mest kända synsätten är det så kallade

”Resursbaserade synsättet” (Resource Based View, i fortsättningen benämnt RBV). Enligt detta synsätt, framlagt av Penrose (1959), ses företaget som bestående av en bunt resurser (som kan vara ekonomiska tillgångar, kunskap etc.) som det gäller att förvalta, kombinera och utnyttja på bästa sätt för att maximera företagets vinster över tiden. Genom att använda och utnyttja företagets unika och värdefulla resurser på bästa sätt kan marknadsfördelar mot konkurrerande företag uppnås. Ju högre barriärer företaget kan resa mot kopiering av dessa unika resurser desto längre kan fördelarna behållas (Wade och Hulland, 2004). Detta synsätt har fått en kritik bland annat för att det inte tar hänsyn till sociologiska faktorer både inom och utom företaget samt även på ett flertal andra punkter (beskrivs i detalj av Dahlander, 2004 och Hedman & Kallin, 2002).

Vidare, med rötter i tidigare litteratur, övergår vi till en flitigt använd modell av Porter (1980) kallad ”The five forces model” eller fritt översatt de fem krafterna. Modellen beskriver marknaden utifrån fem krafter: Hotet om nya aktörer på marknaden, rivalitet bland existerande konkurrenter, yttre tryck från snarlika produkter, förhandlingsstyrka från köpare samt förhandlingsstyrka från underleverantörer. Modellen finns avbildad i figur 6.1 nedan.



Figur 6.1: *The five forces*

Modellen beskriver alltså ett företags position på en marknad och ger en möjlighet att utvärdera marknadsstruktur och position för det egna företaget i förhållande till andra företag. En analys av modellen ger att varje ”kraft” på olika sätt påverkar företagets position till det bättre eller till det sämre beroende på styrkan i den specifika kraften. Exempelvis är företagets position stark om det inte finns några konkurrenter, nya aktörer, utbytesprodukter och om det finns många köpare och många underleverantörer medan det omvända förhållandet råder för krafternas motsatser.

Denna modell är enkel men har fått en hel del kritik. Exempelvis har den kritiserats för att inte ta hänsyn till inre drivkrafter inom företaget (Hedman & Kalling, 2002) samt för att i första hand passa in på den statiska marknadsstruktur som rådde under dess tillkomst i början av 80-talet och inte vara anpassad för dagens globala nätverksmarknad (Recklies, 2001). Vi har ändå valt att ta med modellen främst för dess enkelhets skull. Även om dagens marknad, och speciellt marknaden inom FOSS, består av nätverk av olika aktörer (inte nödvändigtvis företag) som påvisas av bland annat Dahlander (2004) hjälper oss Porters modell dessutom att titta närmare på marknadens aktörsroller och hur själva marknaden byggs upp i en kapitalistisk värld.

I vår studie som har tjänster och mjukvaruprodukter i fokus kan vi identifiera ett antal marknadsaktörer relaterade till Porters modell. En mycket viktig part för att ett företag skall kunna existera är köparen av produkten eller tjänsten som företaget levererar. Utan en köpare av en produkt eller tjänst finns endast ideologiska eller personliga skäl att fortsätta utveckla produkten eller tjänsten. Köparens makt är alltså stor genom att köparen ger den utvecklande parten möjlighet att fortsätta utveckla sin produkt, men vad finns det för köpare när produkten är fri? Jo, som vi har sett så är det ju inte endast produkten som sådan man kan ta betalt för utan även tjänster som support, vidareutveckling/kundanpassning, dokumentation etc.

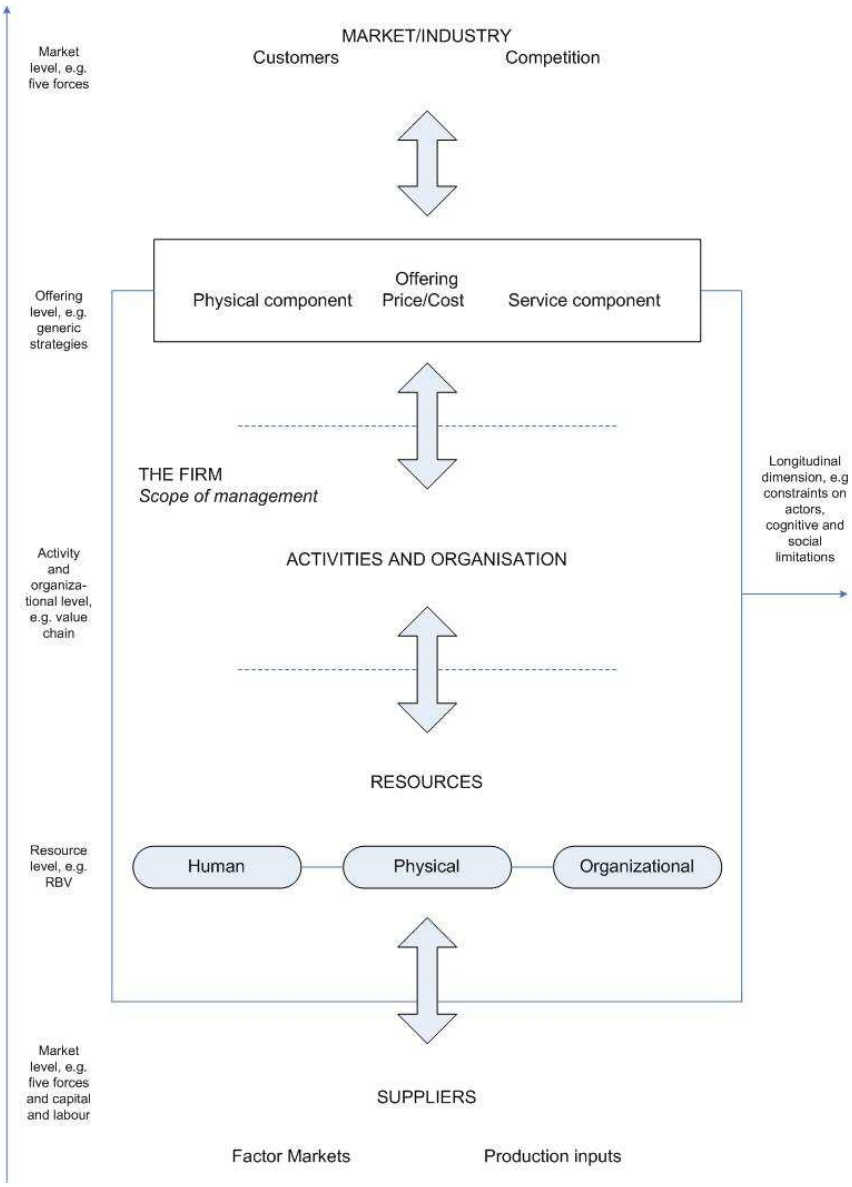
Leverantörer finns ju på flera plan och på flera nivåer. I vårt exempel finns såväl hårdvaru- som mjukvaruleverantörer. Leverantören är också en ekonomisk spelare som måste ha marginal på sina produkter och tjänster (Kotler, 1999). Beroende på antalet leverantörer av likvärdiga produkter är fördelen viktad mot antingen köparen eller leverantören, ju fler potentiella leverantörer desto bättre för köparen enligt principen om tillgång och efterfrågan illustrerad av Porter ovan.

För utvecklaren finns också ett ständigt hot från andra utvecklare/produkter som kan ta fram bättre och billigare produkter (hoten om utbytesprodukter, nya aktörer samt befintliga konkurrenter enligt Porters modell). Detta är kanske viktigare för andra utvecklings- och intäktsmodeller än Open Source där koden/licenser genererar intäkter. Det största hotet i ”Open Source”-fallet ligger troligen mer i att en annan produkt tar över marknaden och även om den inte kostar någonting så kan dess bakomliggande struktur eller utseende vara så olikt den egna produkten att man tappar sitt kunskapsförsprång, exempelvis för konsulttjänster. Detta beskriver Joe Marini från Microsoft i en artikel på Internet som ett skäl till varför inte allting bör vara Open Source*: ”… how do you protect your competitive advantage when your competitors can just look at your source code and cherry-pick the best ideas?”* (Marini, 2004a). Detta är en av de centrala frågeställningarna för ”Open Source”-rörelsen och vi skall i vårt analyskapitel försöka reda ut de mekanismer och övriga faktorer på marknaden som gör att det faktiskt kan gå att tjäna pengar på fenomenet Open Source; beroende på vilken licens man väljer att distribuera sin mjukvara under med eller utan intäkter i form av licensavgifter.

Porter har vidareutvecklat sina tankar till nyare modeller, men vi väljer att gå vidare i vår beskrivning med en modell av Hedman och Kalling (2002) som har en lite vidare syn på begreppen och definierar sju olika komponenter som en affärsmodell består av (här i vår fria översättning): Kunder, konkurrenter, erbjudande, aktivitet/organisation, resurser, leverantörer samt en processkomponent för att

Comment [f2]: Är???

beskriva olika yttre och inre påverkan på modellen. Det finns flera definitioner av affärsmodeller i tidigare litteratur men vi nöjer oss med denna eftersom den är baserad på tidigare litteratur i ämnet och ger en enligt vår mening väl strukturerad syn på marknaden och dess aktörer. Hedman och Kallings modell finns återgiven i figur 6.2 nedan.



Figur 6.2: *Affärsmodell enligt Hedman och Kalling (2002)*

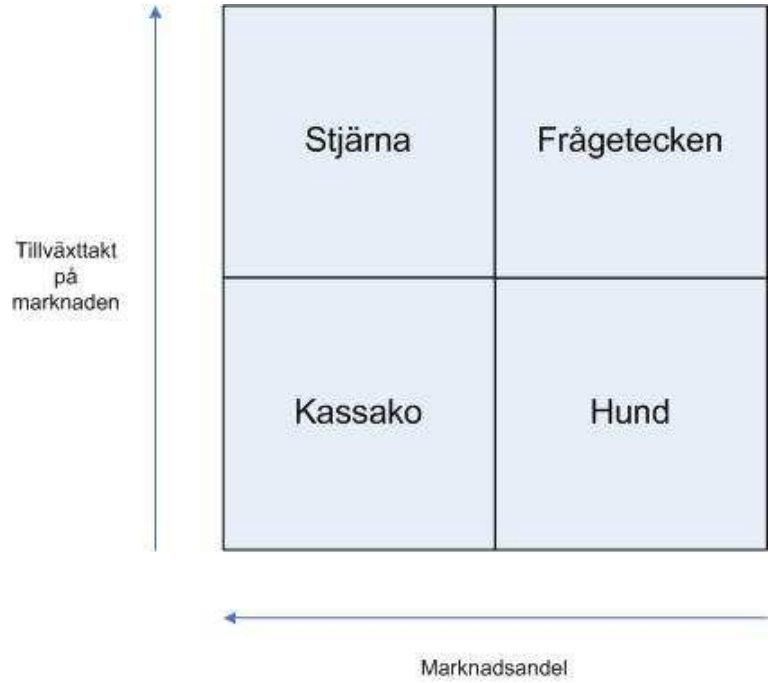
Även om inte explicit uttryckta i modellen visar Hedman och Kalling också att externa aktörer kan vara partners eller konkurrenter i hela affärsprocessen (exempelvis genom att banker och försäkringsbolag delar kunddatabaser). Detta visar att affärsmodellen påverkas av de nätverk av aktörer som existerar runt företaget.

Dessa nätverk är inte minst viktiga när det gäller ”Open Source”-mjukvara vilket bland annat Dahlander (2004) visat.

6.3.2 Övriga ekonomiska modeller

Förutom affärsmodeller skall vi här även undersöka andra ekonomiska teorier vi anser oss behöva för att kunna besvara vår frågeställning adekvat.

En av de mer kända är den matris Boston Consulting Group (BCG) tagit fram som beskriver en produkts livscykel (förkortat PLC, Product Life Cycle, i figuren nedan) – från idé till lönsam produkt till föråldrad produkt. En schematisk bild av denna modell återfinns i figur 6.3 nedan:



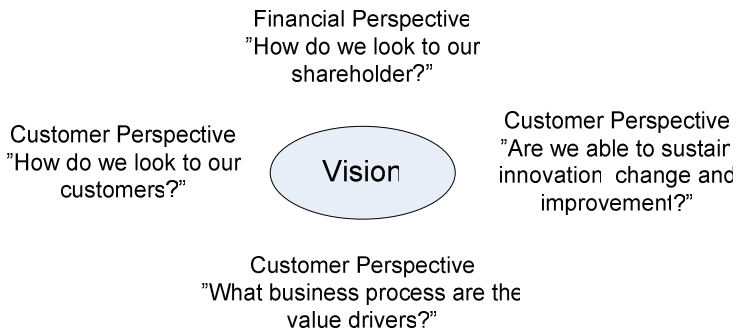
Figur 6.3: *Bostonmatrisen*

Matrisen i figur 6.3 ovan kan kräva en liten förklaring. Tillväxttakten på marknaden symboliserar hur mycket investeringar som behöver läggas på produkten medan marknadsandelen symboliserar hur mycket intäkter produkten ger. På marknader där den säljande parten har en hög marknadsandel och en produkt med hög tillväxt i försäljningen ses produkten som en ”stjärna” (längst upp till vänster i matrisen) medan motsatsen (Låg marknadsandel och vikande försäljningssiffror) ses som en ”hund” (fritt översatt). Om marknadsandelen är hög men tillväxten låg är produkten en så kallad ”kassako” som genererar höga intäkter och hög vinst eftersom de inte kräver någon större vidareutvecklingskostnad. Produkter i den övre högra

kvadranten av matrisen befinner sig under utveckling och symboliseras av ett frågetecken. Vissa av dessa produkter går till att bli stjärnor, andra blir ”hundar” beroende på hur utvecklingen ter sig (BCG, 2005). Därför bör intäkter från ”kassako”-produkter finansiera utveckling för dessa ”frågetecken” för att få fram nya ”stjärnor” och ”kassako”-produkter (Hedman & Kalling, 2002). Det gäller dock att vara aktsam och följa utvecklingen för dessa osäkra produkter för att de inte skall sluka stora investeringar och sedan inte leda fram till en vinstgivande produkt (BCG, 2005).

Denna modell kommer vi att relatera till i analysen. Vi anser denna modell vara tillämpbar på mjukvaruproduktion trots att den tagits fram för en generell marknad där tillverkningskostnad är en stor del av den totala kostnaden för en produkt. Som vi sett är inte tillverkningskostnaden en stor utgift för mjukvara där koden lätt kan kopieras och distribueras via exempelvis Internet men vi ser i stället andra investeringsbehov för en produkt under dess livscykel. Mjukvara kräver vidareutveckling av nya funktioner, felrättningar etc. för att ”hänga med” på marknaden vilket gör att investeringskostnader likväl förekommer för en mjukvaruprodukt efter att den blivit tillgänglig på marknaden.

En annan mycket känd modell för att ge företagsledningar ett sätt att mäta, bedöma och förbättra sin position är vad som kallas ”Balanced Scorecard”. Denna modell, utvecklad i början av 1990-talet av Robert Kaplan och David Norton, ger företag en möjlighet att jämföra och integrera finansiella aspekter med strategi och marknadsföring för företaget (Kaplan & Norton, 1992). Ett exempel på hur denna modell kan se ut visas nedan i figur 6.4:



Figur 6.4: *Balanced scorecard*

Vid användandet av ”Balanced scorecard” är det centrala att fyra grundfrågor ställs mot varandra:

- Finansiellt – Hur ser företaget ut för finansiärerna?
- Kund – Hur ser företaget ut för kunden?
- Internt – Hur hanterar företaget interna system och affärsprocesser?

- Förnyelse – Kan företaget ta till sig förbättringar, ny kunskap och förnyelse och växa på ett bra sätt?

Dessa grundfrågor ställs sedan mot varandra och används för att bedöma hur företaget ”mår”. Det eftersträvas då en balans mellan dessa fyra områden så att inte ett område driver iväg från de andra varpå framgången för företaget teoretiskt minskar. För att kunna bedöma var företaget står krävs emellertid mätbara värden. En viktig uppgift för företaget är därför att ta fram vilka värden man vill mäta samt strukturer och processer som underlättar insamling och analys av dessa mätbara värden (Balancedscorecard, 2004a). Exempel på sådana mätvärden kan vara räntabilitet och tillväxt (finansiellt perspektiv), kvalitet, service, pris och funktionalitet (kundperspektiv), personaltrivsel (internt perspektiv) samt tid för att utveckla nya produkter (förnyelseperspektiv) (Ljung et al, 2001).

Till denna teoretiska modell vill vi koppla en annan modell för att identifiera och mäta framgång för ett företag, nämligen idén om ”Critical Success Factors” (CSF:s). Teorin introducerades av Daniel på 60-talet och beskriver hur ett företag genom att identifiera nyckelfaktorer för sitt företag kan, genom att uppfylla mål knutna till dessa framgångsfaktorer, förbättra sina chanser till marknadsfördelar gentemot andra företag inom samma bransch (Daniel, 1961). Teorin populariserades av Rockart som i en studie nämner exempelvis utveckling av nya produkter samt en god distributionslösning som nyckelfaktorer för ett företag för att lyckas (Rockart, 1979). Rockarts exempel beskriver förvisso ett företag i livsmedelsbranschen, men torde likväl kunna vara relevant även för företag i andra branscher.

Ett annat perspektiv ur vilket man kan försöka förstå sig på fenomenet med öppen källkod är att applicera tankarna kring spelteori på ämnet. Nationalencyklopedin (2004a) ger en koncis beskrivning av spelteori genom: *”spelteori, ekonomisk-matematisk teoribildning som analyserar situationer i vilka två eller flera beslutsfattare interagerar.”* Låt oss även bekanta oss med några andra försök till sammanfattning av spelteori: *”Kortfattat kan spelteori beskrivas som en teori om under vilka förhållanden olika säljare på en marknad väljer att byta strategi för att sälja sina produkter och under vilka förhållanden jämvikt mellan olika leverantörer uppstår.” (Susning, 2004a).*

Ytterligare en vinkling ges i Ross (2004a) som beskriver spelteori enligt *“Game theory is the study of the ways in which strategic interactions among rational players produce outcomes with respect to the preferences (or utilities) of those players, none of which might have been intended by any of them”* och menar även på att paralleller till spelteori kan dras lång tillbaka i tiden till exempelvis Platon.

Det mest kända exemplet på spelteori torde vara fångarnas dilemma, där två fångar, eller om man så vill spelare, kan välja mellan att ange den andre eller att hålla tyst (McCain, 1997a; Ross, 2004a).

Ett närbesläktat exempel är spelet ”Chicken”, baserat på biltävlingsvarianten där två bilar kör emot varandra och där den som svänger först förlorar. Om ingen svänger förlorar givetvis båda förarna katastrofalt (GameTheory.net, 2004a; Rysdam, 2000). Tabell 6.1 nedan visar en så kallad payoff-matris för spelet Chicken.

Tabell 6.1: *Payoff-matris för "Chicken"*

A/B	Svänger	Svänger inte
Svänger	0	-1
Svänger inte	1	-10

Vi ska nu förklara tabell 6.1 ovan. Om båda förarna svänger blir ingen vinnare och båda får noll poäng, låt oss kalla detta fall 1. Om förare A håller sin kurs medan förare B svänger vinner A och förare A får ett poäng, fall 2. Skulle ingen av spelarna svänga kommer de att kollidera och resultatet blir klart negativt för båda vilket exemplifieras med minus tio poäng i tabellen ovan. Vi kan kalla det sista fallet för fall 3. Vi kan notera att fall 2 > fall 1 > fall 3 för att begagna oss av en kraftigt förenklad matematisk notation. För att få bästa möjliga gemensamma genomsnittsutfall bör de båda förarna samarbeta.

Rysdam (2000) har applicerat ett liknande resonemang på mjukvarulicenser för att jämföra olika typer av licenser med varandra. Resonemangen är dock inte oantastbara och har kritiserats av bland annat Goldman (2000), något som vi får anledning att återkomma till längre fram.

Vi kommer senare i analysen att utnyttja tankarna kring spelteori för att bättre förstå förutsättningarna för Open Source på marknaden.

6.3.3 Affärsmodeller i praktiken

Det finns ett oräkneligt antal exempel på praktiska affärsmodeller beskrivna. Vi gör här ett försök att belysa de modeller vi anser vara viktigast för vår frågeställning. Vi anser dem vara viktigast för att de speglar det moderna samhället och därmed också den verklighet mjukvarubranschen (som ju är målet för vår undersökning) befinner sig i. I takt med inträdet för det som kallats den "nya" ekonomin visar bland annat Rappa (2004a) att användandet av Internet för att göra affärer har gett och kommer att ge upphov till nya sätt att göra affärer för företag och specifikt företag som arbetar med elektronisk handel mellan företag och kund (gärna förkortat B2C, "Business-to-Customer") eller mellan två olika företag (förkortat B2B, Business-to-Business"). Eftersom mjukvarubranschen mer och mer knyts till möjligheterna till elektronisk handel och kommers via Internet härrör sig många av våra exempel nedan från denna värld. En stor del av våra exempel hämtas från Rappa (2004a) som bland annat även Hedman och Kalling (2002) refererar till.

Att prenumerera på en vara eller tjänst är en metod för ett företag att tjäna pengar. Det kan handla om att prenumerera på en tidning, böcker, skivor, TV-kanaler, Internettjänster, mjukvara med mera (Rappa, 2004a).

Ett sätt att tjäna pengar som också ligger nära andra branscher än mjukvara är sättet att tjäna pengar genom att ta betalt för en produkt och därmed också överföra äganderätten av produkten till den köpande parten (Rappa, 2004a). Rappa kallar denna affärsmodell ”Tillverkar”- eller ”Direkt”-modellen (vår översättning). Det kan här även handla om licensanvändning där äganderätten inte helt övergår till den köpande parten eller helt enkelt en form av ”leasing” för att använda ett vedertaget begrepp i Sverige. Exempel på företag som använder sig av denna affärsmodell är Dell, Microsoft med flera (Rappa, 2004a).

En del företag tjänar pengar genom annonsering, vad gäller mjukvara och Internet kan detta ta sig uttryck i exempelvis annonser med klickbara länkar på hemsidor (exempelvis sökmotorn ”Google”), eller annonsering av varor till salu där försäljaren måste betala för att få lista sina varor (Rappa, 2004a).

Det finns också företag som använder en mer renodlad traditionell köpmansroll som affärsmetod. Exempel på sådana företag, som ofta kan sälja sina varor direkt på Internet, är Amazon och företag som säljer musik och liknande (Rappa, 2004a).

Att tjäna pengar på köp och försäljning kan också göras genom att driva auktioner för att förmedla varor mellan köpare och säljare, exempelvis som eBay (www.ebay.com) (Rappa, 2004a).

När det gäller köp och försäljning går det även att fokusera på nätverkseffekter som det centrala i sin modell för att tjäna pengar. Ett exempel skulle även här kunna vara eBay, som är beroende av ett nätverk av köpare för att få produkter till sina auktioner sålda (Irvine, 2004a).

Genom att använda nätverk kan företag också sikta på att minska distributionskostnaderna, som Porter (1985) beskriver som en kostnadsdrivare (egenskap eller hantering som kostar pengar eller resurser för företaget), för att på så sätt tjäna pengar. Internet är också ett exempel på ett sådant nätverk som minskar distributionskostnaden till i det närmaste noll för de företag som använder sig av detta. Kotler (1999) beskriver också minskade distributionskostnader som ett sätt att uppnå konkurrensfördelar. Vi har tidigare gjort jämförelser mellan mjukvarubranschen och musik- och filmindustrin i detta avseende.

En form av nätverk är också den modell som är mest central i vår studie, nämligen gemenskapsmodellen, i vårt fall exemplifierad av Open Source. Rappa (2004a) beskriver basen för denna modell som användarmedverkan, och pengar kan enligt Rappa tjänas genom försäljning av tilläggsprodukter eller tjänster. Vi skall analysera detta mer i detalj senare.

Vid en jämförelse mellan Rappas (2004a) samt Hedman och Kallings (2002) beskrivningar av affärsmodeller finns det element som saknas hos Rappa. Detta beskriver också Hedman och Kalling som ett av problemen med beskrivningar av affärsmodeller för elektronisk handel; de tar bara hänsyn till en enkel eller valda delar av den totala affärsmodellen. De nya händelser och aktiviteter som Internet medför för dagens företag är enligt Hedman och Kalling ett sätt att integrera nya

affärsmetoder till existerande affärsmodeller, det finns alltså enligt dem inget behov av att utveckla nya affärsmodeller. Som exempel ger Hedman och Kalling ett företag (Barnes & Noble) som öppnat en Internetbutik för att konkurrera med företaget Amazon (som säljer exempelvis böcker och filmer på Internet). Detta blir då en ny säljkanal för företaget som ett tillägg till existerande kanaler men inte en helt ny affärsmodell.

6.4 Open Source som strategi

Vad har Open Source med strategi att göra? Hamel och Prahalad (1996) gör följande notering som kan vara oss behjälplig: *"The organizational transformation challenge faced by so many companies today is, in many cases, the direct result of their failure to reinvent their industries and regenerate their core strategies a decade or more ago"* (Hamel & Prahalad, 1996 sid. 19).

Hamel och Prahalad (1996) lyfter bland annat upp IBM som ett exempel på ett företag som länge levde på gamla meriter för att sedan bli omsprungna av nya konkurrenter såsom Microsoft och Dell. IBM är numera en stark sponsor av flera "Open Source"-projekt såsom Eclipse, Apache och Linux. Eclipse är extra intressant eftersom det började sitt liv som ett internt projekt inom IBM där resultatet senare "donerades" som öppen källkod (IBM, 2001a). En variant av Eclipse innehåller bland annat en utvecklingsmiljö för Java och C++. Marknadsledande för utvecklingsverktyg på PC-sidan är Microsoft med sin produktfamilj Visual Studio. Den enda någorlunda stora konkurrenten är Borland, men de ligger mycket långt efter Microsoft vad gäller marknadsandelar. Det är därför intressant att notera att IBM gav ut Eclipse som var deras nya plattform för utvecklingsverktyg fritt som Open Source och bjöd in andra, inklusive sina konkurrenter, att hjälpa till att vidareutveckla och bygga nya lösningar på plattformen. Olika kommersiella aktörer kan sedan anpassa plattformen och göra egna tillägg och på så vis specialisera och differentiera sina erbjudanden. Antagandet att IBM genom sin "Open Source"-modell för Eclipse har gjort ett aktivt försök att "uppfinna" en ny affärsstrategi, utöver de förmodade "godwill"-förtjänsterna, borde inte ligga speciellt långt borta.

Hamel och Prahalad (1996) visar att Dell ej kunde matcha Compaqs återförsäljarnät eller IBM:s armé av säljare. Istället började Dell sälja via postorder och senare via Internet. Hamel och Prahalad (1996) tangerar här O'Reilly (2004a) och vice versa vilket vi snart kommer att se.

Dells framgångar skapade nya förutsättningar för leverantörerna av PC-utrustning. Enligt Kotler (1999) tenderar äganderätts- eller "copyright"-produkter att dyka upp i intilliggande steg i värdekedjan när ett annat steg blir modulariserat, exempelvis som Intels intåg på PC-marknaden för processorer när andra företag som Dell började bygga ihop datorerna billigt. Vi skall ju visa hur man kan tjäna pengar på Open Source, men det behöver ju inte betyda att man inte kan tjäna pengar även om det inte är Open Source, bara att fokus kan flyttas efterhand som ett visst marknadssegment blir mättat eller "mognar".

Hamel och Prahalad (1996) lyfter fram ett synsätt för affärsstrategi där de starkt avråder från att hålla fast vid gamla bilder av vilka gränser som gäller och uppmanar företag ett uppfinna framtida strukturer för sina branscher, vilket även påvisas av O'Reilly (2004a) i en artikel på Internet. O´Reilly visar att en parallell kan dras från hur IBM tappade marknadsandelar i hårdvaruförsäljningen till dagens mjukvaruindustri där många företag arbetar enligt de "gamla" reglerna. Nya företag drar nytta av de förändringar som skett i mjukvaruindustrin och kan utnyttja förändringarna till att skapa sig marknadsfördelar helt enkelt genom att förstå dessa förändringar och bygga nya regler och verksamheter kring dessa. Vi ser här alltså exempel på företag som lyckats bättre (Dell, Microsoft) respektive sämre (IBM) i sin förändringsprocess enligt Morgans tankar om lärande organisationer samt var denna förändringsprocess idag befinner sig för företag som arbetar med mjukvaruutveckling relaterat till Open Source.

Hamel och Prahalad (1996) har en syn på strategi *som "…inser att konkurrens ofta sker inom och mellan koalitioner av företag och inte bara mellan enstaka företag"* (Hamel & Prahalad, 1996 sid. 25) (vår översättning). Vid utveckling av "Open Source"-program kan flera företag på samma gång samarbeta och konkurrera. Detta ger en bild av på vilka möjligheter och svårigheter som finns.

Att företag samarbetar samtidigt som de befinner sig i konkurrens är dock inget nytt, vi har sett Hedman och Kalling (2002) exemplifiera detta. Företag kan även, för att ge ett ytterligare exempel, samarbeta för att ta fram ny standard inom sitt område och därefter konkurrera med produkter som implementerar standarden. Ett sådant scenario är inte helt olikt de situationer som Open Source kan ge upphov till.

6.5 Trender

"Like dinosaurs threatened by cataclysmic climatic changes, companies often find it impossible to cope with a radically altered environment." (Hamel och Prahalad 1996 sid. 53)

I den nya globala världsekonomin gäller det för företag och organisationer att "lära sig att glömma". Saker och ting görs och fungerar inte längre som förr.

Philip Kotler beskriver i sin bok "Kotlers marknadsföring : Att Skapa, Vinna Och Dominera Marknader" (Kotler, 1999) bland annat detta fenomen Då förändringstakten på marknaden ökar kan företag inte längre använda sina gamla affärsmetoder för att behålla sin lönsamhet (Kotler, 1999). Det gäller med andra ord att kunna förändra sitt tankesätt snabbt och på rätt sätt för att kunna överleva som företag.

Comment [f3]: Sid???

Att snabbt kunna förändra sitt företag är något som även Gareth Morgan pekar på i sin bok "Organisationsmetaforer" (Morgan, 1997). Han beskriver där olika sätt att se på ett företag med liknelser av olika slag. En av dessa liknelser eller metaforer är att se företaget som en hjärna med dess möjlighet till lärande. Det är då enligt Morgan

viktigt för företaget att kunna bli medveten om problem i omgivningen för att snabbt kunna komma fram till en lösning på dessa problem. Detta kan uppnås bland annat genom inbyggda incitament för företaget och dess medarbetare att genomsöka omgivningen efter förändringar för att upptäcka nya variationer och trender, genom att kunna ifrågasätta invanda arbetssätt och strategier och genom att anpassa organisationens struktur så att den snabbt kan byta strategi eller inriktning för att möta upp förändringar i omgivningen. Morgan skriver om dessa lärande organisationer att *"Det är genom att på ett nytt sätt uppfatta och tänka över det sammanhang deras verksamhet ingår i som de kan föreställa sig och skapa nya möjligheter"* (Morgan, 1997 sid. 103). Som vi skall se framöver finns det exempel på företag som lyckats bättre och sämre med detta.

I denna förändringsprocess kan en jämförelse mellan traditionell tillverkande industri och mer tjänsteinriktade företag ställas upp. Några framstående trender i dagens globaliserade värld är enligt Kotler (1999) att:

- Tillverkning flyttas till låglöneländer
- Företag ändrar fokus från produktion till förädling av tjänster och specialisering i avancerad teknik

Några av de utmaningar som företag möter idag är att hålla nere produktionskostnader och att specialisera sig alltmer exempelvis inom forskning, produktion eller inom marknadsförning och varumärkesbyggande.

Kotler poängterar och belyser ytterligare några trender genom en jämförelse mellan traditionella sätt att arbeta för företag och vad han menar är den nya tidens mer förändringsinriktade verksamhet. Det har bland annat blivit vanligare att köpa in fler produkter och tjänster från andra leverantörer samt att koncentrera sig på att använda få leverantörer (som ju i sin tur kan ha andra underleverantörer). Om man vill följa vad som anses som nutida affärsmetoder bör man enligt Kotler (1999) också inrikta sig på hela värdekedjan istället för enstaka produkter.

En trend som Bassellier och Benbasat (2004) noterar är att dagens IT-specialister behöver anpassa sig till en ny situation där samarbete är nyckeln till nyskapande och framgång. Utöver de rent tekniska kunskaperna behöver en IT-specialist även kunskaper om branscher och affärsrelationer för att kunna jobba närmare sina kunder.

6.5.1 Den "nya" affärsrollen

Företag idag ställs alltså inför nya utmaningar med nya krav från kunder och leverantörer. Tim O'Reilly (2004a) resonerar vidare kring detta fenomen då han bland annat skriver att det inte är tillräckligt med ett starkt varunamn i dag för att bli ett lyckat mjukvaruföretag, det krävs mer för att uppnå lönsamhet. Han tar bland annat upp företaget Flextronix, en industrileverantör av komponenter, som ett

exempel på hur man kan leverera komponenter som tillsammans med andra komponenter bildar en helhet istället för att leverera färdiga lösningar direkt mot kund. O'Reilly (2004a) menar att detta kanske kan vara något för företag inom Linuxbranschen, där marknaden för själva operativsystemet inte är tillräckligt stor för ett stort antal företag. Slutsatsen som O'Reilly drar är att mjukvara för sig själv inte längre är det primära värdet för datorindustrin, hela värdekedjan (som ju även Kotler (1999) tar upp som ett "modernt" fenomen) med tjänster som sammankopplas med mjukvaran blir viktig.

Phipps (2004a) erbjuder ytterligare en vinkling till frågan om hur dagens IT-företag kan nå lönsamhet när han jämför SUNs Java Desktop System (JDS) med en tidning för att försöka lösa paradoxen hur SUN ska kunna tjäna pengar på något som är gratis. JDS består av olika program där de flesta är Open Source och gratis tillgängliga. Phipps noterar att även nyheter är gratis tillgängliga via exempelvis Internet, men att tidningar fortfarande köps. Det som kunden får när hon prenumererar på SUNs produkter är en layout och en redaktörsfunktion och SUN kan ses som en slags koordinator eller förmedlare. Phipps menar vidare att SUN skapar och förbättrar mjukvara som de sedan sätter samman och (åter)använder i olika sammanhang och konstellationer. Författaren framhåller att SUN har skapat nya marknadsplatser där SUN och andra kan lyckas, allt enligt devisen *"a rising tide lifts all boats"* (Phipps, 2004a).

6.6 Övriga aktörer på marknaden

Vi har nu beskrivit marknaden i första hand ur företagen som utvecklar mjukvara och olika underleverantörers perspektiv. Låt oss så även beskriva två ytterligare aktörer: köparen samt utvecklaren.

6.6.1 Köparen

Hur ser köparen av en produkt eller tjänst på marknaden? En indikation på detta får vi om vi studerar statskontorets rapport som vi tidigare refererat till (Statskontoret, 2003a). Det konstateras i denna att ur ett mer ekonomiskt perspektiv har öppen källkod ett antal positiva effekter vid inköp av programvara:

- *inga eller låga licenskostnader*
- *minskat beroende av en produkt eller leverantör, mindre risk för inlåsning*
- *sänkta totala kostnader*
- *ökad konkurrens*

- *ökad kvalitet och stabilitet*
- *ökad stimulans av lokal/inhemsk företagsamhet*
- *ökad säkerhet*
- *öppna format förenklar kommunikation med allmänheten*

Dessa faktorer ger vid handen att ekonomiska incitament finns för köpare att använda produkter baserade på FOSS. Som konstaterats i kapitlet som beskriver utvecklingsmodeller gäller det dock att minimera nackdelarna med dessa produkter som kan betyda höga kostnader, exempelvis omfattande migrationsarbeten eller interoperabilitetsproblem med annan proprietär mjukvara. Ett typexempel på det senare är möjligheterna att använda textdokument skrivna i Microsofts Officepaket (Word, Excel med flera) med Officepaket baserade på FOSS som exempelvis StarOffice. Samtliga dokument måste då konverteras för att kunna användas med det nya programmet (Görling, 2003a). För en fullständig redogörelse över köparens aspekter hänvisar vi enligt vår avgränsning till andra undersökningar som gjorts kring begrepp som TCO och liknande (Wheeler, 2004; Cybersource, 2002).

6.6.2 Utvecklaren

För att det över huvud taget skall bli någon programutveckling måste det finnas utvecklare som skriver kod. Lerner och Tirole (2000) beskriver förutsättningarna för en utvecklare att engagera sig i programutveckling utifrån utsikten till någon form av personlig belöning för den enskilde individen. Det finns enligt dem två former av belöningar; direkta och långsiktiga. Direkta belöningar kan vara exempelvis ekonomisk ersättning eller den personliga tillfredsställelsen av att hitta och lösa ett programfel. Mer långsiktiga belöningar kan vara att andra personer eller företag uppmärksammar ens arbete som därigenom blir en sorts ”marknadsföring” av den egna personen vilket kanske kan leda till ett bättre jobb så småningom med exempelvis hög lön, intressanta arbetsuppgifter eller andra förmåner (Lerner & Tirole, 2000). Det kan också handla om personlig tillfredsställelse i form av att andra personer uppskattar ens arbete, eller en form av ”kändisskap” om man så vill, eller känslan av ha bidragit till utvecklingen av en bra produkt eller mjukvara vilket Raymond beskriver som en del av den tidiga ”hackerkulturen” (Raymond, 2001a).

Dessa olika typer av belöningar får mer eller mindre uttryck i katedral- och basarmodellerna som beskrivits i kapitlet om utvecklingsmodeller. I katedralmodellen är den direkta belöningen oftast ekonomisk ersättning till utvecklaren, men detta kan även vara fallet för utveckling av FOSS om utvecklaren är anställd av ett kommersiellt företag (Lerner & Tirole, 2000). Tillfredsställelsen av att hitta och lösa programfel kan finnas för båda utvecklingsmodellerna, men belöningen av att andra uppmärksammar det har hittills varit reserverad för FOSS-rörelsen. Lerner och Tirole menar att en anledning till att företag som utvecklar enligt katedralmodellen

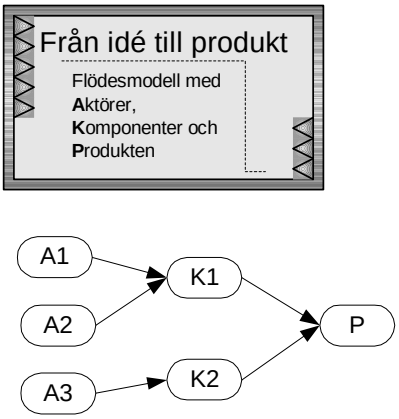
”hemlighåller” sina utvecklare kan vara att de inte vill att andra företag skall locka till sig sina programmerare (exemplifierat av Lerner & Tirole genom företaget Apple).

6.7 Samspel mellan marknadens aktörer

Vi har nu visat vad marknaden är och aktörsroller som finns på denna marknad. Låt oss så visa hur samspelet mellan olika aktörer som exempelvis utvecklare, distributörer och slutanvändare kan se ut.

6.7.1 Utvecklingsmodell

Vi vill först visa på en enkel, generell modell för utveckling av en produkt för att tydliggöra sambanden mellan delarna av det vi tidigare definierat som ”marknaden”.



Figur 6.5: *Utvecklingsmodell.*

Enligt modellen i figur 6.5 ovan samverkar ett antal aktörer (A) med hjälp av ett antal komponenter (K), egenutvecklade eller framtagna av andra aktörer, till att skapa en produkt (P). Modellen är fritt konstruerad av oss enligt beskrivning av mjukvaruutveckling från exempelvis Raymond(2001a) och passar in på utveckling enligt exempelvis Microsofts SSI, Raymonds katedral- och basarmodeller med flera.

Den slutliga produkten skall sedan användas av en slutanvändare. Denna slutanvändare kan antingen finnas i den egna organisationen eller så behöver produkten distribueras till slutanvändaren. Detta kan antingen göras av den egna organisationen eller av en eller flera distributörer (Kotler, 1999).

Ser man utvecklingsmodellen i figur 6.5 mot ”Open Source”-konceptet som bakgrund kan man tänka sig två olika drivkrafter eller sätt att styra processen.

För det första kan det tänkas att en användare tar del av rollen som utvecklare, alltså börjar ta fram en eller flera komponenter för att till sist leda fram till en produkt slutanvändaren ser ett behov av. Raymond (2001a) påpekar bland annat, med Linux som exempel, att många mjukvaruprodukter baserade på FOSS kommer fram som en följd av någon utvecklares personliga behov. När ett första utkast till program släpps till allmänheten är chansen då stor att någon annan utvecklare har identifierat samma behov och kan hjälpa till att utveckla programmet vidare, varpå maskineriet har satts igång och förhoppningsvis fler och fler utvecklare hakar på projektet. Vinsten med Open Source i det här fallet är att användaren kan få hjälp av flera aktörer att ta fram de komponenter som behövs till en färdig slutprodukt (Raymond, 2001a). Eftersom det finns flera aktörer som hjälps åt, finns också chansen att slutprodukten kan innehålla funktioner eller möjligheter som ursprungsutvecklaren inte tänkt på att utveckla eller kanske inte velat driva på egen hand. Ett typexempel på detta är, för att återigen använda Linux som exempel, hur Andrew Tanenbaum först utvecklade Minix som sedan kom att inspirera Linus Thorvalds till att starta projektet med att ta fram Linux (Raymond, 2001a; Rosenberg, 2000).

Ett andra sätt att tänka sig utvecklingen enligt detta koncept är att slutanvändaren driver fram utvecklingen genom att engagera en eller flera utvecklare till att ta fram en slutprodukt enligt sin kravspecifikation. Ett exempel på arbete enligt denna metod beskriver Mårten Mickos, VD för företaget MySQL i en intervju för tidningen Ny Teknik: *”Gå till den första kunden som är beredd att betala och gör det han är beredd att betala för”* (Ny Teknik, 2004a). Fördelen här är för företag att intäkter är garanterade genom konsultarvoden från kunden, nackdelen för utvecklaren (enligt filosofin bakom FOSS) att den enskilde programmeraren eller utvecklaren inte får den frihet att utveckla det han eller hon vill. Detta skall då ställas i relation till den kompensation i form av lön eller andra ekonomiska ersättningar utvecklaren får, men i detta avseende har vi då närmat oss katedralmodellen för utveckling sett ur utvecklarens perspektiv.

Katedralmodellen ter sig enklare att beskriva ur ett rent ekonomiskt perspektiv. Enligt modellen ovan tar ett mjukvaruutvecklande företag fram en idé till en produkt (antingen företaget självt eller genom specifikation från kund). Produkten utvecklas sedan genom att använda någon utvecklingsmodell varpå den säljs till en kund. Intäkterna från försäljningen skall då överstiga kostnaderna för att utveckla produkten (Lerner & Tirole, 2000).

6.8 Sammanfattning

Affärsmodeller för traditionell tillverkande industri har sedan länge betraktats som förlegade och inadekvata för dagens tjänstemarknader och nya ekonomier. Vi har med utgångspunkt i äldre litteratur belyst utveckling för ekonomisk teori i ämnet och gett exempel på affärsmodeller och affärsmetoder som de ser ut i dag. Utifrån detta har vi vidare sett hur Open Source kan användas som en strategi i ”den nya ekonomin”. Slutligen har vi överskådligt visat köparens perspektiv på den marknad vi

beskrivit, hur den enskilde programutvecklaren kan uppfatta sin roll i sammanhanget samt dessa aktörers samverkan på marknaden.

7 Analys och sammanfattning

Invading armies can be resisted, but not an idea whose time has come. – Victor Hugo, ur boken 'Histoire d'un crime,' 1852 citat från http://www.quotationspage.com/quote/3240.html

Fri programvara och öppen källkod har mötts av en viss skepticism där antagonisterna inte har hesiterat för att ta till ren skrämselpropaganda. FOSS-programmens förfäktare har emellertid en inte alldeles trivial uppgift att förklara hur fri programvara och öppen källkod är tänkt att fungera, speciellt om vi har i åtanke att vid befinner oss i en kapitalistisk värld.

I detta kapitel ämnar vi försöka knyta tillbaka till våra ursprungliga frågeställningar i olika former av analyser som vi till slut kommer att foga samman i en syntes. Låt oss dock först återkoppla till vår inledande forskningsansats.

7.1 Frågeställningen

Vi har nu redovisat ett antal teorier och faktorer som relaterar till och kan belysa koncepten Free Software och Open Source. För att återknyta till de grundfrågeställningar vi ställde upp i upptakten av denna uppsats repeterar vi frågorna: ”Hur kan pengar tjänas på fri programvara och öppen källkod?”, eller annorlunda formulerat, ”Hur kan man tjäna pengar på något som är gratis?”. Vi ville också se om vi kunde finna nya vägar till att generera intäkter av programvara som är ”gratis”.

Vår skenbart enkla frågeställning rymmer dock flera bottnar. Order ”hur” har vi valt att tolka på två sätt, dels i meningen ”På vilka sätt…” och dels i betydelsen ”Hur kommer det sig att…”. Möjligen skulle vi kunna kalla det ett yttre och ett inre perspektiv på frågan och i slutändan också ämnet för vår diskurs. I slutändan är det ju, trots den fina tanken med fri kod som kan delas av alla, så att alla entusiaster till trots som bidrar med utveckling på fritiden så kvarstår faktum: För att en produkt skall bli utbredd på marknaden och kunna fortleva så måste (med några få undantag) i den kommersiella världen ett (ekonomiskt) vinstintresse föreligga hos den utvecklande parten. Detta ställs då mot tanken inom FOSS-sfären att utveckla program som är ”till allas nytta” och där alla kan bidra med lite för att få något som alla har nytta av. Att utvecklare engagerar sig inom FOSS-rörelsen kan säkert även härledas till andra faktorer såsom sociala och personliga drivkrafter. De som jobbar med FOSS på sin fritid arbetar möjligen med det de vill och är bra på snarare än det blir tillsagda att göra på sitt ”dagtidsjobb”.

Det förra perspektivet, som vi kan kalla perspektiv 0, syftar vi att bearbeta genom en diskussion knuten till ekonomisk teori vilken vi har presenterat i kapitel 6 ”Marknaden och FOSS” vilken också kommer att kompletteras med en modell för ett marknadsscenario.

Det senare perspektivet, vilket vi också benämner som perspektiv 1, kommer vi att bearbeta med en mer diversifierad verktygslåda i förhoppning om att lösa paradoxen med ”hur det kommer sig att man kan ta betalt för och tjäna pengar på något som är gratis”.

7.2 Perspektiv 0

I detta första perspektiv, perspektivet 0, avser vi att behandla ”på vilka sätt” olika aktörer kan profitera pengar på fri programvara och öppen källkod. Diskussionen i det här avsnittet knyter i första hand an till de ekonomiska teorier och modeller för marknaden vilka presenterades i kapitel 6 ”Marknaden och FOSS”.

7.2.1 I en kommersiell värld

Många argument för och emot ”Open Source”-utveckling är traditionellt givna ur ett utvecklarperspektiv med den gemensamma nyttan/värdet för användaren (som i det här fallet ofta är andra utvecklare) i fokus. Detta vore ju trevligt i en perfekt värld, men det finns naturligtvis även andra aspekter som styr händelseutvecklingen. Den mest givna och centrala av dessa aspekter är ju naturligtvis den ekonomiska. Få saker produceras i världen utan att någon har något att tjäna på det, så också vad gäller mjukvaruutveckling. Det är denna aspekt mycket av debatten kring FOSS har kommit att handla om, framför allt när man studerar de artiklar som skrivits för och emot Microsoft/Öppen källkod på senare år. I denna debatt finns å ena sidan Microsoft, dess förespråkare och likasinnade som enligt vad vi tidigare redovisat hävdar att det krävs en organisation som kan överleva genom att generera intäkter från såld programvara för att kunna säkerställa en produkts fortlevnad, underhålla produkten under dess livslängd och ta fram nya och användaranpassade tilläggstjänster till produkten. Å andra sidan finns ”Open Source”-kulturen, uppbackad av organisationer som FSF och OSI, samt ett antal andra personer och organisationer av större eller mindre dignitet. Denna sida hävdar istället att hela systemet med licenskostnader och avlöningssystem till programmerare som tar fram produkter hämmar kreativiteten och möjligheten för användaren att få ut det bästa ur produkterna. Hur menar då ”Open Source”-rörelsen att någon skall kunna tjäna pengar på mjukvaruutveckling?

Detta är naturligtvis inte en trivial fråga att svara på, det finns många åsikter i ämnet. Ett grundläggande krav för att kunna identifiera lönsamhet är att ha någon form av gemensamma, mätbara begrepp för olika företag inom olika branscher. Det mest uppenbara av dessa begrepp anser vi torde vara vinsten i pengar räknat. Det finns nu

emellertid ett antal sätt att hantera bokföring för dagens företag där vinsten räknat i pengar inte alltid avspeglar hela sanningen. Detta faktum, tillsammans med vårt angreppssätt till denna undersökning, gör att vi försöker se lite vidare än så och se till helheten över hur företag lyckats och kan lyckas överleva baserat på fenomenen fri programvara och öppen källkod, eftersom förmågan att överleva och utvecklas som företag också kan vara ett sätt att mäta framgång. Otvivelaktigt behöver vi ändå ett antal begrepp och teorier för att förklara fenomenet Open Source och dess framtida plats i vår värld. Med hjälp av de modeller och den litteratur vi presenterat i tidigare kapitel skall vi nu göra ett försök att reda ut vår frågeställning.

7.2.2 Inte all programvara FOSS

Enligt Raymond (2001a) lämpar sig utvecklingsmodellen för Open Source inte för alla mjukvaruutvecklingsprojekt. Open Source lämpar sig enligt Raymond bäst för projekt med antingen få utvecklare (varigenom konfliktsituationer mellan utvecklare som vill ta olika vägar undviks) eller aktuella projekt med stort allmänintresse där många är intresserade av att snabbt få fram en användbar produkt. Ju mer generell en produkt är från början desto bättre, eftersom det alltid är lättare att anpassa en generell produkt till ett speciellt ändamål än att generalisera en specifik produkt.

Som en bieffekt till dessa kommersiella projekt fås också fördelar för privata användare som inte behöver skriva alla program själv eller alternativt köpa dyr programvara, utan dessa användare kan istället ladda ned FOSS-programvara och sedan eventuellt modifiera den för sina egna specifika behov. Det är då naturligtvis viktigt att beakta licensvillkor och allmänt ”hyfs” inom rörelsen, gör man en intressant anpassning skall man naturligtvis dela med sig av den. Huruvida detta är avtalsmässigt tvingande att dela med sig eller ej beror på de aktuella licensvillkoren.

Utvecklingsmetoden lämpar sig då sämre för motsatsen till ovan nämnda typer av projekt, alltså i typexemplet väldigt specifika projekt med många utvecklare som är strikt styrda mot slutmålet. Huvudargumenten mot FOSS-utveckling är i ett sådant fall dels att utvecklingen lätt kan ”driva iväg” åt olika håll, eller som vi tidigare noterat ”forka” för att ta den engelska termen, om utvecklarna börjar dra åt olika håll och dels att allmänintresset för den producerade koden är lågt.

Med detta sagt, låt oss alltså först och främst fastslå att företag som inte använder sig av FOSS-utveckling, exempelvis Microsoft, har sin plats i mjukvaruvärlden. Ingen kan på goda grunder påstå att Microsoft inte är ett framgångsrikt vinstgivande företag. Logotypen och företagets produkter är kända över hela världen. Det finns flera exempel på företag med samma affärsmodell, men Microsoft är det mest kända och dominerande. Affärsmodellen, det Rappa (2004a) kallar ”Tillverkarmodellen”, har visat sig fungera för flera företag i olika branscher i många år och går alltså i stort sett för Microsofts del ut på att producera kod som sedan genererar intäkter genom att licenser säljs till användaren som ger denne rätt att använda koden. Så länge intäkterna överstiger utgifterna är allting frid och fröjd, men det finns naturligtvis saker som påverkar företagets intäkter såväl som utgifter. Bostonmatrisens regler om ”stjärnor” och ”hundar” gäller även här, i synnerhet för produkter i dagens IT-värld

där utvecklingen går mycket snabbt. Mjukvarubranschen tillför dessutom ytterligare en dimension till Bostonmatrisen genom att det är mycket lättare att skapa och distribuera tilläggsprodukter och nya versioner av datorprogram än vad det är att till exempel förändra tillverkningen av en bilmodell, ett verktyg eller liknande även om det naturligtvis kan ligga omfattande utvecklingsarbete bakom samtliga dessa fall.

Varför gör då inte alla företag som Microsoft eller andra med samma affärsmodell? Ja, ett förenklat svar vi kan ge är att det ju tack och lov finns olika synsätt på hur saker och ting bör fungera här i världen (samtidigt som företag med stor marknadsandel naturligtvis gör sitt bästa för att hålla konkurrenter på behörigt avstånd från sig). Det finns mer komplexa svar på denna fråga som till viss del hör hemma för djupt nere i ekonomins värld för att rymmas inom ramen för vår frågeställning, men likväl till viss del kommer att besvaras av vårt nästa avsnitt där vi analyserar företeelsen FOSS ur ett annat perspektiv. Med vår huvudsakliga frågeställning för denna uppsats och perspektivet hur företag kan profitera på FOSS i åtanke, samt baserat på de exempel vi tidigare refererat till, nöjer vi oss här med att konstatera att det finns företag som faktiskt lyckats tjäna pengar på företeelsen Open Source (IBM, Red Hat, Netscape etc.). Hur har de då lyckats med detta?

För att svara på den frågan ser vi oss tvingade att göra en viss uppdelning, återigen baserat på syftet med denna uppsats. Vi ser dels vad vi har benämnt som ekonomiska intressen för företag och dels bakomliggande mekanismer som också främjar utveckling enligt ”Open Source”-konceptet. Låt oss börja med den ekonomiska frågan.

7.2.3 Ekonomiska aspekter på FOSS utvecklingsmodell relaterat till Closed Source

De affärsmodeller som står till buds för dagens företag är som tidigare nämnts otaliga. I takt med intåget för Internet i affärsvärlden och ombildandet av världsekonomin till den nya ekonomi som vi sett bland annat Kotler (1999), Hedman och Kalling (2002) samt Rappa (2004a) diskutera blir dock vissa faktorer viktigare än andra vilket också styr vilka affärsmodeller som blir intressanta för mjukvaruindustrin.

Kotler (1999) säger att det är viktigt att ta hänsyn till hela värdekedjan, O´Reilly (2001a) menar att det inte räcker med ett starkt varumärke för att bli ett lönsamt mjukvaruföretag idag. I värdekedjan ingår ju enligt Porter (1985) flera steg för att förädla en produkt eller tjänst innan slutleverans till kund. Det gäller då, för att kapa kostnader, att hitta smidiga sätt att göra saker bättre än de gjorts förut och i rätt led i värdekedjan. Givet Hedman och Kallings (2002) definition av en affärsmodell kan vi också dra slutsatsen att det även är viktigt att ta hänsyn till andra saker än det egna företagets interna processer och kunder, det finns även en omvärldsfaktor med externa marknads- och kundprocesser samt en nätverkseffekt människor och företag emellan som ju bland annat Dahlander (2004) beskrivit.

I fallet med mjukvaruproduktion är, som vi har sett, ett av sätten att spara pengar att förbättra distributionen av mjukvara. Leverans av färdig produkt kan ske genom distribution av CD med programvara, vilket exempelvis Red Hat använt sig av, eller till och med genom direkt nedladdning via Internet. Fallet Red Hat exemplifierar en blandning av några av de olika affärsmetoder vi tidigare presenterat (exempelvis genom Rappa, 2004a) för sin affärsmodell; En nätverksmodell som utnyttjas för distribution, en köpmansroll för att förmedla produkter, gemenskapsmodellen då Red Hats intäkter till stor del kommer från tjänsten att paketera mjukvara någon annan i stor utsträckning producerat (även om Red Hat också bidragit till mycket av Linux utveckling) samt tjänsten support. Licensfrågan klaras genom att inte primärt ta betalt för mjukvaran utan för nämnda tjänster. Redan här har vi alltså visat ett sätt att tjäna pengar på Open Source, baserat på olika affärsmodeller. Det finns dock fler exempel.

Google (http://www.google.com) är ett sådant exempel som säkert de flesta känner till. Baserat på FOSS, använder detta företag affärsmetoden annonsering via sin sökmotor för att generera intäkter. Fler exempel som nyttjar annonseringsmetoden är tidningar, tidskrifter eller andra publikationer som exempelvis kan husera artiklar, föreläsningsmaterial eller rapporter om Open Source. Affärsmetoden i detta fall är prenumeration som innebär att den som vill läsa de fullständiga publikationerna måste erlägga en avgift varefter publikationen i fråga kan laddas ned direkt eller skickas hem till prenumeranten. Notabelt i detta fall är att för många inom FOSS-rörelsen är det kutym att fritt offentliggöra sina alster (exempelvis Raymond och Stallman har publicerat många av sina texter på Internet) men det finns ju inget inom licensområdet som påvisar att detta är ett krav för ren text utan det står ju varje författare eller förläggare fritt att göra som han eller hon vill. Som en bisats kan vi även nämna att det finns flera licenser för dokumentation (av fri och öppen programvara), exempelvis GNU Free Documentation License, FreeBSD Documentation License och Open Publication License (Olofsson, 2003a).

Detta är några exempel på företag som lyckats tjäna pengar på Open Source. Vi vill emellertid lyfta diskussionen en nivå högre än att bara titta på exempel på hur företag gjort, vi försöker också analysera situationen av idag och hur den skulle kunna tänkas forma sig till nya lösningar.

Dagens marknadsläge för Open Source ter sig vidare än någonsin. Projekt utvecklade enligt FOSS-konceptet blir bara fler och fler till antalet (se exempelvis http://www.sourceforge.net) och det finns så vitt vi kan se inget i dagsläget som tyder på att intresset för dessa projekt skulle minska, snarare tvärtom ju mer utrymme grundtankarna bakom FOSS får bland utvecklare och hos allmänheten. Program utvecklade enligt FOSS-konceptet är i dag en stark konkurrent till program med sluten källkod som vi har sett uttryck för i bland annat statskontorets utredning kring öppen källkod (Statskontoret, 2003a). Program baserade på öppen källkod finns dock inte överallt eller i alla branscher. Vi skall nu titta vidare på vilka ekonomiska för- och nackdelar som kan finnas med utveckling enligt ”basarmodellen” som beskrivits av Raymond (2001a) och symboliserar utveckling enligt FOSS-principen.

7.2.3.1 Är FOSS ett ekonomiskt alternativ för dagens företag?

Vi har tidigare beskrivit hur basarmodellen fungerar och några praktiska för- och nackdelar med denna. Men vilka ekonomiska konsekvenser får då dessa för- och nackdelar för den utvecklande och den användande parten?

Till att börja med håller vi med Raymond (2001a) såtillvida att vi anser att en eventuell tidplan för utvecklingsprojekt enligt basarmodellen har goda möjligheter att kunna klaras av inom tidsramen eller till och med tidigare än beräknat beroende på hur många utvecklare som kan engageras, inte bara för att utvecklingsgruppen är mer motiverad som Raymond skriver och fler utvecklare kan skriva kod utan även för att fler personer då kan testa, underhålla och dokumentera koden. Om flera utvecklare utanför det ursprungliga projektet bidrar till komponentutvecklingen kan också kostnaderna för utveckling hållas nere vilket kanske blir mest accentuerat när den utvecklande och användande parten är samma företag eller person. Olika krav för att kunna attrahera externa utvecklare är bland annat att som vi har sett att produkten har ett allmänintresse för andra företag eller privatpersoner samt att intressanta utvecklingsområden finns kvar i projektet (Raymond, 2001a; Lerner & Tirole, 2000).

Ett möjligt problem är om användaren inte lyckas väcka intresse hos flera andra användare att hjälpa till att ta fram komponenter och testa dem, vilket vi ju sett Lerner och Tirole (2000) samt Raymond (2001a) observera i fallet Netscape när källkoden till deras webbläsare släpptes fri. Produkten riskerar då att ta lång tid att utveckla eller inte bli färdig om något problem skulle uppstå i utvecklingsfasen. Det kan också vara så att olika användare tar processen åt olika håll beroende på deras behov och intresseområden vilket kan göra det svårt att hålla ihop slutprodukten, alternativt kan produkten delas upp i flera delar enligt vad vi tidigare sett om ”forking” (Raymond, 2001a; Lerner & Tirole, 2000). Detta kan i någon mån motverkas genom att bryta ned slutprodukten i flera delkomponenter som kan utvecklas var för sig.

En modell för hur man som mjukvaruutvecklande företag med utveckling enligt FOSS-konceptet kan gå tillväga är också att arbeta enligt devisen ”gör-vad-kunden-vill-ha”, det vill säga utvecklingen initieras inte förrän en traditionell kravspecifikation över kundens behov är utarbetad. Fördelen är då att man får lite mera kontroll över tidsåtgången för projektet och kanske också över hur utvecklingen drivs, nackdelen naturligtvis att man får en högre utvecklingskostnad. Tänkbart är naturligtvis även här att man får positiva effekter om man lyckas engagera flera utvecklare som då naturligtvis kan minska utvecklingskostnader eller snabba på tidplanen. Fördelar uppnås då både för det utvecklande konsultföretaget (som ju får konsultintäkter) samt för den köpande parten (som kan få struktur över utvecklingen samtidigt som kostnadsbesparingar kan uppnås). En affärsmässig vinkling här är att produkten som utvecklas inte bör vara kärnverksamheten för den köpande parten eller ligga alltför nära knuten till dess huvudsakliga affärsidé för att undvika problem med vad Lerner och Tirole (2000) kallar ”free-riders”, det vill säga personer som helt enkelt tar ett projekt, modifierar det något och sedan stänger källkoden. Exempel på sådana produkter kan vara analysverktyg för olika sorters data eller mätvärden. Mest lämpligt är då å andra sidan denna metod för produkter med ett gemensamt intresse och låg koppling till företagets affärsidé, som till exempel tidsbokningssystem och liknande.

MySQL AB är ett företag som använt sig av denna affärsmetod med framgång enligt dess VD Mårten Mickos (Ny Teknik, 2004a).

En potentiell fördel för såväl utvecklare som användare av mjukvara vad gäller framtagning av program är också att det inte spelar någon större roll var på jordklotet den enskilde utvecklaren befinner sig så länge denne har en koppling till Internet och rätt plattform att utveckla och testa koden på. Kotler (1999) pekar som vi sett på trenden att tillverkning (också mjukvaruproduktion) flyttas till låglöneländer, besparingarna gäller alltså inte bara för utveckling av öppen källkod utan även för utveckling av sluten källkod. En möjlig risk med denna trend för utvecklare av stängd, proprietär mjukvara är att deras jobb hotas om utvecklingscenter och kontor flyttas eller ”outsourcas” till låglöneländer.

Vi tror dock inte att vi ännu sett slutet på utvecklingen av FOSS-produkter men att det finns begränsande faktorer för spridningen. En trend vi själva har observerat är att företag alltmer inkluderar FOSS-produkter i sina egna mjukvaruprodukter. En begränsning för att släppa ytterligare källkod fri är dock naturligtvis vilken ställning ett företag har på marknaden (jämför med Porters (1985) modell om marknadskrafter och Hedman och Kallings (2002) affärsmodell). Ett företag med en mjukvaruprodukt som är ledande på marknaden med en stor kundkrets och höga licensintäkter från sin produkt har naturligtvis inte så stort intresse av att öppna koden för allmänheten som ett företag med en mycket liten marknadsandel och en liten kundkrets som vill utmana den marknadsledande produkten.

Det kan ju vara intressant att ställa sig frågan hur ett marknadsledande företag skulle reagera på ett sådant scenario (att en konkurrent öppnar sin källkod för att attrahera nya utvecklare och ”slå undan benen” på det marknadsledande företaget), det troliga är väl enligt vår mening att man skulle avvakta utbredningen för konkurrenten och så småningom om utfallet för den nya konkurrenten blev väl övergå till att tjäna sina pengar på konsulttjänster och support i stället för licensintäkter eftersom man då inte kan konkurrera med priset på mjukvaran. Å andra sidan skulle det marknadsledande företaget då också kunna spara in på utvecklingskostnader för sin produkt antingen genom att öppna källkoden för att intressera fler utvecklare (mindre troligt eftersom a) allmänintresset förmodligen är lågt för en specialiserad produkt, b) få intressanta utvecklingsuppgifter lär finnas kvar samt c) företaget troligen inte vill låta konkurrenter titta på sin källkod) eller genom att utveckla mer enligt konceptet ”gör-vad-kunden-vill-ha” ovan fast med sluten källkod.

Oavsett sida (utvecklande/användande) eller bransch är det viktiga alltså ur ett ekonomiskt perspektiv för företag som vill generera intäkter baserat på produkter utvecklade med öppen källkod att se till att man använder eller stödjer en utvecklingsmodell som säkerställer att man bibehåller konkurrensfördelar och kritiska framgångsfaktorer som exempelvis snabb leveranstid och hög kvalitet vilket borgar för nöjda kunder.

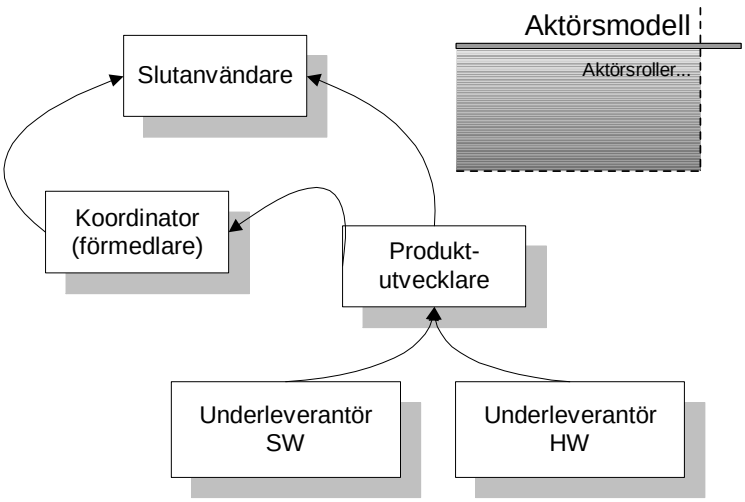
Hedman och Kalling (2002) säger att nya affärsmodeller inte behövs för dagens företag, inte ens inom elektronisk handel, utan företag anpassar i stället sina befintliga affärsmodeller till den nya ekonomin exempelvis genom öppnandet av nya försäljningskanaler via Internet. Definitionsmässigt (per definition av en

affärsmodell) kan detta måhända vara riktigt. Vi anser dock att några intressanta observationer kan göras i detta avseende, åtminstone vad gäller användning och utveckling av FOSS-produkter.

Den ”nya” ekonomi som Hamel och Prahalad (1996), Kotler (1999) med flera talar om är ny i avseendet snabbt föränderlig och med globala nätverk av aktörer på en marknad som spänner över hela jordklotet (vilket också Dahlander, 2004 visar). FOSS-rörelsen har påskyndat denna utveckling och gett upphov till de förändringar i befintliga företags affärsmodeller som Hedman och Kalling diskuterar och exemplifierar med Barnes & Noble (även IBM, Netscape med flera, våra exempel) men det finns också en mängd nya företag som uppkommit och fortfarande uppkommer i FOSS-rörelsens spår. Dahlander (2004) gör en uppdelning i företag som aktivt arbetar med FOSS-utveckling (så kallade ”FRIENDS” – engelska för vänner) och företag som försöker generera intäkter från FOSS-produkter andra utvecklat (så kallade ”FOES” – engelska för fiender) men konstaterar att oavsett vilken sida så förändras sättet att göra affärer för dessa företag ständigt. Eftersom det inte finns något enhetligt sätt att göra affärer med produkter baserade på öppen källkod försöker företag idag enligt Dahlander snabbt förändra sitt sätt att göra affärer efterhand som de märker att tidigare strategier och affärsmetoder inte fungerar. I takt med att olika strategier prövas och utformas eller förkastas får så nya och befintliga företag en snabbt växande erfarenhets- och kunskapsbas för vilka typer av affärsmetoder och modeller som fungerar respektive inte fungerar. Den osäkerhet som finns över hur man bäst utnyttjar öppen källkod för att generera intäkter menar Dahlander får företag att vilja kunna förändras snabbt och byta strategier, vilket stämmer väl överens med de resonemang vi tidigare framfört från bland annat O´Reilly (2001), Kotler (1999), Morgan (1997), Rappa (2004a) samt Hedman och Kalling (2002). Som Hedman och Kalling (2002) också uttrycker det ger den nya ekonomin inte upphov till nya affärsmodeller men väl till nya affärsmöjligheter i form av tjänster som efterfrågas då nya kontaktytor uppstår mellan aktörer på marknaden i spåren av att aktörer från den ”gamla” ekonomin med gamla affärsmodeller gör sitt intåg på Internet och den nya globala marknaden. Detta kan leda till problem för företag att positionera sig och hitta ”rätt” på marknaden, vilket också Dahlander (2004) noterar när han påvisar att företag som arbetar med produkter baserade på öppen källkod ställs inför stora utmaningar då viktiga resurser för att hitta konkurrenskraftiga produkter och tjänster ligger utanför företagets egen gräns och kontroll och kanske till och med i allmänhetens egendom och inte ens hos andra företag. Vi skall nu bekanta oss med ett exempel på en affärsmetod som vi med stöd i detta resonemang ser har framtiden för sig när det gäller att skapa mervärde för kunder utifrån processer och produkter utanför det egna företagets kontroll.

Ett företag som lyckats hitta en ny affärsmetod eller om man så vill affärsmodell i den ”nya” ekonomin som ser till största möjliga värde för kunden är Collab.net (http://www.collab.net). Den koordinatorroll företaget använder sig av innebär att värdet för kunden blir det centrala och koordinatorns uppgift blir att tillgodose kundens behov genom att plocka samman komponenter från olika leverantörer till en helhet som blir det bästa för kunden. Denna möjlighet blir oändligt mycket större med öppen mjukvara än med stängd eftersom små delkomponenter kan hämtas varifrån som helst (där avstånd och personliga affärskontakter i detta fall spelar en mindre roll) och koordinatorn/kunden inte blir låst till en stor mjukvaruutvecklare

med dess möjliga brister för varje specifik kunds behov. Vi har illustrerat en tänkbar aktörsmodell i figur 7.1 för att förtydliga resonemanget.



Figur 7.1: *Aktörsmodell med koordinatorroll (egen)*

Produktutvecklare och slutanvändare kan, men behöver inte, vara samma person eller företag. Detta spelar mindre roll för koordinatorrollen, kärnpunkten i resonemanget är i stället att eftersom floran av ”Open Source”-program ständigt ökar kan det ibland vara svårt för en slutanvändare att veta vad som redan finns utvecklat och vad som kan passa till deras behov. Här finns alltså som exempelvis Collab.net upptäckt en god möjlighet utöver de traditionella inom Open Source att tjäna pengar; Att helt enkelt vara så pass bra på marknaden av FOSS-produkter att konsulttjänster och stöd kan erbjudas till företag för att hjälpa dem att ta fram bra produkter till deras behov till så låg kostnad som möjligt. En styrka är specialkompetensen att helt eller delvis återanvända befintliga ”Open Source”-lösningar i nya sammanhang och miljöer, vilket då blir en nyckelfaktor (CSF) för att uppnå marknadsfördelar mot sina konkurrenter. En rimlig förutsättning för att lyckas med denna affärsmetod inom IT-branschen skulle enligt vår modell vara att ha goda kontakter inom såväl hårdvaru- som mjukvaruindustrin men även inom ”Open Source”-rörelsen. Bassellier och Benbasat (2004) understryker också vikten av kunskap om kundens verksamhet och affärsprocesser. Det går också att göra en koppling till tankegångarna kring ”Balanced Scorecard”-modellen. Finansiellt bör då Collab.net visa att deras affärsmodell fungerar för att generera vinst till företaget eller åtminstone hög tillväxt av kunder och marknadsandelar. För kundperspektivet gäller det att visa att det finns en positiv effekt av att anlita Collab.net, vilket exempelvis kan göras genom marknadsföring av lyckade kundprojekt eller genom en attraktiv prissättning på tjänsterna som levereras. Det interna perspektivet kan för Collab.net innebära att företaget har en låg personalomsättning eller lätt att hitta kompetent personal. Förnyelseperspektivet slutligen kan innebära en god förmåga att ta till sig och använda ny teknik och nya FOSS-produkter och integrera dem i kunders system och affärsprocesser.

Jämför vi dessa resonemang med Hedman och Kallings (2002) modell över affärsmodellen ser vi att koordinatorn har en viktig uppgift i att samordna och söka upp de bästa tillgängliga resurser som finns på marknaden för att kunna ge marknaden och sina slutkunder ett så bra erbjudande som möjligt. Man angriper här alltså i första hand företags problem att kontrollera de yttre processer och nätverk som existerar på marknaden för mjukvara och hjälper företag att maximera avkastningen på de resurser företaget förfogar över eller har åtkomst till på marknaden för leverantörer i detta specifika fall. Personer i koordinatorroller är i sin tur enligt synsättet RBV (Resource Based View) och enligt Bassellier och Benbasat (2004) det koordinerande företagets unika resurser som bringar konkurrensfördelar.

Sett till vår aktörsmodell enligt ovan och Porters (1980) modell över marknadskrafter kan även koordinatorn fungera som köpare om denne köper in delkomponenter (som ju också kan vara tjänster eller produkter med stängd källkod om det inte existerar några goda alternativ baserade på öppen källkod) för att leverera en större enhet till slutanvändaren, vilket påverkar affärsmodellen för koordinatorföretaget och i första hand då affärsmetoder för styrning av kommunikation och handel med underleverantörer. Koordinatorn kan också ses som en leverantör till slutanvändaren beroende på vilken roll och vilket ansvar koordinatorn har mot slutanvändaren.

Typiska prov på andra branscher där affärsrollen för koordinatorn finns redan idag är exempelvis finans- och försäkringsmarknaden, där speciella företag finns som agerar som förmedlare och rådgivare vid till exempel fondsparande eller som rena mäklare när det gäller både olika former av sparande och försäkringar.

En fara med exempelvis finans- och försäkringsbranschen är att ”mäklaren” eller ”koordinatorn” blir färgad av eventuell bonus eller förmåner om denne lyckas sälja eller förmedla produkter från ett visst företag. Denna risk finns naturligtvis också även inom mjukvarubranschen, men med en ökande mängd FOSS-alternativ finns förhoppningen att organisationer med en ökad kunskap om marknaden skall kunna erbjuda kunden bästa alternativet till lägsta möjliga kostnad. Detta skulle kunna vara intressant att undersöka vidare.

7.2.4 Sammanfattning Perspektiv 0

För att lyckas som företag i dagens komplexa affärsvärld gäller det alltså (och detta gäller så vitt vi kan se för mer än mjukvarubranschen) att vara duktig på att positionera sig rätt på marknaden, exempelvis enligt Porters modell för marknadskrafter eller enligt ”Balanced scorecard”-modellen, samtidigt som det gäller att se över sina inre såväl som yttre processer och sina resurser (baserat på exempelvis företagsmodellen RBV (Resource Based View) samt Hedman och Kallings (2002) tankar kring affärsmodell) genom att exempelvis konstant identifiera och övervaka sina kritiska framgångsfaktorer (Critical Success Factors, CSF:s). Att som organisation konstant lära sig nya saker om marknaden och dess aktörer är också viktigt för att inte tappa marknadsandelar i dagens komplexa och globala affärsvärld; jämför Morgans hjärnmetafor i Morgan (1977) och Kotlers tankar om den nya ekonomin i Kotler (1999). O´Reilly (2004a) menar som vi tidigare refererat

till specifikt för mjukvarubranschen att det inte räcker med ett starkt varunamn för att bli ett lönsamt mjukvaruföretag idag. Detta är en realitet både för företag som arbetar med öppen och stängd källkod, ingen vill ju köpa dåliga produkter. Ett varumärke som IBM, Sun eller Microsoft är ju enligt vår mening ändå en bra bit på väg mot goda försäljningssiffror för en produkt eller tjänst.

7.3 Perspektiv 1

I följande avsnitt kommer vi att resonera kring hur det kommer sig att man kan ta betalt för och tjäna pengar på något som är ”gratis”.

7.3.1 Analys av påstådda fördelar med FOSS

Låt oss för en stund betrakta några påstådda fördelar med FOSS och resonera kring dem.

Bruce Perens skriver i Open Sources: Voices from the Open Source Revolution: *”Companies that use open-source software have the advantage of its very rapid development, often by several collaborating companies, and much of it contributed by individuals who simply need an improvement to serve their own needs.”* (Perens, 1999). Låt oss försöka analysera detta påstående och undersöka rimligheterna i det lite närmare.

Det är genom friheterna med öppen källkod och då främst genom de villkor som stipuleras i de ”copyleftade” licensvarianter som främst förespråkas av Free Software Foundation, det vill säga GPL och LGPL, som argumentet ovan kanske bäst kan förstås.

Eftersom var och en garanteras samma rättigheter till en programvara kommer alla som bidrar till utvecklingen att befinna sig på samma nivå gentemot varandra vad gäller tillgång till programmet både i källkodsform och i exekverbart format. Det är också genom rättigheterna som vi kanske bäst förstår varför individer bidrar till utvecklingen av FOSS-program. Personer som bidrar till utvecklingen av fri programvara är genom licensvillkoren garanterade att de tillägg som de själva gör nu blir öppet tillgängliga för alla utan att någon ges rätten att begränsa friheterna för programmet i framtiden. Om någon annan part vid ett senare tillfälle förbättrar programmet med egna tillägg så kommer även dessa nya förbättringar att vara öppna och tillgängliga för alla.

Även för programvara som är Open Source, men inte copyleftad, finns starka skäl för aktörer att dela med sig av modifikationer och tillägg åtminstone om programmet, eller de delar av programmet som ändringarna rör, inte ligger i aktörens kärnverksamhet. Genom att dela med sig och bidra med exempelvis en buggrättning finns chansen att buggrättningen kommer med i senare versioner av den officiella varianten av programmet. Detta skulle minska integrationsarbetet för den bidragande

parten samtidigt som andra, även potentiella konkurrenter, kan dra nytta av förbättringen.

7.3.1.1 Applicering av spelteori

Ett sätt att försöka förstå fenomenen fri programvara och öppen källkod är att nyttja tankar och exempel från den ekonomiska teori som kallas för spelteori. Vi kommer inte att försöka oss på en alltför djuplodande analys, men genom att applicera tankarna kring spelteori på FOSS får vi ytterligare en horisont utifrån vilket vi kan försöka vinna insikt i om och hur det i så fall kommer sig att FOSS kan fungera på en marknad och under vilka villkor.

Vi bekantade oss tidigare med spelet ”chicken” och analyserade det med hjälp av spelteori. Låt oss nu spela ett ”chicken”-inspirerat spel med mjukvarulicenser där resonemangen är hämtade från Rysdam (2000).’

Ponera att vi har följande typer av licenser:

- **T**raditionell: Källkoden är ej tillgänglig.
- **O**pen: Källkoden tillgänglig och kan inkluderas fritt i andra program.
- **G**PL: Källkod tillgänglig, men den kan enbart inkluderas i program som även de har en GPL-kompatibel licens.

Låt oss nu se valet av licens (T, O eller G) som ett val av strategi. Rysdam (2000) ställer sedan upp en spelplan där med en mängd program som alla ska utföra samma uppgift och där vart och ett av de olika programmen kan välja någon av licenstyperna T, O eller G. Rysdam låter sedan de olika licenstyperna tävla mot varandra iterativt i flera omgångar. Författaren ger följande exempel:

- ***“T1 meets T2: T1 proves superior. No future change as source code is not available for T2 to examine and thus improve self.***
- ***O1 meets O2: O1 proves superior. O2 can examine the code from O1 and incorporate it. On the next encounter (remember that an ESS includes iteration) O1 and O2 have more nearly equal performance.***
- ***G1 meets O1: G1 proves superior. O1 can examine the code from G1 but cannot incorporate it without changing strategies.”*** (Rysdam, 2000)

Vi kan sammanfatta resonemanget i tabell 7.1 nedan.

Tabell 7.1*: Licensspel 1, antagande att det är negativt att koden* delas ut

	T	**O**	**G**
T	0	10	0
O	0	5	0
G	0	10	5

När ett program med en traditionell (T) licens möter ett annat T program byts ingen kod och poängen blir 0. När T möter O kan T använda O- programmets kod (vilket i exemplet ger 10 poäng). Om T däremot möter G kan T-programmet inte använda någon kod eftersom G-licensen enbart gör koden tillgänglig för program med samma villkor.

I exemplet ser vi att ”Open Source”-licenserna förlorar. Rysdam (2000) ställer sen upp ett spel mellan GPL och Traditionella licenser som visar att GPL vinner.

Det finns dock svagheter i Rysdams modell där ett problem är att Rysdam förutsätter att det är negativt för ett program om ett annat program kan använda dess källkod. Detta har uppmärksammats av Goldman (2000) som visar på dessa brister i Rysdams resonemang. Goldman ställer upp ett spel där det inte är negativt att andra använder ens kod. Resultatet blir då enligt tabell 7.2 nedan.

Tabell 7.2: *Licensspel 2, antagande att det inte* är *negativt att* koden *dela*s ut

	T	**O**	**G**
T	0	10	0
O	0	10	0
G	0	10	10

I detta fall vinner fortfarande GPL, men utan att licenser av typen Open förlorar mot Traditionella licenser. Antag nu att vi vill att vår kod ska spridas. Ett spel enligt detta scenario skulle ge en resultatskärm som i tabell 7.3.

Tabell 7.3: *Licensspel 3, antagande att det är positivt att koden* delas ut

	T	**O**	**G**
T	0	10	0
O	10	20	10
G	0	10	20

Goldman (2000) drar slutsatsen att det finns rum för debatt kring huruvida licenser av typen O eller licenser av typen G är överlägsna och att utfallet är beroende på det resultat som författaren vill ha. Däremot vill Goldman mena att det är belagt att licenser av typen T är dömda till att vara ständiga förlorare i detta spel.

Om vi ser som exempel en situation var två eller fler parter samtliga bidrar till utvecklingen av ett program där var och en av parternas bidrag ger övriga parter potentiella fördelar. Om nu en av parterna väljer att behålla sina förbättringar för sig själv utan att dela med sig och fortfarande inhämta förbättringar från de andra parterna skulle denna part kunna gagna fördelar gentemot de andra. I en sådan situation vore det inte långsökt att anta att de andra parterna skulle svara med samma mynt och även de sluta dela med sig. Summan av ett sådant scenario skulle vara att den totala förtjänsten för samtliga parter, främst i meningen sparat arbete per deltagare eller programvara som har mer eller bättre funktionalitet, skulle minska.

Susning (2004a) konstaterar att *”Begrepp som kartellbildning och orsakerna till att bryta sig ur en sådan kan också beskrivas med hjälp av spelteori.”*. Låt oss därför spinna vidare kring detta då Free Software (i meningen programvara som är copyleftad, exempelvis program med licensen GNU GPL eller liknande) kan ses som en variant av en kartell där de som väljer att samarbeta tvingas dela med sig. Det går inte bryta ur sig ur ”kartellen” och dra fördelar enbart för sig själv. I en vanlig kartell är medlemmarna en sluten klubb som samarbetar. Låt oss jämföra klubbarna med licenstyperna T, O och G ovan. I kontexten av öppen källkod av typ G är klubben garanterat öppen för alla och alla medverkandes arbeten delas. Medlemskap i klubben är dock inte på något vis tvingande. Klubbar av typ T är exklusivt tillgängliga för den som betalar, medan klubbar av typ O är öppen för vem som helst att gå med i utan plikter.

Ett potentiellt problem är det med ”freeriders”, det vill säga parter som drar nytta av andras arbete utan att själva bidra. Spelfältet för freeriders begränsas i olika utsträckning av olika licensvillkor om parten själv ska vidareutveckla programmet. På ena extremen finner vi copyleftade licenser såsom GNU GPL och på andra sidan återfinns olika former av public domain.

7.3.2 FOSS som skenbar värdeförstörare

Relationen mellan FOSS och marknaden kan betraktas på olika sätt, men klart är att det krävs att aktörerna behöver anpassa befintliga eller anamma nya affärsmodeller där vår koordinatorroll är ett exempel.

I artikeln "The Open Source Paradigm Shift" skriver Tim O´Reilly (2004a) att *"A huge amount of software value appears to have vaporized."* (angående Red Hat som inte tjänar jättemycket pengar på licensförsäljning i jämförelse med Microsoft). O´Reilly (2004a) går sedan vidare och frågar retoriskt om det verkligen är så att äkta värde har försvunnit eller om det rör sig om *"overhead"*. Vad är det då för intäktsmodell som fungerar för Open Source om det inte går att bli stormrik på support och underhåll? O´Reilly (2004a) lutar åt att företag ska bli mer inriktade på att erbjuda tjänster baserat på Open Source snarare än support. Ett utmärkt exempel på detta är Google. Google levererar sina tjänster på en plattform som till stor del bygger på Open Source och linuxservrar (O´Reilly, 2004a).

Även Raymond (2001a) var inne på tanken att tjänster baserade på mjukvara skulle bli framtidens melodi för Open Source. Den första vågen av "Open Source"-entreprenörer gjorde dock annorlunda och de *"fokuserade på tjänster associerade med underhåll och support av programvaran, snarare än äkta mjukvara som en tjänst"* (i sig) (O´Reilly, 2004a) .

Dessa tankegångar är nära besläktade med våra egna då vi i början försökte ställa kompassen mot målet att finna nya spännande sätt att tjäna pengar på fri programvara och öppen källkoden. Själva naturen i FOSS gör det omöjligt för någon att införskaffa en monopolställning som exempelvis den Microsoft har idag.

Coleman (2004a) noterar att IBM och andra företag som använder FOSS i sina produkterbjudanden väljer att betona att FOSS möjliggör snabbhet och smidig marknadsanpassning och att position stärks visavi företagets konkurrenter.

7.4 Perspektiv 0 + 1

Vi har redan sett hur perspektiven överlappar varandra, men vi vill även knyta ihop och förtydliga sambanden.Vi vill försöka illustrera detta med en matris där vi försöker väva ihop begreppen för att få en klarare bild över var företeelsen Open Source befinner sig i dagens globala affärsvärld relaterat till de licensmodeller som finns att tillgå.

Vi utgår från fyra generella modeller för licensiering av fri och öppen källkod från Olofsson (2003a) samt proprietära licenser:

- Tillåtande licenser, såsom BSD och Apache license, som ger licenstagaren stora friheter och inte ställer som krav att källkod till licenstagarens modifieringar är öppen.

- Restriktiva licenser, såsom GPL och andra copyleftade licenser, som försäkrar sig om att licenstagaren som bearbetar programmet ger ändringarna tillbaka till upphovsmannen om programmet distribueras vidare.

- Dubbellicensering, såsom MySQL och TrollTech, där licensgivaren ger ut programmet under flera licenser och i fallet MySQL exempelvis både en traditionell proprietär licens och GPL;

- Kompromissande licenser, såsom LGPL och MPL, där man försöker förena det bästa ur två världar.

- Proprietära licenser, såsom Microsoft Office EULA, där användaren får rätt till använda en begränsad mängd instanser av programmet i binär form men inte har rätt till källkoden.

Tabell 7.4 nedan baseras på tidigare redovisad litteratur kring licensmodeller och ekonomi exempelvis Porter (1980). De centrala externa aktörerna vi har valt att plocka ut är leverantör, kund och konkurrent. Tabellen relaterar vi val av licensform till aktör.

Tabell 7.4: *Val av licenser för ett mjukvuruutvecklande företag, i relation till aktörer.*

Licens- och aktörs-perspektiv	**Tillåtande**	**Restriktiva**	**Dubbellicensering**	**Kompromissande**	**Proprietära**
Leverantör	OK	OK? (om L har samma licens)	OK? (om L har samma licens)	OK	OK
Kund	OK(kund vill kanske dock även ha tillgång till källkod)	OK? (om kunden skickar program-met vidare i sin tur måste hon bifoga källkod; problem-atiskt om kunden samman-fogar med egna proprietära program)	OK (kan välja öppen eller stängd licens)	OK	OK (kund vill kanske dock även ha tillgång till källkod)
Konkurrent	OK (vill kanske inte visa källkod för konkur-rent)	Ej OK, (tvingas göra källkoden öppen; dock OK om källkoden ej är kärnverk-samhet)	Ej OK, (tvingas göra källkoden öppen; dock OK om källkoden ej är kärnverk-samhet)	Ej OK, (tvingas göra källkoden öppen; dock OK om källkoden ej är kärnverk-samhet)	OK

I tabellen ovan försöker vi visa hur ett mjukvaruutvecklande företags val av licens påverkar affärsmöjligheterna. Vi har i tabellen därför ställt tre externa aktörers förhållningssätt mot företagets potentiella licensval.

En annan aspekt utifrån vilken vi kan relatera licensval med affärsmetoder är Bostonmatrisen. För ett mjukvaruutvecklande företag med en ”stjärna”, eller ”kassako” finns det inga akuta anledningar att öppna källkoden, utom möjligen fallet då företaget har en begränsad produktportfölj som börjar ansättas av attacker från

konkurrerande företag med nya produkter då öppning av källkoden kan ses som ett sätt att överleva på lång sikt. En sådan förändring kräver möjligen att affärsmodeller omformas så att intäkter inte främst kommer från licenser. För ”frågetecken” är produktens eller tjänstens marknadsposition ett frågetecken. Val av licens bör göras utifrån den möjliga framtida marknadsandelen. Produkter som ”hundar” bör avlivas eller öppnas om inget annat för att därigenom idka sabotage mot konkurrenter. För befintliga kunder till produkter som klassas som ”hundar” skulle öppning kunna vara av godo då underhåll och vidareutveckling av produkten blir möjlig i en öppen gemenskap.

7.5 Reflektioner kring Open- och Closed Source

I denna avdelning kommer vi att presentera ett antal argument för- och emot fri programvara och öppen källkod. Syftet är att komplettera bilden från perspektiv 0 och perspektiv 1 ovan med andra relaterade faktorer.

För att underlätta analysen kommer vi att nyttja ”Closed Source” (eller om man så vill proprietär programvara) som kontrast. Vi börjar med en fiktiv dialog mellan en förespråkare för Open Source och en skeptiker.

7.5.1 En inledande dialog

Vår dialog har två kombattanter. Pingvinen Tux, som är den officiella maskoten för Linux (Baker, 2004a), gillar det där med fri programvara och öppen källkod. Motståndaren kan vi kalla för Mr X och han får ikläda sig rollen som sund skeptiker. Mr X inleder samtalet med att påstå följande:

- **Mr X:** Det går ju inte tjäna pengar på Open Source eftersom det är gratis!
- **Tux:** Du kan visst ta betalt för Open Source och kringliggande tjänster såsom exempelvis support och utbildning.
- **Mr X:** Då kan man likaväl köpa ett vanligt traditionellt program!
- **Tux:** När du köper stängd programvara får du oftast inte tillgång till källkoden och ursprungligen följde faktiskt programmen och deras källkod med datorsystemen.
- **Mr X:** Jag kan ju ändå inte programmera så vad spelar det mig för någon roll?

- **Tux:** Genom att göra källkoden öppet tillgänglig kan fler utvecklare rätta fel och lägga till nya funktioner som du sedan har nytta av.

- **Mr X:** Ok, men varför arbetar de gratis för mig!?

- **Tux:** Det kan handla om att de får en ”kick” av att rätta fel och att det sedan syns att de har varit med och byggt ett program som andra använder. Det kan också vara så att de genom att själva lokalisera och åtgärda ett fel direkt kan fortsätta att använda programmet utan att behöva vänta på nästa ”release” något de vore tvingade att göra om det rört sig om ett proprietärt stängt program.

I den här debatten fick Tux sista ordet men Mr X hade säkert haft fler invändningar om vi hade låtit honom fortsätta. Dialogen är givetvis påhittad, för när hörde du en pingvin senast dryfta andra saker än sill, men likväl borde ovanstående resonemang kunna ha ägt rum - mellan andra parter såklart.

7.5.2 Ordval och värderingar

Vi kommer nu att undersöka fenomenet med fri programvara och öppen källkod ur ett slags idéperspektiv där vi försöker beakta val av termer och ordens laddningar.

7.5.2.1 Open Source

Open Source, eller öppen källkod för att använda en svensk översättning, har många positivt laddade konnotationer. Öppen som i tillgänglig för alla, fri, en slags jämlikhet och ja närmast broderskap något som lätt för tankarna till den franska revolutionen. Makt åt folket! Och visst ligger det något revolutionärt i en del av tankarna. Vissa av förespråkarna och då kanske främst Richard M Stallman (RMS) framstår för en del säkert som något av en övervintrad hippie.

7.5.2.2 Closed Source

Nåväl om vi vrider blicken och väljer att betrakta motsatsen till Open Source, det vill säga Closed Source, eller stängd källkod, möts vi av andra negativt laddade huvudvärden. Stängd betyder inte öppen. Stäng som i inlåst. Stängd som i för mig, men minsann inte för dig eller helt enkelt stäng som i inte (vid)öppen. Den senare associationen har förespråkare för Closed Source försökt vända till något positivt genom att hävda att stängd källkod är säkrare mot angrepp från ”crackers”. Givetvis hävdar förespråkare för ”Open Source”-lägret motsatsen, med argumentet att det är fler personer som kritiskt granskar koden och därigenom kan upptäcka potentiella säkerhetsbrister i tid. Ett annat argument för att Open Source skulle vara bättre ur ett säkerhetsperspektiv är att säkerhetsfixar snabbt blir tillgängliga, allt i enlighet med devisen *”Release early. Release often. And listen to your customers.”* (Raymond, 2001a sid.

Comment [f4]: Kex !

29). Noterbart är att denna devis omfamnas av olika så kallade agilemetoder såsom XP (eXtreme Programming) (Xprogramming, 2004a; Agilealliance, 2004a).

7.5.2.3 Free Software

Låt oss även studera termen Free Software, eller fri programvara, för en stund. Även benämningen Free Software har en grundläggande positiv innebörd för de flesta även om just ordet Free blir lite problematiskt, som tidigare diskuterats, då det i det engelska språket är oklart om ordet Free syftar till fri som i gratis (det vill säga utan pris) eller fri som i (yttrande)frihet. Det är den senare betydelsen som förespråkare för Free Software har i tankarna, men ofta är det den förra meningen som folk i allmänhet får som bild framför ögonen. Free Software skall inte blandas samman med freeware och inte heller med public domain.

Något som är öppet, gratis och fritt kan väl inte vara något för den kapitalistiska världen?

7.5.2.4 Infekterad av Open Source

Än om vi i denna uppsats har försökt undvika att dyka alltför djupt ner i de mer anskrämliga inslagen i debatten mellan öppen och stängd källkod är det svårt att avstå från att resonera lite kring begreppet FUD (Fear Uncertainty Doubt) och vad som på engelska kallas ”the viral nature of GPL” och tal om att GPL ”infekterar” annan programvara (Greve, 1999a).

”Om din kod blir smittat av GPL kommer du tvingas ge ut din källkod under GPL”. Detta påstående kan verka avskräckande för den som har en verksamhet där källkoden ses som en konkurrensfördel och där man absolut inte vill riskera att behöva ge ut källkoden. Det kan dock hävdas att GPL är en licens och inte ett kontrakt och att GPL därigenom aldrig kan få som effekt att någon tvingas dela med sig av sin kod. Snarare kan det handla om att man tvingas ta bort GPL-koden från sitt program och/eller att man får betala skadestånd till copyright-innehavaren. (Groklaw, 2004a).

7.5.2.5 Ideologiska och ekonomiska aspekter kring FOSS

Eric S Raymond (ESR) och Richard M Stallman (RMS) är båda förespråkare för öppen källkod, men de har olika medel och agendor. Utgångspunkten för RMS är snarast ideologisk och han kan uppfattas som något av en fundamentalist. ESR är mer pragmatisk och försöker flirta med den kommersiella världen. ESR försöker vädja till vårt ratio (förnuft) genom att argumentera att Open Source är den bästa modellen att utveckla i rent krasst ur en logisk eller ekonomisk vinkel snarare än en moralisk.

Andra som exempelvis Coleman (2004a) menar att FOSS-rörelsen vill inta en agnostisk ställning till politik och att det som håller på att hända är att IP och

copyright används för att skydda yttrandefriheten snarare än det traditionella syftet att säkra äganderättigheterna.

Det eleganta trick som Stallman uppfann var copyleft-mekanismen. Coleman kommenterar detta: *"Using copyright as its vehicle, the copyleft places copyright literally on its head and in the process demystifies copyright's "absolute" theory of economic incentive. The copyleft says, we are not the passive "subjects" of an almighty, unchangeable law, but actually can create the law to serve us for other ends: in the case of FOSS, that of free speech."* Coleman (2004a). Stallmans ståndpunkt kan ses som en ideologisk eller moralisk om man så vill. Programvara ska vara fri, inte proprietär och stängd.

Låt oss därför resonera lite kring privat äganderätt och se om det hör ihop med FOSS och marknaden. Det är dags att ta ett kliv bakåt i historien och bekanta oss med "An Inquiry into the Nature and Causes of the Wealth of Nations" av Adam Smith (1776). Nåväl, Smith argumenterade att priset för en vara var beroende enbart av produktionskostnaden vilket inte kan vara helt rätt. Extra tydligt blir det när vi betraktar mjukvara som både billigt och enkelt kan reproduceras. Däremot argumenterade Smith (1776) även för att individuell frihet ger grunden till framgång och rikedom för samhället i stort; eller *"Free as in freedom - not beer?",* för att jämföra med FSF (2004a). Det som framförallt Free Software handlar om är att garantera rättigheter för alla.

Om vi resonerar kring rättigheter kan det vara intressant att sätta fokus på äganderätten och då är "Two Treatises of Government" av John Locke relevant (Locke, 1698). Locke (1698) argumenterade för bland annat äganderätten för var och en. Locke menade att individen har rätten till sin kropp, sitt arbete och egendom. FOSS handlar om att nyttja sin rätt till att dela med sig till andra ibland, som i fallet med GNU GPL, under förutsättning att mottagaren ger andra samma rättigheter som hon fick. Numera används bland annat patent för skydda egendom (i form av immateriella rättigheter). Samtidigt som patent skyddar verkar de hindrande och inskränker andra parters frihet.

Marc Andreesen, en av företaget Netscapes grundare, har gjort följande uttalande om Internet: *"What the Net is, more than anything else at this point, is a platform for entrepreneurial activities -- a free market economy in its truest sense. It's a level playing field where people can do anything they want to."* (Andreessen, 1995). Låt oss travestera Netscapes grundare genom att byta ut Internet mot Open Source; *"Vad Open Source ger, mer än någonting annat just nu, är en plattform för entreprenörisk verksamhet – en fri marknadsekonomi i dess sannaste mening".* Open Source kan också kopplas till Internet, nätverkande samarbete och så vidare. Vi avslutar dock avsnittet med en annan tänkvärd vinkling på temat Netscape. Microsoft kom att konkurrera ut Netscape genom att ge bort sin webbläsare Internet Explorer gratis*; "Free as in free beer - not freedom".*

7.5.3 Ur ett utvecklarperspektiv

Vi kommer nu att diskutera kring olika aspekter på utveckling av fri programvara och öppen källkod, vilka för- respektive nackdelar finns.

En av de största fördelarna och lärdomarna som kan ses exempelvis på Linux eller Apache för att ge ett par exempel är att koden kan bli av hög kvalitet. Ytterligare fördelar är låg utvecklingskostnad (eftersom många hjälps åt utan att få betalt för sitt arbete) samt stabilitet och framtidssäkerhet eftersom programvaran inte riskerar att dö ut för att något företag går i konkurs, något som annars kan hända med kommersiell programvara.

Det finns naturligtvis även nackdelar. En stor fara med ”Open Source”-projekt är om projektägaren/-ägarna tröttnar på projektet, varigenom utvecklingen kan avstanna. Ytterligare ett problem, som även i ”Open Source”-historiens relativt sett korta tidsperiod har visat sig, är att med så många utvecklare är det oundvikligt med konflikter mellan olika grupper av utvecklare som vill driva utvecklingen åt olika håll det vill säga forkar (Rosenberg, 2000). Detta ses vanligtvis som ett problem, både för utvecklarna själv som då har två kärnor att hålla uppdaterade med eventuella felrättningar/utvecklingar i den gemensamma koden och för användaren som får problem att välja rätt version av programmet, men möjligheten att knoppa av är också ett utslag för den anarkistriska ådran som ibland lyser igenom inom ”FOSS”-rörelsen. Möjligheten att knoppa av är också ett uttryck för de friheter som är kopplade till FOSS.

En konfliktsituation kan också uppstå när någon (person eller organisation) finansierar utvecklingen antingen helt eller delvis och därmed kan kräva visst inflytande över mjukvaran. Exempelvis finansierade FSF delar av Debianutvecklingen och stod dessutom för en stor del av källkoden till Linux via sitt nätverk av utvecklare av fri mjukvara. FSF med dess företrädare Stallman ville därför att Linux med dess anpassningar skulle döpas om till Lignux för att understryka vikten av GNU som en förutsättning för Linux (Moody, 2001). En annan kanske vanligare benämning är GNU/Linux (Stallman, 2004b; Stallman, 2004c ; Debian, 2004b). Namnet är ju en nog så viktig del av produkten, men naturligtvis kan även innehållet i produkten/mjukvaran påverkas av kommersiella intressen.

7.5.4 Transitioner

I denna avdelning kommer vi att undersöka transitioner, eller om man så vill byte, mellan olika licensvillkor. Transitioner gav vi en provsmakning på i kapitel 5 ”Exempel på programvarukategorier”. I avsnittet kommer vi att relatera transitioner med olika typer av licenser vilka vi beskrivit i kapitel 4 ”Kategorier av programvara”.

Den som äger copyrighten för en programvara kan på eget bevåg ändra licensvillkoren för nya instanser av sitt program. Befintliga program som redan har levererats till en slutanvändare påverkas emellertid inte av en sådan manöver såtillvida att detta inte uttryckligen har avtalats mellan parterna.

För ett proprietärt program är detta inga som helst problem rent juridiskt och ett exempel på ett företag som har ändrat licensvillkor är Microsoft.

När det gäller fri programvara och öppen källkod är det hela lite mer problematiskt. Program som täcks av GNU GPL garanterar, genom copyleften, att alla instanser av program härledda, det vill säga rena kopior eller modifikationer, av ett annat program också de kommer att täckas av GNU GPL. Notera dock att upphovsmannen kan välja att låta nya instanser av ett program gå under andra licensvillkor som tar bort de friheter som användaren får via GNU GPL. Vem som helst kan dock fortsätta att utveckla programmet vidare från de gamla instanserna.

Vanligast, i fallet att upphovsmannen har ett behov att introducera nya licensvillkor, är kanske dock att copyrighthållaren ger ut instanser av sitt program under flera, i sig möjligen inkompatibla, licensvillkor parallellt. Ett exempel på det sistnämnda är databashanteraren MySQL som ges ut under både GNU GPL och en kommersiell licens (MySQL, 2004). Om användaren inte önskar att nyttja programmet under GNU GPL kan hon köpa en kommersiell licens av upphovsrättsägaren. Ett annat exempel på dubbellicensiering är skriptspråket Perl där användaren får välja om hon vill nyttja Perl under Artistic License eller under GNU GPL (Olofsson, 2003a).

Om man inte har copyrighten till ett program kan man ändå för vissa licensvillkor byta licensen för instanser av programmet som man själv distribuerar vidare. Det går till exempel att omvandla ett program under GNU LGPL till GNU GPL utan copyrighthållarens godkännande, men inte omvänt.

Ett program som ligger i Public Domain kan överföras till valfri licens och liknande för X- eller BSD-liknande licenser. Ett exempel på en BSD-liknade licens är villkoren för webbservern Apache som vi har lärt känna tidigare.

Att byta licensvillkor kan vara lite knepigt för fri programvara och öppen källkod då det ligger i modellens natur att det är flera personer som äger copyrighten för delar av ett program. Utan att gräva ner oss djupare i juridiska spetsfyndigheter så kan ändå betrakta ett praktiskt exempel. En del användare av applikationsservern JBoss var tveksamma till den licens som programmet skeppades under nämligen GNU GPL bland annat eftersom de var rädda att deras egna program skulle infekteras av GNU GPL (JBoss, 2004a). Efter en rad diskussioner beslöt sig de huvudsakliga utvecklarna av JBoss att försöka byta licensvillkor till GNU LGPL. Utvecklarna var dock tvungna att kontakta var och en som hade bidragit med källkod till programmet för att få deras godkännande till att inskränka licensvillkoren till GNU LGPL för nya instanser av programmet eller skriva om de delar där upphovsmännen inte samtyckte till ändringen. Eftersom det var en mängd utvecklare som hade bidragit till källkoden var detta en ganska mödosam uppgift som dock löstes och licensen för JBoss är numera GNU LGPL. De förefaller emellertid troligt att utvecklarna såg lite mellan fingrarna och inte inhämtade samtycke till varje liten kodrad som bidragits med till programmet.

7.6 Sammanfattning och slutsats

Frågorna som vi formulerade i inledningen var ”Hur kan FOSS fungera i en kommersiell värld?”, eller löst uttryckt, ”Hur kan man tjäna pengar på något som är gratis?”.

Vi har försökt beakta olika ekonomiska teorier kring intäktsmodeller och affärsstrategier tillsammans med villkoren för fri programvara och öppen källkod. Vi har visat på att FOSS inte primärt handlar om ”gratis” som i utan kostnad utan att essensen istället bättre beskrivs med ordet ”frihet”. Gratisaspekten finns dock där och blir brutalt påtaglig för den som söker profit inom mjukvarubranschen genom att följa gamla affärsvägar.

Låt oss försöka sammanfatta tanken med öppen källkod i en mening. Den grundläggande tanken med Open Source är enkel: *”När programmerare kan läsa, vidaredistribuera och modifiera källkoden för ett program kommer programmet att utvecklas”* (vår översättning) (OSI, 2004d). Det kan hävdas att gåvokulturen inom FOSS-världen på lång sikt gynnar alla inblandade och devisen tycks vara ”a rising tide lifts all boats”. Motståndare till Open Source basunerar ut budskap som exempelvis ”There’s no such thing as a free lunch!”.

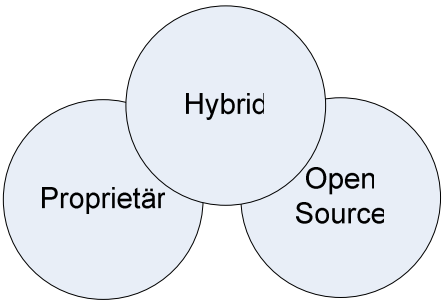
Den som är marknadsledare har inga större incitament att, på kort sikt, överge en befintlig traditionell proprietär modell för licensiering av programvara. Företag kan hyra ut programvara snarare än att sälja licenser som ger rätt att använda programmet i all framtid. I någon mening skulle ett sådant affärssätt säkert kunna ses som ett steg mot att se mjukvara som en tjänst snarare än en artefakt. Det är också ett sätt som passar väl in med begrepp och uttryck som ”out-sourcing”, ”fokusera på kärnverksamheten” och kanske även akronymet ”TCO” som i Total Cost of Ownership

Är det då för kunden någon skillnad om hon hyr in säg en office-miljö baserat på Open Source (exempelvis baserat på Linux och Open Office) eller en mer traditionell proprietär modell (t.ex. Microsoft Windows och Microsoft Office). Beroende på kundens affärsrelationer med mera samt om man kommunicerar med andra företag som kör Office från Microsoft kan det bli mer eller mindre svårt att dela dokument. Andra strulfaktorer vid en övergång skulle kunna inkludera saker som exempelvis konvertering av makron samt bindningar mellan olika program. Dessa punkter kan studeras närmare i exempelvis Stadskontorets (2003) rapport.

Medan ”Open Source”-världen tycks vilja anpassa sig till öppna standarder för exempelvis filformat för dokument behåller Microsoft till exempel filformatet för Microsoft Word stängt och hemligt. Eftersom Microsoft är marknadsledande så är deras strategi att bygga olika former av barriärer både förståelig och försvarbar. Aktörer som inte är nummer ett söker andra sätt att tävla på.

Om produkten inte är unik eller dominerande (med inlåsningseffekter, exempelvis vad gäller användarerfarenheter, kompabilitet med mera) är leverantören sårbar, vilket kan relateras till Porters (1980) ”Five forces”.

För företag som ämnar utveckla ett program finns det tre huvudsakliga vägar att gå. Antingen nyttjas en ren traditionell stängd proprietär modell, eller en hybridmodell där stängd och öppen källkod kombineras i slutprodukten eller så väljs en ren ”Open Source”-modell.



Figur 7.2: *Licensieringsalternativ för mjukvaruutveckling (egen)*

Oavsett modell så kan pengar tjänas på tilläggsprodukter och tjänster kring programvaran. Exempel på detta är teknisk support, utbildning, tilläggsmoduler till program, dokumentation samt konsulttjänster.

Om du som företag äger rätten till all källkod i ditt program kan du ge ut hela programmet under dubbla licenser, något som kan exemplifieras med MySQL AB som ger ut sin produkt under både licensen GNU GPL och en proprietär licens.

Låt oss även erbjuda ett annat synsätt. I stället för att se källkoden som det centrala så kan en leverantör fokusera på sin expertis inom exempel en vertikal. Strategin är här att göra kunden beroende av kunnande snarare än själva programvaran.

Det finns också en intressant aspekt vad gäller företag som har mjukvara som en facilitator, något som lockar folk att köpa deras produkt. Förr rörde det sig om systemleverantörer som sålde hårdvara och skeppade med mjukvaran ”utan extra kostnad”. I andra branscher kan det exempelvis röra sig om rakbladstillverkaren som skänker bort rakhyveln och istället tar betalt för rakbladen eller mobiloperatören som subventionerar telefoner för att tjäna pengar på tjänster. Apple och andra tillverkare ger ut utvecklingsverktyg gratis för att sporra utvecklingen av applikationer till sina system och därigenom öka intresse och värdet på sitt egentliga erbjudande.

Vissa gamla affärsmodeller håller inte rakt av, exempelvis försvinner många mellanhänder då kontakt producent – konsument blir närmare, som i fallet med exempelvis Dell, dock återkommer vissa i form av E-bay med flera. För att vara mer korrekta så handlar det kanske snarare om anpassningar av existerande affärsmodeller än helt nya tankar och idéer. Nya modeller och anpassningar av befintliga modeller kommer efterhand att kristallisera sig.

Det viktigaste för kommersiella företag är troligen inte “gratisaspekten” för fri programvara och öppen källkod, något som även Phipps (2004a) tycks vara inne på

då han skriver *”For many companies, there has been an initial rush based on the abilities of skilled and visionary employees to obtain software gratis from open source communities but increasingly CIOs are realizing this doesn't secure all the freedoms they need for long-term business. They're turning to commercial suppliers to act as their intermediaries with the open source communities - ”they join the community so we don't have to.”* (Phipps, 2004a).

Argumenten i Phipps (2004a) tycks bekräfta att den koordinatorroll som vi föreslagit skulle kunna vara av intresse. Med syftning på att underlätta för företag att hitta bästa produkterna, i vårt fall dock med tillägget att inte behöva knacka koden själv utan ”bara” hitta rätt, bland redan existerande mjukvara eller hyra in rätt personer att göra de anpassningar som behövs.

Slutsatser som vi drar är att:

- Bjud inte fritt ut källkoden till program som är din kärnverksamhet. (Effekten av att ge ut program som Open Source blir att du saboterar din egen marknadsposition)

Comment [f5]: Ej slutsatser?

- Öppna i stället upp produkter som kompletterar dina kärnprodukter. (Detta skulle kunna locka kunder samt andra utvecklare och aktörer till din kärnprodukt eller tjänst)

- Företag kan genom att ge ut sina produkter som fri programvara och öppen källkod påverka sin marknadsposition och sina relationer med andra aktörer

- Intäkter kan för produkter baserade på fri eller öppen mjukvara genereras av support, dokumentation, uppdateringar, utbildning och konsulttjänster med mera

- Basarmodellen kan ge besparingar i utvecklingskostnad för företag om många externa utvecklare kan lockas till produkten

- Genom att plocka in ”Open Source”-licensierade delkomponenter av program till sin egen produkt kan licens- och utvecklingskostnader minskas.

- Program som är tillgängliga som Open Source skulle kunna vara någons kärnverksamhet eller ”kassako”. I den meningen så saboterar Open Source för utvecklande företag.

- Dagens affärsvärld är starkt föränderlig. Det saknas väldefinierade affärsmodeller för att tjäna pengar på FOSS, men företag får efterhand ökad kunskap inom detta område då företag lyckas eller misslyckas med sina affärsidéer inom området. En möjlighet att tjäna pengar på grundtanken bakom Open Source är rollen som koordinator mellan olika aktörer på marknaden för att ta fram bästa möjliga produkter för den enskilde kunden.

Open Source står dock inte ohotat. Licensvillkoren kan uppfattas som snåriga och deras legala värde har egentligen inte mätts i någon större utsträckning. Det finns även frågetecken kring områden som ”patent” och ansvar. Om ett ”Open Source”-program kräver en patentlicens är det inte uppenbart vem som ska garantera eller betala för det. Företaget IBM släppte nyligen 500 av sina patent fria för användning i ”Open Source”-program (IBM, 2005).

Sålunda kan resultatet av denna uppsats sammanfattas med att det finns goda möjligheter att tjäna pengar på FOSS-produkter, men att det gäller att ha en god förståelse för marknaden och dess aktörer för att hitta rätt väg att gå. Vägarna skiljer sig för företag med fokus på att utveckla mjukvara och företag med fokus på att använda mjukvara.

Det utvecklande företaget måste ta hänsyn till licensieringsformer för sin produkt samt potentiella kunder när en ny produkt skall utvecklas för att kunna uppnå lönsamhet, exempelvis kan fokus läggas mer på konsulttjänster och kundanpassningar eller mer på utbildning, support och andra tilläggstjänster till produkten. Med andra ord är det själva ”erbjudandet” till kunden som är det centrala för om ett företag kan uppnå lönsamhet och konkurrensfördelar genom att utveckla FOSS-produkter. Det användande företaget däremot måste i stället hitta ”rätt” mjukvara för sin verksamhet för att stödja sina affärsprocesser. Detta kan vara svårt, och här finns då en affärsmöjlighet för företag som fokuserar på att hjälpa sina kunder att hitta rätt mjukvara. Ett annat perspektiv på FOSS är att se det som möjlighet till att sänka licenskostnader och spara pengar vid utveckling genom bygga vidare på FOSS-komponenter.

7.7 Förslag på vidare studier

Baserat på erfarenheterna från denna undersökning vill ställa upp förslag på vidare studier av området.

- Kvalitativa fallstudier på användning av FOSS i företag och organisationer.
- Kvalitativa fallstudier av projekt som utvecklar FOSS.
- Studier av fenomenet FOSS under en längre tid.
- Analyser av koordinatorrollen i mjukvaruindustrin.
- Undersökning av sociala drivkrafter kontra ekonomiska drivkrafter bakom utveckling av FOSS-program.

Appendix A - Länklista

Nedan har vi listat länkar till webbplatser med information relaterat till Open Source. Listan är på inget vis komplett utan länkarna ska snarare ses som utgångspunkter för den intresserade läsaren.

Linux

- www.linux.org – en av många portaler med länkar till information om linux och olika linux-distributioner.
- http://www.distrowatch.com/ - information och länkar till de flesta linux-distributionerna.

Organisationer för öppen källkod

- http://www.gnu.org – hemsida för GNU-projektet som arbetar för att utveckla ett fritt unix-liknande operativsystem. Även hemsida för FSF och http://www.fsf.org/ länkar till samma hemsida som gnu.org.
- http://www.opensource – hemsida för organisationen Open Source Initiative (OSI) som arbetar för att generellt främja användningen av Open Source och specifikt göra Open Source attraktivt för företag.

Nyheter

- http://www.gnuheter.com/ - en svensk sida med nyheter rörande ”Open Source”-världen.
- http://freshmeat.net/ - stor samling av länkar till Unix- och multiplattformsprogram.
- http://slashdot.org/ - en webbplats med följande slogan, ”News for Nerds. Stuff that Matters.”. Nyhetssida och diskussionsforum för “geeks”, ofta om Open Source och ”motståndarna” (t.ex. Microsoft och patent).

Motpoler till ”Open Source”-rörelsen

- http://www.microsoft.com/resources/sharedsource/default.mspx - lite information Microsoft sätt ”anpassa” sig till Open Source.
- http://www.sco.com/ - ett företag som försöker stämma bland annat IBM för licensbrott i samband med AIX, UNIX och Linux.
- http://www.softwarechoice.org/ - beskriver sig själva som ett globalt initiativ “for promoting neutral government procurement, standards and public R&D policies for software!”. (Har en motpol i webbplatsen: http://www.sincerechoice.org/).
- http://www.neilgunton.com/open_source_myths/ - en artikel som ifrågasätter vissa av tankarna bakom Open Source.

Program som är Open Source

- http://www.sourceforge.net - världens största samlig av ”Open Source”-projekt.
- http://www.perl.org/ - The Perl Foundation (2004), hemsida för skriptspråket Perl
- http://www.apache.org – hemsida för Apache-stiftelsen som bland annat ligger bakom världens mest använda webbserver – Apache Server

Appendix B - Ordlista

Nedan förklaras några av de förkortningar och begrepp som vi använder i uppsatsen.

- ASF - Apache Software Foundation.
- BCG - Boston Consulting Group, en strategi- och managementkonsultfirma
- CSD – Commercial Software Development, Microsofts benämning på traditionell mjukvaruutveckling enligt Eric *Raymond*s ”katedralmodell”.
- CSF - Critical Success Factors, modell för att mäta framgång för ett företag.
- Copyleft – en licenstyp som kortfattat går ut på att den som erhåller en kopia av en copyleftad licens själv är skyldig att ge de som hon i sin tur distribuerar programmet vidare till samma rättigheter som hon själv erhöll vanligen rätt till källkod och rätt att använda programmet till vad hon vill och utan kostnad.
- CPAN - Comprehensive Perl Archive Network, en webbplats som centralt samlar en enorm mängd perl-moduler.
- DEC – Digital Equipment Corporation.
- Emacs – i grunden en texteditor men den är utökningsbar och programmerbar. Ursprungligen skriven av *RMS*.
- EPL - Eclipse Public License.
- ESR – Initialerna för Eric Steven Raymond, en frontfigur för *OSI* och mångtalig förespråkare av Open Source.
- FLOSS – Free Libre Open Source Software, förkortning för mjukvara som är ”fri som i frihet” med öppen källkod.
- FOSS – Free/Open Source Software, ett samlingsnamn och förkortning för fri programvara och öppen källkod.
- Forka – term som används för att beskriva att ett utvecklingsprojekt ”grenar av” sig och utvecklas vidare på olika håll både i meningen av olika personer och med olika inriktningar.
- Fri programvara, se *Free Software*.
- Free Software – fri programvara i mening att användaren har friheten att använda, undersöka, anpassa, distribuera och modifiera programmet. En förutsättning för att rättigheterna ovan ska fungera är att källkoden till programmet finns fritt att tillgå. Om användaren i sin distribuerar programmet vidare måste hon ge mottagaren samma rättigheter.
- Freeware – program som distribueras utan kostnad. Ej att förväxla med *Free Software* eller *Open Source*.
- FSF – Free Software Foundation, en stiftelse som sponsrar *GNU*-projektet och förespråkar *Free Software*.
- GCC - GNU Compiler Collection, en samlig kompilatorer med frontends för flera olika programmeringsspråk och backends för ett stort antal plattformar.
- GNOME - http://www.gnome.org/, en fri grafisk användarmiljö (desktop environment) för UNIX, jämförbar med *KDE*.
- GNU – GNU’s Not UNIX, ursprungligen ett projekt som startades 1984 för att ta fram ett UNIX-liknande operativsystem helt uppbyggd på fri

programvara. Licensen *GPL* har tagits fram för fri programvara och licensen använd.

- GNU/Linux – En del hävdar att *Linux* bara är själva kärnan i operativsystemet och att resten av en Linux-distributition är *(L)GPL*-program och att det därför är mer korrekt att använda termen GNU/Linux.
- GPL – GNU General Public License. En copyleft:ad mjukvarulicens framtagen av *GNU*-projektet.
- IDE - Integrated Development Environment, integrerad utvecklingsmiljö, exempelvis med stöd för editering, kompilering och avlusning av ett program.
- IPR - Intellectual Property Rights, immaterialrätt, det vill säga lagar kring rätten till arbete som man inte direkt kan ”ta på”, exempelvis kan en författare ha upphovsrätt till sin bok eller en utvecklare till sitt program.
- JDS - Java Desktop System, en miljö för kontorsapplikationer från SUN baserat bland annat på *Linux*, *GNOME* och Java.
- KDE – K Desktop Environment, en fri grafisk användarmiljö (desktop environment) för UNIX med en rik uppsättning av applikationer såsom exempelvis officeprogram, webbläsare, utvecklingsmiljöer och mycket mer. (http://www.kde.org/).
- Kernel, operativsystemets kärna som hantera grundläggande funktioner och abstraherar exempelvis åtkomst av hårdvara.
- LGPL – GNU Lesser General Public License. LGPL är en variant av *GPL*.
- Linux – egentligen en *UNIX*-likande *kernel* ursprungligen skapad av Linux Torvalds. Linux används vanligen för att beskriva operativsystem som baserar sig på Linuxkärnan och *GNU*-program. Ibland kallas Linuxpaket därför för GNU/Linux.
- LUG - Linux Users Group, sammanslutning av linux-användare som bland annat ger linux-support till varandra och hjälper nybörjare att komma igång med *linux*.
- MPL – Mozilla Public License, efterföljaren till *NPL*, finns revision 1.0 och 2.0.
- NPL – Netscape Public License, den licens under vilken de första versionerna av Netscape med öppen källkod släpptes under.
- OSD - Open Source Definition, kriterier som program måste uppfylla för att få räknas som Open Source, framtagna av organisationen *OSI*.
- Open Source, öppen källkod, till detta räknas program som uppfyller villkoren i *OSD*.
- OSI – Open Source Initiative, ett icke-vinstdrivande företag som arbetar för att generellt främja användningen av Open Source och specifikt göra Open Source tilltalande i kommersiella sammanhang.
- OSS – Open Source Software, Microsofts benämning på det Raymond kallar ”basarmodellen” eller utveckling enligt Open Source.
- POSIX - Portable Operating System Interface, specifikation för operativsystem baserat på *UNIX*.
- Public domain, saker, exempelvis program, utan aktuell upphovsman eller där upphovsmannen avsagt sig sin rätt och så att säga donerat sitt verk den till allmänheten. Ej att förväxla med *Free Software* eller *Open Source*.
- Raymond Eric – Se *ESR*.

- RMS - Richard M Stallman, grundare och förgrundsfigur för *FSF* och *GNU*-projektet. Upphovsman till bland annat *Emacs*.
- Shareware, program som är gratis att använda typiskt under en begränsad tid. Efter det att prövotiden gått ur skall en avgift erläggas till upphovsmannen. Ska inte förväxlas med *Free Software* eller *Open Source*.
- SSI – Shared Source Initiative, Microsofts sätt att ge utvecklare från andra företag tillgång till Microsofts källkod för att utveckla ny/förbättrad mjukvara.
- Stallman Richard – Se *RMS*.
- TCO – Total Cost of Ownership, ekonomisk modell för att beräkna totalkostnad.
- UNIX – Ett operativsystem som ursprungligen skapades av Bell Labs 1969. *Linux*, BSD och Solaris är exempel på operativsystem härstammar från UNIX.
- WWW - World Wide Web. W3, webben.

Appendix C – Källförteckning

Källorna i företeckningen nedan är indelad i de två kategorierna böcker respektive webbreferenser.

Böcker

- Anderberg, Martin & Dahlberg, Magnus (1999): ”Säkerhet i Linuxbaserade system”, Lunds Universitet, Institutionen för Informatik, Magisteruppsats

- Bach, Marice J (1986): “Design of the Unix Operating System”, 1 edition (May 27, 1986). Prentice Hall PTR

- Bjurwill, Christer (2001): ”A, B, C och D – vägledning för studenter som skriver akademiska uppsatser”, Studentlitteratur, Lund

- Andreessen, Marc (1995): Marc Andreessen, citerad av Chip Bayers i ”Why Bill Gates Wants To Be the Next Marc Andreessen”, Wired, December 1995)

- Bassellier G, Benbasat I (2004): “Business Competence of Information Technology Professionals: Conceptual Development and Influence on IT-Business Partnerships”, MIS Quarterly Volume 28, Number 4, December 2004, Minneapolis USA

- Bell, Judith (1995): ”Introduktion till forskningsmetodik”, andra upplagan. Studentlitteratur, Lund.

- Bjerstedt, Åke (1997): ”Rapportens yttre dräkt”, Studentlitteratur, Lund

- Bonnier (1996): ”Bonniers Lexikon 11”, Bonnier lexikon AB, Mladinska Knjiga, Ljubljana.

- Clayberg, Eric, Rubel, Dan (2004): “Eclipse, Building Commercial Quality Plug-ins”, Addison-Wesley, USA

- Dahlander, Linus (2004), ”Networks of Innovators and the Private-Collective Innovation Model: Why Do Firms Engage in Open Source Software?”, Göteborg, Reproservice CTH, Chalmers University of Technology

- Daniel, R.H. (1961): “Management data crisis“, Harvard Business Review. Sept-Oct

- Denscombe, Martyn (2000): ”Forskningshandboken – för småskaliga forskningsprojekt inom samhällsvetenskaperna”, Studentlitteratur, Lund

- Fowler, Martin (2002): “Patterns of enterprise application architecture”, Addison-Wesley. Boston USA.

- Gamma, Erich, Helm, Richard, Johnson, Ralph, Vlissides, John (1995): ”Design Patterns”, Addison-Wesley, USA

- Hedman, Jonas, Kalling, Thomas (2002): “IT and Business Models”, Malmö, Liber ekonomi

- Hamel, Gary, Prahalad C.K. (1996): “Competing for the future”, Harvard Business School Press, Boston USA

- Hamlyn, D.W. (1994): ”Filosofins historia”, Bokförlaget Thales, Stockholm

- Hansson, Bengt, utg (1992): ”Metod eller anarki - moderna teorier om vetenskapsens väsen och metoder”, (2:a upplagan). Filosofiska Institutionen, Lund

- Kaplan R S, Norton D P (1992): “The Balanced Scorecard – measures that drive performance”, Harvard Business Review jan – feb

- Kotler, Philip (1999): ”Kotlers marknadsföring - Att Skapa, Vinna Och Dominera Marknader”, Liber AB

- Kotler, Philip, Trias de Bes, Fernando (2003): ”Lateral marknadsföring”, svensk översättning av Richard Gustafson, Pagina Förlags AB/Optimal Förlag

- Lerner, Josh, Tirole, Jean (2000), “The simple economics of Open Source”, National Bureau of Economic Research, Cambridge, Working Paper 7600

- Ljung, Nilsson, Olsson et al (1994): ”Företag och Marknad – Samarbete och konkurrens”, Studentlitteratur, Lund

- Ljung, Nilsson, Olsson et al (2001): ”Företag och Marknad – Flexibilitet och förändring”, Studentlitteratur, Lund

- Lunell, Hans (1991): ”Datalogi – en inledande översikt”, Studentlitteratur, Lund

- Miles M B, Huberman A M (1994) : "Qualititative Data Analysis", Sage Publications, London

- Moody, Glyn (2001): "Rebel Code – Inside Linux and the Open Source Revolution", Perseus Publishing, USA.

- Morgan, Gareth (1997): "Organisationsmetaforer (Images of Organization)", Studentlitteratur 1999

- Nilsson, Kristoffer (1998): "Operativsystem", Lunds Universitet, Institutionen för Informatik, Kandidatuppsats

- Nordström, Rickard, Schmidt, Peder (2000): "Open-source: En studie av metoden ur ett kvalitetsperspektiv", Lunds Universitet, Institutionen för Informatik, Magisteruppsats

- Olofsson, Jessica (2003a): "Upphovsrättsliga aspekter på licenser för fri programvara och öppen källkod", IRI-rapport 2003:1, Institutet för rättsinformatik. Stockholms universitet, Book-on-demand,Visby

- Patel, R. & Davidson, B. (1991): "Forskningsmetodikens grunder. Att planera, genomföra och rapportera en undersökning", Studentlitteratur, Lund

- Penrose, E.T (1959): "The theory of the growth of the firm", Oxford University Press, Oxford

- Porter, Michael E (1980): "Competitive Strategy", New York, Free Press

- Porter, Michael E (1985): "Competitive advantage", New York: Free Press.

- Raymond, Eric S (2001a): "The Cathedral and the Bazaar", O'Reilly & Associates Inc., 2001

- Rockart, J.F. (1979): "Chief executives define their own data needs", Harvard Business Review, 57 (2)

- Rosenberg, Donald K (2000): "Open Source – The Unauthorized White Papers", M&T Books, USA.

- Schmidt, Douglas, Stal, Michael, Rohnert, Hans, Buschmann, Frank (2000): "Pattern-oriented software architecture volume 2", John Wiley & Sons, New York USA

- Wade, M, Hulland, J (2004): "The Resource-Based View and Information Systems Research: Review, Extension, and Suggestions for Future Research", MIS Quarterly Volume 28, Number 1, March 2004, Minneapolis USA

- Wall, Larry, Christiansen, Tom, Orwant Jon (2000): "Programming Perl - Third Edition", O´Reilly, USA

- Williams, Sam (2002): "Free as in Freedom, Richard Stallman's Crusade for Free Software", O´Reilly, USA

Webbreferenser

- Agilealliance (2004a): "AgileAlliance", http://www.agilealliance.org/programs/roadmaps/Roadmap/index.htm, 2004-12-12

- Apache (2004a): "About the Apache HTTP Server Project - The Apache HTTP Server Project", http://httpd.apache.org/ABOUT_APACHE.html, 2004-01-27

- Apache (2004b): "Foundation Project - The Apache Software Foundation", http://www.apache.org/foundation/, 2004-01-27

- Apache (2004c): "Apache License, Version 2.0 - The Apache Software Foundation", http://www.apache.org/licenses/LICENSE-2.0, 2004-12-29

- Apache (2004d): "Licenses - The Apache Software Foundation", http://www.apache.org/licenses/, 2004-12-29

- Baker, Steve (2004a): "A Complete History of Tux. (So far)", http://www.sjbaker.org/tux/, 2004-10-10

- Balancedscorecard (2004a): "What is the Balanced Scorecard?", www.balancedscorecard.com, 2004-12-09

- BCG (2005): The Growth Share Matrix, http://www.bcg.com/publications/publication_view.jsp?pubID=720&language=English (2005-01-16)

- Bell-labs (2004a): "The Creation of the UNIX Operating System: Bell Labs Early Contributions to Computer Science", http://www.bell-labs.com/history/unix/blcontributions.html, 2004-09-02

- Bell-labs (2004b): "Chistory", http://cm.bell-labs.com/cm/cs/who/dmr/chist.html, 2004-02-02

- Cybersource (2002):" Linux vs. Windows Total Cost of Ownership Comparison", http://www.cyber.com.au/cyber/about/linux_vs_windows_tco_comparison.pdf, 2004-08-09

- Bezroukov, Nikolai (2004): "Nikolai Bezroukov. Portraits of Open Source Pioneers. Ch 4: A Slightly Skeptical View on Linus Torvalds", http://www.softpanorama.org/People/Torvalds/index.shtml, 2004-09-01

- Coleman, Gabriella (2004a): "Political Agnosticism Open Source, Politics of Contrast", http://www.linuxinsider.com/story/36906.html, 2004-09-28

- CPAN (2004a): "Comprehensive Perl Archive Network", http://www.cpan.org, 2004-11-25

- CPAN (2004b): "The CPAN Frequently Asked Questions", http://www.cpan.org/misc/cpan-faq.html, 2004-11-25

- Debian (2004a): "A brief history of Debian", http://www.debian.org/doc/manuals/project-history/ch-intro.en.html, 2004-10-08

- Debian (2004b): "About Debian", http://www.debian.org/intro/about, 2004-10-10

- Distrowatch (2004a): "DistroWatch.com: Put the fun back into computing. Use Linux", BSD, http://distrowatch.com/, 2004-11-25

- Eclipse Foundation (2004a): "Eclipse Public License - Version 1.", http://www.eclipse.org/legal/epl-v10.html, 2004-09-27

- Eclipse Foundation (2004b): "CPL to EPL Transition Plan", http://www.eclipse.org/legal/CPL2EPLTransitionPlan.pdf, 2004-12-29

- E-mule (2004a): "Official Homepage of eMule", http://www.emule-project.net/, 2004-05-31

- Europeiska kommissionen (2004): "Immaterialrätt", http://europa.eu.int/business/sv/topics/ipr/index.html, 2004-06-22).

- FFII - The Foundation for a Free Information Infrastructure (2004a): "FFII: Logic Patents in Europe", http://swpat.ffii.org/index.en.html, 2004-11-29

- FSF (2004a): "The Free Software Definition - GNU Project - Free Software Foundation (FSF)", http://www.fsf.org/philosophy/free-sw.html, 2004-06-08

- FSF (2004b): "GNU's Who – GNU Project – Free Software Foundation (FSF)", http://www.fsf.org/people/people.html, 2004-10-08

- FSF (2004c): "Free Software Foundation - GNU Project - Free Software Foundation (FSF)", http://www.fsf.org/fsf/fsf.html, 2004-11-24

- GameTheory.net (2004a): “Game of Chicken”, http://www.gametheory.net/Dictionary/Games/GameofChicken.html, 2005-01-04

- GNOME (2004a): “What is GNOME?”, http://www.gnome.org/about/, 2004-12-16

- GNU (2004a): ”Categories of Free and Non-Free Software”, http://www.gnu.org/philosophy/categories.html, 2004-01-23

- GNU (2004b): “Licenses”, http://www.gnu.org/licenses/licenses.html#WhatIsCopyleft, 2004-02-03

- GNU (2004c): ”The Free Software Definition”, http://www.gnu.org/philosophy/free-sw.html, 2004-08-11

- GNU (2004e): ”Why ``Free Software" is better than ``Open Source"”, http://www.gnu.org/philosophy/free-software-for-freedom.html, 2004-11-29

- GNU (2004f): ”Overview of the GNU Project”, http://www.gnu.org/gnu/gnu-history.html, 2004-11-29

- GNU (2004g): “GNU General Public License”, http://www.gnu.org/licenses/gpl.html, 2004-11-29

- GNU (2004h): ”GNU Lesser General Public License”, http://www.gnu.org/licenses/lgpl.html, 2004-11-29

- GNU (2004i): “Some Confusing or Loaded Words and Phrases that are Worth Avoiding”, http://www.gnu.org/philosophy/words-to-avoid.html, 2004-12-16

- Goldman, Mike (2000); “[CrackMonkey] GPL game theory”, http://www.crackmonkey.org/pipermail/crackmonkey/2000q3/014251.html, 2004-12-12

- Greve, Georg C. F. (1999a): “Georg´s Brave GNU World.”, http://www.gnu.org/brave-gnu-world/issue-10.en.html, 2004-06-07

- Groklaw (2004a): ”The GPL is a License, Not a Contract, Which is Why the Sky Isn't Falling”, http://www.groklaw.net/article.php?story=20031214210634851, 2004-02-10

- Görling, Stefan (2003a): "A critical approach to Open Source software", http://opensource.mit.edu/papers/gorling.pdf, 2005-01-05

- Hertel G, Niedner S, Herrmann S (2003): "Motivation in Open Source Software Projects - An Internet-based Survey of Contributors to the Linux Kernel", http://opensource.mit.edu/papers/rp-hertelniednerherrmann.pdf, 2005-01-25

- IBM (2001a): "WebSphere Pressrelease - IBM donates $40 million of software to open source community", http://www-306.ibm.com/software/info1/websphere/news/ibmnews/pr011105a.jsp, 2004-09-06

- IBM (2004a): "WebSphere", http://www-136.ibm.com/developerworks/websphere/, 2004-05-31

- IBM (2005): " IBM Statement of Non-Assertion of Named Patents Against OSS", http://www.ibm.com/ibm/licensing/patents/pledgedpatents.pdf, 2005-01-21

- International Institute of Infonomics, University of Maastricht and Berlecon Research GmbH (2002a): "FLOSS FINAL REPORT", http://www.infonomics.nl/FLOSS/report/, 2004-06-09, International Institute of Infonomics

- IPR-Helpdesk (2004a): "IPR-Helpdesk: Intellectual Property", http://www.ipr-helpdesk.org/controlador.jsp?cuerpo=notaDescripcionIP&seccion=principal&cod_nodo_padre=t_01&len=en, 2004-11-29

- Irvine, Martin (2004a): "Network Effects and the University", http://www.georgetown.edu/faculty/irvinem/articles/neteffects.html, 2004-12-12

- JBoss (2004a): "JBoss: Professional Open Source", http://www.jboss.org/index.html, 2004-10-10

- JBoss (2004b): "JBoss Inc. - The Professional Open Source Advantage", http://www.jboss.com/pdf/open_source_short_0704.pdf, 2005-01-09

- Jboss (2004c): "JBoss Enterprise Middleware System (JEMS)", http://www.jboss.com/products/overview, 2005-01-09

- Jones, Russel (2004a): "Open Source Is Fertile Ground for Foul Play", http://www.devx.com/opensource/Article/20111, 2004-11-30

- KDE (2004a): "What is KDE?", http://www.kde.org/whatiskde/, 2004-12-16

- Kenwood, Carolyn A. (2001a): "A Business Case Study of Open Source Software", http://www.mitre.org/work/tech_papers/tech_papers_01/kenwood_software/index.html, 2004-11-24, The MITRE Corporation

- Linux.com (2004a): "Linux.com: The Enterprise Linux Resource", http://www.linux.com, 2004-11-25

- Linux.org (2004a): "The Linux Home Page at Linux Online", http://www.linux.org/, 2004-11-25

- Locke, John (1698): "Two Treatises of Government", http://www.lonang.com/exlibris/locke/loc-205.htm, 2004-10-10

- Lowendpc (2004): "Personal Computer History", http://www.lowendpc.com/history/index.shtml, 2004-01-23

- Lysator (2004a): "Fred Fisk Disk collection", http://www.lysator.liu.se/amiga/info/guide/amigafaq2.guide?software.fish, 2004-09-27.

- McCain, Roger A. (1997a): "Prisoners' Dilemma", http://william-king.www.drexel.edu/top/eco/game/dilemma.html, 2005-01-04

- Marini, Joe (2004a): "Not Opening Everything - Some Things are Meant to be Secret", http://www.joemarini.com/articles/notOpeningEverything20041121.php, 2005-01-02

- Microsoft (2003a): "Basic Principles of Software Source Code Licensing", http://www.microsoft.com/resources/sharedsource/Articles/LicensingBasics.mspx, 2004-06-22.

- Microsoft (2004a): "Microsoft Shared Source Initiative Overview", http://www.microsoft.com/resources/sharedsource/Initiative/Initiative.mspx, 2004-07-27.

- Microsoft (2004b): "END-USER LICENSE AGREEMENT FOR MICROSOFT SOFTWARE (Microsoft Office 2003)", http://download.microsoft.com/download/1/2/5/12538ba0-3d24-4f00-aab1-dd9ff4aacfc9/en_client_eula.pdf, 2004-11-29

- Microsoft (2004c): ”GNU General Public License (GPL)”, http://www.microsoft.com/resources/sharedsource/Articles/GNU.mspx, 2004-11-29

- Microsoft (2004d): “Licenser – Microsoft AB”, http://www.microsoft.com/sverige/license/, 2005-01-25

- Mossberg, Walter S (2004a): “Personal Technology from The Wall Street Journal - Cool Features Of Firefox Web Browser Beat Microsoft's IE”, http://ptech.wsj.com/archive/ptech-20041230.html, 2005-01-06

- Mozilla (2004a): “Trademarks and Licenses”, http://www.mozilla.org/foundation/licensing.html, 2004-08-11

- Mozilla (2004b): ”Mozilla Relicensing FAQ”, http://www.mozilla.org/MPL/relicensing-faq.html, 2004-08-11

- MySQL (2004a): ”MySQL Product”, http://www.mysql.com/products/index.html, 2004-05-31

- MySQL (2004b): ”MySQL Commercial License”, http://www.mysql.com/company/legal/licensing/commercial-license.html, 2004-11-25

- Nationalencyklopedin (2004a): ”Spelteori”, http://www.ne.se/jsp/search/article.jsp?i_art_id=312650&i_word=Speltteori, 2004-04-28

- Netcraft (2004a): ”Netcraft: November 2004 Web Server Survey”, http://news.netcraft.com/archives/2004/11/01/november_2004_web_server_survey.html, 2004-11-25

- Ny Teknik (2004a): ”Tjäna pengar på givmildhet”, http://nyteknik.se/pub/ipsart.asp?art_id=34640, 2004-08-09

- Olerup, Agneta & Steen, Odd (2004a): ”Utformning av rapporter i informatik”, http://www.ics.lu.se/studentservice/kurser/INF630/rapport.pdf, 2004-12-13

- OpenBSD (2004a): ”OpenBSD Copyright Policy”, http://www.openbsd.org/policy.html, 2004-08-11

- Open Office (2004a): ”Mission Statement”, http://www.openoffice.org/about.html#mission, 2004-05-31

- Open Office (2004b): "License Page", http://www.openoffice.org/license.html, 2004-12-29

- O'Reilly, Tim (2004a): "he Open Source Paradigm Shift", http://tim.oreilly.com/opensource/paradigmshift_0504.html, Tim O'Reilly, 2004-07-31.

- OSI (2004a): "OSI History", http://www.opensource.org/docs/history.php, 2004-08-11, Open Source Initiative (OSI) – *OSI (2004a)*

- OSI (2004b): "Open Source Initiative (The Open Source Definition, Version 1.9", http://opensource.org/docs/def_print.php, 2004-02-03

- OSI (2004c): "Open Source Initiative OSI - Why "Free" Software is too Ambiguous:Advocacy", http://www.opensource.org/advocacy/free-notfree.php, 2004-09-02

- OSI (2004d): "Open Source Initiative OSI - Advocacy", http://www.opensource.org/advocacy/, 2004-10-10

- OSI (2004e): "Open Source Initiative OSI - Welcome", http://www.opensource.org/, 2004-11-24

- OSI (2004f): "Open Source Initiative OSI - Halloween Documents", http://www.opensource.org/halloween/, 2004-12-12

- Perens, Bruce (1999): "Open Sources: Voices from the Open Source Revolution - The Open Source Definition", http://www.oreilly.com/catalog/opensources/book/perens.html, 2004-08-11

- Perl.org (2004a): "The Timeline of Perl and its Culture v3.0_0505", http://history.perl.org/PerlTimeline.html, 2004-11-25

- Phipps, Simon (2004a): "How Will Companies Ever Make Money Off Open-Source?", http://www.sys-con.com/story/?storyid=46131, 2004-09-01

- Rappa, Michael (2004a): "Business Models on the Web", http://digitalenterprise.org/models/models.html, 2004-12-12

- Recklies, Dagmar (2001): "Porters Five Forces", http://www.themanager.org/Models/p5f.htm, 2005-01-02

- Ross, Don (2004a): "Game Theory", http://plato.stanford.edu/entries/game-theory/, 2005-01-04

- Rysdam, David (2000): "Open Source as ESS", http://web.archive.org/web/20010427134243/www2.fastdial.net/~drysdam/essays/GPL-as-strategy.html, 2004-12-12

- Slackware (2004a):"The Slackware Linux Project: Slackware Linux Essentials", http://www.slackware.com/book/index.php?source=x68.html, 2004-11-29

- Smith, Adam (1776): "An Inquiry into the Nature and Causes of the Wealth of Nations", http://www.econlib.org/library/Smith/smWN.html, 2004-10-10

- SPCS (2004a): "SPCS - Scandinavian PC Systems - förenklar företagets lön, bokföring och administration", http://www.spcs.se, 2004-11-25

- SPCS (2004b): "Standardavtal för abonnemang / service", http://www.spcs.se/butik/licensavtal_new.htm, 2004-11-25

- Stallman, Richard (2004a): "Richard Stallman's Personal Home Page", http://www.stallman.org/, 2004-10-08

- Stallman, Richard (2004b): "GNU/Linux FAQ", http://www.gnu.org/gnu/gnu-linux-faq.html, 2004-10-10

- Stallman, Richard (2004c): "Linux and the GNU Project", http://www.gnu.org/gnu/linux-and-gnu.html, 2004-10-10

- Stanford (2003a): "Fair Use - Public Domain Trouble Spots", http://fairuse.stanford.edu/Copyright_and_Fair_Use_Overview/chapter8/8-b.html, 2004-02-06)

- Statskontoret (2003a): "Statskontoret 2003:8", http://www.statskontoret.se/pdf/200308.pf, 2004-06-09

- SUN (2004a): "The source for developers", http://developers.sun.com, 2004-11-30

- SUN (2004b): "OpenOffice.org", http://wwws.sun.com/software/star/openoffice/, 2004-12-16

- SUN (2004c): "StarOffice 7 Office Suite", http://wwws.sun.com/software/star/staroffice/, 2004-12-16
- Susning (2004a): "Spelteori", http://susning.nu/Spelteori, 2004-04-28
- The History of Computing Foundation (2002): "Linus Thorvalds", http://www.thocp.net/biographies/torvalds_linus.html, 2004-08-31
- The Linux Counter (2004): "Estimating the number of Linux users", http://counter.li.org/estimates.php, 2005-01-25
- Tysver, Daniel A. (2000): "Why Protect Software Through Patents", http://www.bitlaw.com/software-patent/why-patent.html, 2004-11-29
- UCAR (2004): "UCAR Legal Services", http://www.fin.ucar.edu/legal/publicdomain.html, 2004-06-09
- Wheeler, David A. (2004a): "Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers!", http://www.dwheeler.com/oss_fs_why.html#tco, 2004-11-30
- Wheeler, David A. (2004b): "Open Source Software / Free Software (OSS/FS) References", http://www.dwheeler.com/oss_fs_refs.html, 2004-10-09
- Winzip (2004a): "WinZip Evaluation License", http://www.winzip.com/elicense.htm, 2004-11-29
- Wong, Kenneth & Sayo, Phet (2004a): "Free/Open Source Software – A General Introduction", http://www.iosn.net/foss/foss-general-primer/foss_primer_current.pdf, 2004-09-28
- Xprogramming (2004a): "What is Extreme Programming?", http://www.xprogramming.com/xpmag/whatisxp.htm, 2004-12-12