GÖTEBORGS UNIVERSITET
Institutionen för Informatik | Mars 2003

OPEN SOURCE SOM AFFÄRSIDÈ

PENGAR, POLITIK ELLER BARA PROGRAMVARA

Magisteruppsats i Informatik skriven av Mats Larsson.
Handledare: Anna Maria Szczepanska, Viktoriainstitutet, Göteborg.

Open Source som affärsidé

”And this was the beginning of the Fourth Age, the age of Open Source.”
Regan (1999).

Magisteruppsats (20 poäng) i informatik av Mats Larsson.
Institutionen för informatik, Handelshögskolan vid Göteborgs universitet.

Abstrakt

Denna magisteruppsats presenterar resultatet av en etnografisk intervjustudie av fyra svenska företag. Den gemensamma nämnaren för företagen i studien är att de har Open Source som grund för sin verksamhet. Uppsatsen fokuserar på vilka affärsidéer som företagen har för att dra nytta av Open Source och jämnför dessa med internationellt praktiserade idéer. Fokus är även vilka fördelar och nackdelar de ser med Open Source och på vilket sätt de förhåller sig till Open Source-rörelsen.

Resultatet visar att företagen i huvudsak inriktat sig på kringtjänster, service och supporttjänster kopplade till Open Source-mjukvara. Vidare visar resultatet att flera av företagen prioriterar ett gott förhållande till Open Source-rörelsen. Detta innebär att man antingen startar och driver olika Open Source-projekt eller bidrar till redan existerande projekt. De styrkor som de ser i Open Source-mjukvara är främst öppenhet, inkrementell utveckling och en stor extern utvecklarbas. Förutom dessa resultat presenteras också en affärsidé som används av ett av företagen och som bygger på utbildning och konstruktion av infrastrukturella förutsättningar för utvecklingsarbete baserat på Open Source-liknande metoder.

Nyckelord: Open Source, öppen källkod, fri programvara, affärsidéer och programvaruutveckling.

Abstract

This Master's thesis presents the result of an ethnographic interview study conducted at four Swedish companies with use of Open Source as the common denominator. The focus of the thesis is the business ideas used by the companies to make economic use of Open Source. The business ideas found are compared to internationally employed ideas. Focus is also set on the pros and cons of using Open Source Software seen by the companies and in what way they relate to the Open Source movement.

The result shows that the companies are mainly focused on custom development, post-sales support and service. Further more the companies demonstrated a wish to have a good relation with the Open Source community. This is fulfilled either by contributing to existing projects or by starting and maintaining there own ones.

The benefits of using Open Source, from the companies' views, can be summarized with openness, lowered costs, continuity, incremental development and a large base of developers. Apart from these results yet another business idea is presented found. This idea is based upon the education and construction of infrastructural prerequisites for development based on Open Source methods.

Keywords: Open Source, Free Software, Business Ideas and Software Development.

Förord

Denna magisteruppsats är resultatet av de studier och intervjuer jag genomfört under våren och hösten 2002.

Jag vill tacka min handledare Anna Maria Szczepanska, doktorand i sociologi vid Viktoriainstitutet, Göteborgs universitet, för det stöd hon gett mig i skrivandet. Ett stort tack går givetvis också till de personer som jag intervjuat. Utan deras välvillighet att berätta om sina respektive företag och sitt arbete skulle denna uppsats aldrig blivit verklighet.

Slutligen vill jag även tacka Anneli och min familj, inklusive metervaran och tillika taxen Frasse, för stöd och glada tillrop under skrivandet av uppsatsen. Även Annelis mormor förtjänar ett tack, då hon efterlämnade en viktig sak till förmån för mitt skrivande: kaffekvarnen.

Göteborg, mars 2003

Mats Larsson

Innehållsförteckning

Tabeller och illustrationer

Kapitel 1 – Introduktion

Open Source. Direkt översatt till svenska blir begreppet ”öppen källa”. Oftast översätts dock begreppet med ”öppen källkod”. Källkod kan sägas vara ett datorprogram i dess grundform, den form då det fortfarande är möjligt för en programmerare att läsa, förstå och ändra programmets funktioner. Öppen källkod innebär att det inte bara är möjligt för programmerare att läsa och ändra källkoden, utan att det till och med uppmuntras av programmets skapare.

Motpolen till Open Source, öppen källkod, är sluten källkod, ofta likställt med proprietär programvara. Den slutna källkoden innebär att källkoden inte sprids i en form som är läsbar för programmerare utan endast för en dator. Den slutna källkoden har kompilerats, anpassats för körning på en dator i form av ett färdigt program, innan den sprids. Detta gör det svårt för den intresserade programmeraren att läsa och ändra programmets funktioner. Oftast är det inte heller tillåtet på grund av den licens som programmets sprids under.

För att tydliggöra dessa två poler kan man ta två program som exempel: webbservern Apache respektive Microsoft Internet Information Server (IIS). Apache är ett exempel på Open Source Software (OSS) och sprids alltså i dels i en form som är läsbar för andra än de programmerare som skapat källkoden, dels i en form som är körbar av en dator. Som en följd av att vem som helst har rätt att se källkoden till Apache finns denna programvara att hämta gratis från Internet. IIS å andra sidan är proprietär programvara: den kostar pengar att använda och skaparen, Microsoft, bestämmer vilka som eventuellt får se källkoden. Båda programmen utför i princip samma sak: de möjliggör för människor att publicera webbplatser på Internet.

Hårddrar man exemplet ovan har vi alltså två poler: gratis och inte gratis. Riktigt så enkelt är dock inte. Förespråkarna för Open Source menar att det inte i första hand handlar om kostnadsfrågan, även om det ofta är en positiv bieffekt, utan om den frihet det ger programmerare och systemutvecklare i form av tillgång till källkoden och rätten att vidareutveckla den samma. Om en programmerare hittar ett fel i Apache kan hon eller han hämta hem

källkoden till programmet och korrigera felet och sedan sprida sin förbättring till andra som använder Apache. Man skulle kunna kalla det för hjälp till självhjälp. Om samme programmerare däremot hittar ett fel i IIS är det i princip omöjligt för denne att lokalisera och korrigera felet. Det enda programmeraren kan göra är att påtala felet för skaparen, Microsoft, och hoppas att denne korrigerar felet.

En av grunderna i Open Source kontra proprietär programvara är licenserna: det avtal som sluts mellan skapare och användare av programmet. När du handlat ett program från, till exempel Microsoft eller Adobe, har du köpt programmet under en licens. Licensen beskriver vad du får och inte får göra med programmet, du får tillexempel inte, som i exemplet ovan, försöka ändra i programmets källkod och sprida det vidare. Du får inte heller ta kopior av programmet och sprida det vidare. Skulle du, i strid med licensen, göra detta kallas det piratkopiering vilket i sin tur är straffbart. Nyckeln till ovanstående är upphovsrätten. Den ger möjlighet för företag som Microsoft och Adobe att ha egenrätt på sina program och sälja den körbara varianten av programmen för att tjäna pengar, betala vidare utveckling av program och få avkastning. I Open Source-fallet utnyttjas inte denna del av upphovsrätten: programmen sprids ju även i källkodsform och licensen som programmen sprids under säger att vem som helst har rätt att sprida dem.

Detta ger upphov till en, till synes kanske besynnerlig, paradox för de företag som sysslar med utveckling av Open Source-programvara (OSS). Hur skall de kunna tjäna pengar på något som distribueras gratis och som vem som helst får sprida och ändra? Det ger också upphov till en konflikt mellan de som sysslar med proprietär programvara och de som sysslar med OSS. Microsofts VD, Steve Ballmer, har tillexempel jämnfört OSS med en cancer som smittar ner mjukvaruindustrin och sätter kapitalismens vanliga regler ur spel (Newbart, 2001). Förespråkarna för OSS menar dock ofta att det handlar om ärlighet: kunden skall få med ritningarna, källkoden, till det program den köpt. Ofta nämns också argument som transparens, möjligheten att vidareutveckla, förändra, återanvända och förbättra programmen med hjälp av källkoden (se kapitel 3.3).

En stor samarbetsorganisation för programvaruföretag, Business Software Alliance (BSA), menar att ”Programvara är en av informationsålderns värdefullaste redskap som driver allt från persondatorer till Internet” (BSA, 2002). Just värdet, det vill säga den ekonomiska vinning som kan fås från programvaran genom försäljning, är något som skiljer OSS och proprietär programvara. Detta till trots har företag världen över lyckats nå kommersiella framgångar, i varierande grad, med hjälp av OSS. Företag som Cygnus, Red Hat har genom Open Source-prgramvara och egna varianter av operativsystemet Linux nått ekonomiska framgångar. Även stora företag som IBM har under de senaste åren riktat blickarna mot OSS. Open Source verkar få allt starkare fäste inom många områden och regeringar i länder som Tyskland, Frankrike, Argentina, med flera har i olika grad diskuterat krav på användande av Open Source inom statlig förvaltning (News.com, 2001).

Framgångarna som nämnts ovan har mötts av hårt motstånd av företag som Microsoft. Som nämndes ovan har ledande personer inom företaget liknat Open Source med mer eller mindre fördelaktiga företeelser.

Med detta som grund anser jag det vara intressant att närmare studera svenska företag som stödjer sin ekonomiska framgång på Open Source. Gör OSS i sig att de skiljer sig från andra företag? Speglas till exempel de stundtals starka ideologiska inslagen inom Open Source-rörelsen i företagens verksamhet och historia?

Min studie består dels av empirisk del där jag intervjuat företrädare för fyra svenska företag som sysslar huvudsakligen med OSS, dels består studien av en teoretisk del där jag utifrån litteratur placerar de svenska företagen i en internationell kontext med fokus på deras affärsidéer.

1.1 Syfte

Syftet med denna uppsats är dels att försöka skapa en bild av hur svenska Open Source-företag verkar på marknaden. Ett syfte är också att studera

vilka affärsidéer företagen har och vilka problem respektive fördelar de ser med att basera sin verksamhet på OSS.

1.2 Problem och frågeställningar

I dagsläget har jag inte kunnat hitta någon forskning som studerat svenska Open Source-företag med fokus på hur verksamheten ter sig. Det jag frågar mig i denna uppsats är vilka affärsidéer svenska Open Source-företag har och hur de tjänar pengar på Open Source Software och vilka problem respektive positiva faktorer de ser i sin verksamhet.

Hur beskriver de studerade företagen sin verksamhet och finns det särskiljande faktorer i deras verksamhet som kan härledas till arbetet med OSS.
Hur ser de studerade företagens affärsidéer ut om man ser det i sken av de affärsmodeller som finns beskrivna för utländska OSS-företag.

1.3 Mål och förväntat resultat

Målet med uppsatsen är att kunna presentera vilka affärsmodeler de studerade företagen arbetar utifrån och jämföra dessa med modeller som observerats utanför Sverige. Ett mål är också att försöka utröna vilka fördelar och nackdelar företagen ser med att använda Open Source som grund för verksamheten.

Resultatet av uppsatsen blir förhoppningsvis en bild av Open Source-företag i Sverige och deras kommersiella bas som dels kan fungera som en orientering i ämnet men också en grund för vidare forskning genom att visa på tidigare ej undersökta aspekter av Open Source.

1.4 Avgränsning

Denna uppsats syftar inte till att vara en uppsats *om OSS*, utan snarare om *hur företag kan nyttja* Open Source. I kapitel tre ges dock en bakgrund till

Open Source Software för att försöka visa hur OSS uppkommit och vad som gör att den särskiljer sig från proprietär programvara. Uppsatsen innefattar inte heller någon studie av företag utanför Sverige, vilket givetvis tillsammans med den kvalitativa ansatsen begränsar resultatet och möjligheten till generella slutsatser.

1.5 Terminologi

I denna uppsats används begreppet Open Source huvudsakligen på tre sätt. Nedan följer av ett förtydligande av dessa olika sätt:

Open Source som mjukvara. Open Source Software (OSS) avser mjukvara vars licens överensstämmer med Open Source Initiatives (OSI) definitioner av Open Source. Mer om licenser och OSI finns att läsa i kapitel 3.3. Synonymer till OSS som förekommer i uppsatsen är bland andra fri programvara, Open Source-mjukvara och Open Source-program.
Open Source som utvecklingsmetod. Avser det sätt att utveckla programvara som anses vara karraktäristiskt för Open Source, snarare än programvaran i sig. En mer detaljerad beskrivning av utvecklingsmetoder finns i kapitel 3.3.5 samt under kapitel 4.1.
Open Source som rörelse. En slags samlande beteckning för de som utvecklar OSS och deras samverkan och mål. Begreppet Open Source-rörelsen är inte sällan politiskt färgat av tankar kring Open Source som sätt att förändra synen på programvara. Exempel på detta finns bland annat i kapitel 4.2 men även i litteratur som DiBona, Ockman och Stone (1999) och Raymond (2001).

Jag har försökt att poängtera skillnaden i de fall där begreppets användning inte framgår av sammanhanget.

I uppsatsen, särskilt den del som behandlar historien bakom Open Source, använder jag Open Source synonymt med Free Software även om jag är medveten om att detta kan ifrågasättas (se kapitel 3.1 för en kort beskrivning av skillnaden mellan de två begreppen).

Inom ämnet Open Source tenderar förkortningar och begrepp att göra delar av uppsatsen svårläst. För att förenkla läsningen finns en ordlista med förklaringar (se appendix A).

Slutligen finns ett index över relevanta personer, program, begrepp, med mera som förekommer i uppsatsen (se appendix B).

1.6 Disposition

För att ge en överblick över uppsatsen följer här en kort disposition:

Kapitel 2 innehåller metoden som använts för att samla in materialet och beskriver bland annat kvalitativ och kvantitiv metod. Kapitlet tar även upp mitt urval av företag och det tillvägagångssätt jag använt mig av i studien.

Kapitel 3 ger läsaren en bakgrund till vad Open Source är. Genom att förklara begrepp och historiska aspekter hoppas jag kunna underlätta förståelsen av ämnet. Bland annat behandlas utvecklingsmetoder och vad som anses vara Open Source styrkor och svagheter.

Kapitel 4 tar upp ett antal teorier som jag ansett relevanta för att kunna placera Open Source som affärsidé i sin rätta kontext.

Kapitel 5 visar resultaten av studien med hjälp av det teoretiska ramverket och utgår huvudsakligen från citat från respondenterna.

Kapitel 6 innehåller en diskussion med jämförelse av resultat och teoretiskt ramverk samt slutsatser.

Kapitel 7 består av en summering av såväl uppsats som resultat.

Kapitel 2 – Metod

Metodkapitlet ska lägga grunden för systematik och ge ett ramverk för hur vi närmar oss ett problem och analyserar det (Holme & Solvang, 1997). Detta kapitel tar sitt avstamp i den syn en forskare har på sig och sin relation till omvärlden. Längre fram i detta kapitel flyttas fokus mot olika kvalitativa och kvantitativa metoder för att sedan närmare beskriva mitt val av metod. Slutligen, när kapitlet närmar sig sin ände, presenterar jag studiens urval och de undersökta företagen samt en kritik av mina källor.

2.1 Vetenskaplig ansats

Som forskare kan man välja att se på kunskap på olika sätt. Positivism, hermeneutik och kritisk teori är begrepp som används för att förtydliga vilken syn forskaren antar på kunskap.

2.1.1 Positivism, hermeneutik och kritisk teori

Den positivistiska synen på kunskap är den att kunskapen absolut och riktig. En positivistisk ansats bygger på logik och fakta som i sin tur grundar sig på mätningar. Enligt positivismen finns det endast två källor till denna absoluta kunskap: det vi registrerar med våra fem sinnen och det vi kan resonera oss fram till med logik. Positivismen är utpräglat kvantitativ (för exempel på kvantitativa forskningsmetoder se kapitel 2.2) och har sitt ursprung inom naturvetenskapen (Eriksson & Wiedersheim-Paul, 1997). Den traditionella positivistiska synen på forskning innebär att man söker ställa sig ovanför individuella preferensramar och subjektiva iakttagelser (Holme & Solvang, 1997), sålunda skall det inte heller spela roll vem som genomför undersökningen. En följd av detta är att forskningen skall vara möjlig att genomföra igen och kunna bevisas genom repetitiva undersökningar.

Mot den positivistiska synen står den hermeneutiska synen. Ordet hermeneutik kan översättas som "tolkningslära" och tar till skillnad från positivismen hänsyn till faktorer i samhället kring forskaren. Man menar att rådande etiska, moraliska och politiska värderingar kan påverka

forskningsresultatet. Som följd av detta kan man inte säga att den hermeneutiska forskningen grundar sig på eller söker den absoluta "sanningen". Snarare studerar man sociala helheter genom att försöka få reda på hur de olika aktörerna uppfattar den samma. Ofta handlar det om att försöka förstå människor och deras agerande och dialogen är en viktig grund för att uppnå kunskap (Eriksson & Wiedersheim-Paul, 1997). Enligt den hermeneutiska synen på forskning spelar forskarens värderingar och vem den samme är in och som en följd av detta anses att inte all forskning är exakt eller mätbar i kvantitativa termer.

För att sammanfatta skillnaden mellan den positivistiska och hermeneutiska ansatsen kan man säga att positivismen beskriver och förklarar utifrån fakta medan hermeneutiken söker helhetsförståelse och insikt.

En tredje syn är kritisk teori som kan sägas vara något mitt emellan den positivistiska och den hermeneutiska synen. Den kritiska teorin hävdar att man kan behålla såväl åskådarperspektivet, det positivistiska analyserandet av fakta, som deltagarperspektivet, det hermeneutiska studerandet av agerande och beteende. Man hävdar att båda perspektiven är nödvändiga för att förstå den sociala verkligheten. (Eriksson & Wiedersheim-Paul, 1997).

Ofta antar man ett hermeneutiskt eller kritiskt teoretiskt synsätt i situationer då det är svårt att tala om absolut kunskap (ibid).

I denna uppsats har jag valt ett hermeneutiskt synsätt då jag just söker svaren på hur ett antal företag, representerade av enskilda individer, agerar snarare än att bara se till kvantifierbara fakta.

2.2 Att välja metod

De två metodgrenarna inom forskning är kvalitativ och kvantitativ metod. För att medvetet kunna välja metod krävs en förståelse för de två och därför följer här en beskrivning av kvalitativ och kvantitativ metod samt viktiga kvalitativa metoder.

2.2.1 Kvalitativ respektive kvantitativ

Den kvalitativa metoden syftar till att studera företeelser med ett inifrånperspektiv, det vill säga försöka nå förståelse för hur studieobjekten själva upplever en situation, samt analysera handlingsmönster, sociala samband och strukturer. Detta skall åstadkommas genom närhet till forskningsobjektet, till exempel med intervjuer eller på-platsstudier (se kapitel 2.3). Man använder verbala formuleringar och instrumenten består av det traditionella ordet (Backman, 1998; Holme & Solvang, 1997).

Den kvantitiva metoden studerar förhållanden just genom att försöka kvantifiera det fenomen eller den företeelse som skall studeras. Genom att till exempel framställa statistik söker man samband i den data som framkommer genom studien. Förhållningssättet är, jämfört med det kvalitativa, ett utifrånperspektiv där inte samma grad av interaktion med respondenter och andra inblandade förekommer. Kvantitativa metoder utmynnar i numeriska observationer eller låter sig transformeras till sådana (Backman, 1998; Holme & Solvang, 1997).

2.2.2 Intervju och observation

Två viktiga metoder inom kvalitativ metod är intervju och observation. Här följer en beskrivning som utgår från Holme och Solvang (1997) och Repstad (1993). De två metoderna liknar varandra i flera avseenden, båda är tänkta att utföras i den kontext som skall undersökas och de samtal som förekommer i en observationsstudie liknar de i intervjusituationen.

Intervjun är en mer formell form och det finns lite utrymme för forskaren att studera företeelsen på håll. Den kvalitativa intervjun liknar mycket ett vanligt samtal, samtidigt som den har inslag av den journalistiska intervjun. Istället för att ställa ett antal frågor i följd skall intervjuaren försöka få respondenten att själv berätta om sina upplevelser eller åsikter. Intervjuaren skall bara utöva en svag styrning av samtalet och leda det in på de områden där den söker svar. Tanken bakom detta är att man skall uppnå en flexibilitet

och frihet för den intervjuade inom det avgränsade område där forskaren valt att söka förståelse.

Den kvalitativa intervjuns brist är att den ofta anses ge en subjektiv bild av det som studeras, då man låter respondenterna få stort inflytande över vad som kommer fram i intervjun. Subjektiviteten behöver inte vara något negativt, helt beroende av vilka resultat man söker i sin studie. Dock bör man absolut vara medveten om denna faktor när man analyserar resultaten av en kvalitativ intervju.

Nackdelar med såväl observation som intervju är att båda metoderna är tidskrävande. Samtidigt kan det vara svårt att veta om det man får fram i observationen eller intervjun är representativt. Såväl beteende som svar på frågor riskerar att vara tillrättalagda och sålunda ge en felaktig bild. De intervjuade eller observerade kan också påverkas av forskaren och tillrättalägga beteenden och yttranden. En fördel är att man som forskare kan få veta vad de studerade tycker, formulerat genom deras egna ord respektive handlingar.

2.4 Val av metod

Jag har valt den kvalitativa intervjun som metod då den förståelse jag söker inte går att få genom statistiska jämförelser. Jag söker snarare enskilda personers tankar och åsikter samt konkret information om det företag de representerar och varför de valt den väg för sitt företag som de valt. Utifrån detta hoppas jag sedan kunna svara på mina frågeställningar.

Alternativen till en kvalitativ metod, till exempel en (kvantitativ) enkät, kräver ett omfattande underlag för att bli intressant. I mitt fall finns förhållandevis få företag tillgängliga för studien vilket omöjliggör ett tillräckligt underlag. Till detta kommer att företagen, enligt min åsikt, inte kan betraktas som en homogen grupp vilket skulle ha varit en fördel vid en kvantitativ studie. Det enda de har gemensamt är att de sysslar med Open Source, i övrigt befinner de sig inom olika verksamhetsområden.

Den kvalitativa metoden lämpar sig alltså bättre på grund av mitt numerärt begränsade underlag. Genom valet av kvalitativ metod fås mer nyanserade svar än vad som varit möjligt genom till exempel en enkätundersökning eller någon annan form av statistisk undersökning (Repstad, 1993).

Vad gäller valet av kvalitativ metod, den kvalitativa intervjun kontra till exempel observation, är det huvudsakligen två aspekter som inverkat. Dels en rent praktisk aspekt i form av den begränsade tid jag har till förfogande samt respondenternas geografiska spridning. Det skulle helt enkelt inte vara möjligt för mig, av ekonomiska skäl, att tillbringa tillräckligt lång tid på annan ort. Den andra aspekten är att observation i första hand inriktar sig på individers och/eller gruppers agerande. Vad jag söker är personers tankar kring vissa företeelser, något som i detta fall knappast manifesteras i direkt handling utan snarare erfarenheter och information hos företagen och dess ledare, de intervjuade.

2.5 Studiens urval

Mitt slutmål för urvalsprocessen var att hitta en samling företag som huvudsakligen och uttryckligen har Open Source-produkter som grund för sin verksamhet. Som stöd för mitt urval använde jag mig av de respektive företagens hemsidor och företagspresentationerna på dessa sidor. För att dels få en bild av företaget som var representativ för hela verksamheten och dels den bild företaget ville kommunicera utåt valde jag att i möjligaste mån intervjua någon ur företagsledningen, till exempel verkställande direktör. Ett delmål i urvalsprocessen var även att försöka variera de olika respondenternas hemvist i olika marknadssegment som utbildning, mjukvaruutveckling och konsulttjänster. I praktiken kunde jag inte aktivt välja att realisera detta delmål då företagen var så pass få.

Jag gjorde urvalet i två steg:

Första urval
Första urvalet av företag hämtade jag från Open Source Forum Scandinavia (Open Source Forum, 2002), som är ett svenskt initiativ till att skapa en forum för svenska Open Source-företag och företag som är intresserade av

Open Source, främst genom anordnande av mässor och konferenser. Open Source Forum Scandinavia har sammanlagt elva medlemsföretag och dessa blev första urvalet. Open Source Forum Scandinavia är den enda svenska Open Source-sammanslutningen med medlemsföretag som jag hittat. De företag som jag fann genom Open Source Forum var Compaq AB, Cendio Systems AB, Codefactory AB, Data Construction/Caldera, GeekOnline, Hewlett Packard AB, Leissner Data AB, Memstore AB, Nohup AB, Raditex AB, SUSE Linux AG.

Renodling av urval

Vid en närmare genomgång av de elva medlemsföretagen fann jag att tre av dessa till stor del baserar sin verksamhet på att levererar hårdvara (Hewlett-Packard, Compaq och Memstore) och jag ville främst fokusera på mjukvaruorienterade företag. Ett företag valde jag bort på grund av företagets geografiska placering (SuSe Linux AG, Tyskland), två företag valde jag bort på grund av att de till stor del baserar sin verksamhet på proprietär programvara (Data Construction/Caldera, Leissner Data AB), ett företag sorterades bort då det huvudsakligen säljer t-shirtar och böcker som appellerar Open Source-kulturen (GeekOnline, enskild firma).

Det sista steget var rent praktiskt till sin natur. De företag där någon i ledningsfunktion hade möjlighet att ställa upp på intervju ingår i studien. I detta urval föll ett företag bort: Nohup AB i Stockholm. Jag försökte även få en intervju med företaget MySQL/TCX Datakonsult men detta gick tyvärr inte att passa in tidsmässigt.

Efter min urvalsprocess hade jag funnit tre företag, vilka närmare beskrivs under resultatkapitlet, kapitel 5. Ett företag som motsvarade mina kriterier, South Pole AB, tillkom genom en av respondenternas förtjänst.

Det bör nämnas att antalet företag på den svenska marknaden som huvudsakligen sysslar med Open Source, öppen källkod, är idag ytterst begränsat och även om det saknas exakta siffror uppskattar jag det till att vara maximalt runt tio företag i mjukvarubranschen och kanske ytterligare tio inom service, support och utbildning. Det är också något komplicerat att

hitta dessa företag, då de saknar någon uttalad gemensam intresse- eller branschorganisation i Sverige.

2.6 Studiens genomförande

Kontakten med företagen skedde initialt genom telefonsamtal till deras respektive kontor. Innan jag ringde hade jag via deras webbsidor, tidningsartiklar eller Patent- och registreringsverket tagit reda på antingen namn eller namn och direktnummer till den jag ville intervjua. Detta för att minska risken för att fastna hos exempelvis en sekreterare och behöva sköta kommunikationen genom en tredje part. Alla respondenter satt vid intervjutillfället i företagsledningen. Anledning till att jag valt att fokusera på personer i företagsledningen är att jag tror att dessa har större insikt i de övergripande målen för verksamheten.

Bokning av intervjutid gjordes i nästan alla fall via e-post efter det initiala samtalet. Samtidigt med denna e-post skickade jag en presentation av uppsatsen och dess syfte.

Studien genomfördes genom intervjuer av typen kvalitativ intervju som jag beskrivit tidigare i detta kapitel. Intervjuerna utfördes på företagens huvudkontor i respektive företagssäte.

Som regel tog intervjuerna mellan 60 och 90 minuter och följde den tematisering som finns att se i intervjumanualen (Appendix A). Intervjumanualen var uppdelad i tre huvuddelar: ”bakgrund om företaget”, ”vad betyder Open Source för er?” och ”Varför Open Source?”. Den första delen fokuserade huvudsakligen på den ursprungliga affärsidén och om den förändrats över tid samt rena faktafrågor som när företaget startades. Del två var ganska lös i sin struktur och huvudpunkten var att den intervjuade skulle ge sin/företagets syn på vad Open Source är och vad denna syn fått för praktiska effekter. Den tredje och sista delen fokuserade på frågan om varför företaget valt Open Source som metod, grund för verksamheten och programvara samt vad detta lett till. Det bör tilläggas att intervjumanualen

inte följdes slaviskt utan snarare fungerade för som ett stöd för att intervjun inte skulle fastna på ett ämne eller gå in på helt andra ämnen.

Samtliga intervjupersoner har gett sitt medgivande till publicering av deras samt företagets namn.

Nedan följer en kort presentation av de intervjuade och var intervjun genomfördes:

Inge Wallin, vice VD, Cendio Systems AB
Tidigare VD för företaget. En av grundarna, liksom de andra grundarna utbildad vid Linköpings tekniska högskola. Wallin sysslar idag med försäljning och kundkontakt. Intervjun genomfördes i företagets lokaler i Linköping.

Patrik Fransson, chief executive officer, Codefactory AB
CEO för företaget sedan några år. Kom tidigare från WM-Data. Intervjun genomfördes på Umestans restaurang, Umeå.

Göran Hasse, VD, Raditex AB
VD som arbetat med öppna system/Unix/BSD i ett tjugotal år. Arbetar med utbilning och försäljning. Intervjun genomfördes i företagets lokaler i Nacka.

Jakob Sandgren, VD, South Pole AB
En av grundarna för företaget. Utbildad i datateknik i Luleå och sysslat med Linux sedan 1994. Intervjun genomfördes i företagets lokaler i Solna.

Representanterna för företagen har tillfrågats angående publicering av deras namn och godkänt detta.

2.7 Källkritik

De val som jag gjort under uppsatsens gång har givetvis påverkat resultatet. I den kvalitativa intervjuform jag valt har de intervjuade, uteslutande personer i ledningsposition, fått ge sin bild av företaget i fråga och dess

verksamhet. Det är möjligt att den bild de beskrivit har varit tillrättalagd och även om så inte varit fallet är deras bild just deras, subjektiv. Man kan misstänka att denna bild inte överensstämmer med den bild exempelvis en konkurrent, eller till och med en annan anställd på samma företag, skulle givit av företaget. Samtidigt som jag varit inriktad på att få reda på vad företaget ersätter den proprietära programvaran med har jag också varit intresserad av deras strategier, visioner, mål och affärsidéer. Presentationer av sådana ting tenderar att till stor del bli marknadsföring av företaget men det ger även en bild av hur företaget vill framstå och vart ledningen vill styra skeppet.

Bilden av företagen i studien skulle kanske ha blivit annorlunda om jag intervjuat personer i företaget som sysslade med exempelvis programmering. Dock anser jag att bilden av just företagets strategier, visioner och mål kan bli mer rättvisande om en person med ansvar för detta, det vill säga någon i ledande position, får uttala sig.

Man måste också vara medveten om att delar av intervjuerna, såväl som delar av litteraturen, är partsinlagor till förmån för Open Source. De som sysslar med Open Source och ibland i viss mån har det som födkrok vill naturligtvis framställa det i en god dager. Jag bedömer dock inte detta som ett problem, då jag i denna uppsats inte är ute efter att argumentera för eller mot Open Source utan snarare söker svaret på vilka företagsmodeller som används av ett antal Open Source-företag i Sverige. Den tidigare forskningen på området är begränsad och den allmänna forskningen kring Open Source inriktar sig i mycket på de eventuella fördelarna med Open Source.

En del av materialet till uppsatsen kommer från Internet. Detta kan i sig vara en fara då det som regel är svårare att verifiera materialets äkthet och relevans. Dock hör det till ämnets natur att den mesta informationen finns att hämta på webben. För att minimera risken för tvivelaktigt material har jag uteslutande använt mig av källor vars relevans jag anser mig kunnat kontrollera. Antingen genom att källorna refererats i litteratur, eller att personerna som skrivit artiklarna varit forskare eller praktiker knutna till universitet, organisationer eller företag med god renommé.

Validiteten i studien kan även bli begränsad av det lilla urvalet av företag, trots att de utgör en stor procentuell del av antalet företag i Sverige som sysslar uteslutande med Open Source.

Kapitel 3 – Open Source - historik och karaktäristika

Detta kapitel ger en bakgrund till Open Source, dels som rörelse men också som sätt att traditionellt utveckla programvara inom rörelsen. Tanken är att orientera läsaren på området och beskriva vissa för Open Source centrala begrepp för att sedan gå vidare med relevanta teorier. En beskrivning av ämnet ur bland annat historisk och kulturell synvinkel är viktigt för att orientera läsaren inom ett ämne vars bakgrund inte alltid är allmänt känd.

3.1 Open source

Termen Open Source myntades så sent som i februari 1998, av ett antal personer aktiva inom rörelsen kring fri mjukvara. En av huvudanledningarna till att man ville skapa denna nya term var att komma bort från den tidigare definitionen Free Software (History of the OSI, 2002) som ofta förväxlades med till exempel freeware och shareware (gratis- och spridprogram). Open Source (såväl -rörelsen som –mjukvaran) går dock mycket längre tillbaka än 1998 och sägs ha uppkommit på de amerikanska universiteten under 1960-talet. Open Source Software (OSS) har blivit ett allmänt känt begrepp under de senaste åren och haft ett stort genomslag inom IT-sektorn, såväl som i styrelserummen som i akademien (Raymond, 2001; Young & Goldman Rohm, 1999).

En annan förklaring till uppkomsten av termen Open Source och användning därav istället för Free Software är att man strävade efter en kommersialisering av den ”fria mjukvaran”. Genom att använda termen Open Source skulle man kunna framhäva de rent praktiska fördelarna med ”fri mjukvara”, snarare än etiska och moraliska aspekter. Följden av detta skulle enligt förhoppningarna göra marknadsföringen av det nya begreppet Open Source gentemot företag lättare (Moglen, 2000).

I vissa delar kan man fortfarande se en motsättning mellan de organisationer och personer som representerar Free Software (Free Software Foundation med bland annat R. Stallman som förgrundsgestalt) respektive Open Source (Open Source Initiativ och bland annat E. Raymond). Free Software

Foundation har traditionellt en mer politisk radikal framtoning än Open Source Initiative (OSI), som å sin sida uttryckligen vill marknadsföra Open Source tillsammans med och gentemot företag.

3.1.1 Olika sidor av myntet

Man bör vara medveten om att bilden av Open Source, ofta representerat av operativsystemet Linux, är mångfasetterad.

Open Source som utvecklingsmodell sägs av vissa lösa flera av de problem som mjukvaruutvecklingen dras med idag. De lösningar som Open Source-rörelsen producerar i en snabb takt består av pålitliga och högkvalitativa mjukvaror till en låg kostnad (Feller & Fitzgerald, 2000).

Lyssnar man på andra källor är inte bilden av Open Source lika ljus. I juni 2001 kommenterade Steve Balmers, Microsofts VD, Linux och således Open Source-utveckling med följande ord: ”Linux is a cancer that attaches itself in an intellectual property sense to everything it touches.” (Newbart, 2001). Den välbekante grundaren av Microsoft, Bill Gates, har även kommenterat Open Source med att det skulle vara en ideologisk skapelse som strider mot kapitalismens regler (Gates, 2002).

Microsoft har för Open Source-rörelsen kommit att symbolisera den proprietära mjukvaruindustrin. Företaget är enligt ”hacker-jargong” (se Appendix B) synonymt med ”the Evil Empire” (Raymond, 2002). Det finns också olika grader av anti-kommersialism inom Open Source-rörelsen som inte går att bortse från. Vissa personer engagerade i rörelsen ser Open Source-utveckling som en motståndsrörelse mot mjukvaruindustrins jättar (Ljungberg, 2000). Även om denna tendens kanske märks starkast inom Free Software Foundation (se till exempel Moglen, 2000).

Det kommersiella intresset för Open Source har dock inte uteblivit. Ett exempel på detta är IBM som 2001 spenderade ungefär en miljard dollar på utveckling av Open Source-produkter och hade under det året omkring 1500 utvecklare sysselsatta med OSS-utveckling (Feller & Fitzgerald, 2002). Sun Microsystems, SAP, Nokia och Oracle är andra stora företag som även de, på

olika sätt (genom allt från offentliga uttalanden till praktisk handling), närmat sig området OSS (Young & Goldman Rohm, 1999; OSI, 2000b).

3.2 Historisk återblick

Open Source i sig är inget nytt, även om termen myntades så sent som i februari 1998 (OSI, 2002d). I den tidiga programvaruutvecklingens tidsålder, på 60- och 70-talet, skapades de flesta datorprogram på universitet och högskolor. Programmen i sig själva hade litet eller inget ekonomiskt värde, de var snarare redskap för och resultat av forskning utvecklat av forskarna själva. En naturlig följd av detta var att man visade upp sina resultat, inklusive programkod, för andra forskare och intresserade. Genom att dela med sig av sina program kunde utvecklarna, i enlighet med den akademiska traditionen, läsa andras programkod, kritisera, kommentera och utveckla den. Faktum var att flera amerikanska institutioner, bland annat National Science Foundation och Defense Advanced Research Projects Agency, krävde att program som utvecklades med hjälp av deras medel skulle göras fritt tillgängliga med komplett källkod (Wayner, 2000). Till detta kommer dessutom att marknaden för datorer var ytterst begränsad och hårdvarutillverkarna skickade med programkod för att man skulle kunna använda deras maskiner (Raymond, 2001).

3.2.1 Hackarnas uppkomst

Att dela med sig av sin programkod var sålunda den vedertagna principen och kring denna princip och de som var delaktiga i utvecklingen uppstod en subkultur: hackerkulturen. Hackerkulturen bestod av skickliga programmerare och hade inget med den bild av ”hackers” som idag ofta förmedlas, via till exempel medierna: personer som bryter sig in i andras datorsystem. Hackers var ofta anställda vid universitet eller stora datorföretag och arbetade med programmering och systemutveckling. En beskrivning av hackers som hackare själva vill förmedla beskrivs med Eric Raymonds begrepp ”Riktiga Programmerare”, något som kanske kan ge en vägledning till vad det innebar och till viss del fortfarande innebär att tituleras ”hacker” (DiBona, Ockman & Stone, 1999).

3.2.2 Från öppen till sluten källkod och tillbaka

Under början av 80-talet, när Pc:n gjorde sitt intåg på marknaden, förändrades synen på mjukvara radikalt. Företag som tidigare distribuerat egenutvecklad mjukvara fritt började ta betalt för den samma. Vanligt blev också att kunden inte längre fick med källkoden till programmen, vilket bland annat innebar att kunden varken kunde modifiera dem eller se hur de var uppbyggda. En förklaring till detta är att kunderna inte längre i första hand var forskare och utvecklare utan snarare hem- och företagsanvändare. Företagen behövde inte längre ta hänsyn till den lilla grupp som ville kunna ändra och modifiera programmen, utan inriktade sig på de som nöjde sig med fungerande program. Med detta som grund gick mjukvaruföretagen till största del över till sluten källkod som inkomstbringare. Tidigare hade mjukvaran snarare varit en bonus som följde med när man köpte hårdvara (Raymond, 2001; Wayner, 2000).

I många år levde Open Source-rörelsen ett ganska undanskymt liv, främst begränsat till olika universitet, och inte förrän under mitten av 80- och 90-talet började det, utåt sett, hända saker. I januari 1984 startade Richard M. Stallman GNU-projektet med målsättningen att skapa ett fritt operativsystem (The Gnu Project, 2002). Stallman hade tidigare arbetat på Massachusetts Institute of Technology, MIT, men lämnade det samma för att skapa ett eget fritt operativsystem. Den kanske viktigaste anledningen att Stallman bestämde sig för att starta GNU-projektet var att amerikanska AT&T bestämt sig för att begränsa tillgången till källkoden för sitt Unix-operativsystem. AT&Ts Unix var ett av de mest använda operativsystemen vid MIT i början av 80-talet men även många andra mjukvarutillverkare gick över till sluten källkod. Stallman såg förändringen som ett hot mot den gamla traditionen att dela med sig av källkod och ville skapa en motkraft, till stor del av moraliska skäl då han ansåg att det handlade om frihet för användaren (DiBona, Ockman & Stone, 1999; Wayner, 2000).

Stallman skrev vid denna tid också The GNU Manifesto (Stallman, 2002). GNU Manifesto förklarar Stallmans syn på öppen källkod och vikten av fri programvara (ibid). Med GNU Manifesto som grund har senare Gnu Public Licens (GPL) växt fram, en programvarulicens som idag används för en stor mängd program (se kapitlet ”Licens eller virus”). Under samma period växte också andra öppna mjukvaruprojekt som till exempel Berkeley Software

Distribution (BSD). Även BSD var en reaktion på att AT&T kommersialiserat sitt Unix-operativsystem (Young & Goldman Rohm, 1999).

Under 80-talet kan vi också se de första exemplen på inkomster relaterade till Open Source-produkter. 1985 började Richard Stallman sälja magnetband med sitt program GNU Emacs för 150 dollar. Stallman försörjde sig också på att erbjuda konsulttjänster kring sin fria programvara. 1987 startade Cygnus Solutions i USA med affärsidén att erbjuda konsulttjänster kring den fria produkten GCC, GNU C Compiler. Cygnus ansåg att GCC var avsevärt mycket bättre än konkurrerande programvara på marknaden och kom fram till att pengarna inte fanns att tjäna på programvaran i sig, utan på kringtjänster som support och utbildning (Moody, 2001).

3.2.3 Pingvinernas intåg

Riktig stor uppmärksamhet och kommersiell användning skulle dock Open Source få först på 90-talet då den unge finske datastuderande Linus Torvalds började sitt arbete med Linux. Den sjuttonde september 1991 såg Torvalds operativsystem dagens ljus för första gången (Moody, 2001; Torvalds & Diamond, 2001). Sedan dess har Open Source, mycket genom Linux, haft stora framgångar och nådde under år 2000 en 27-procentig marknadsandel på servermarknaden (Shankland, 2001). Linux och dess skapare har på många sätt fått symbolisera motpolen till Microsoft. Symbolen för Linux, den lilla pingvinen Tux, har en ödmjuk framtoning som framstått som milsvitt skiljd från Microsoft i skuggan av företagets anti-trust rättegångar och försenade operativsystem. Idag är Linux troligen det största samarbetsprojektet mellan programmerare som världen sett. Och tankarna om att källkoden skall vara tillgänglig för alla utgör fortfarande en grundsten för utvecklingen av Linux (Torvalds, 2001).

Trots att Linux är den mest kända skapelsen vad gäller Open Source-mjukvara är det långt ifrån den mest använda. Open Source-programvara utgör idag till stor del stommen i Internet genom till exempel webbservern Apache som används på över 60 procent av de aktiva webbplatserna på Internet (Netcraft Web Server Survey, 2002). Andra exempel på spridd Open Source-mjukvara är namnservern BIND, e-postservern Sendmail samt programmerings-och redigeringsverktyget Emacs (Raymond, 2001).

3.3 Karakteristika

I detta kapitel behandlar jag dels vad Open Source innebär ur licenssynpunkt och dels vad som utmärker det arbetssätt som OSS-utveckling ofta uppvisar. Syftet med kapitlet är att ge en förståelse för hur de olika Open Source-licenserna verkar och vad de innebär samt att ge en bild av det i vissa avseenden nya arbetssättet som ofta används vid utveckling av OSS.

3.3.1 38 licenser att tämja dem

The Open Source Initiative (OSI), en organisation som bland annat arbetar med licensfrågor kring Open Source, har klart definierat termen Open Source i något som kallas The Open Source Definition (OSD). Utifrån OSD har man godkänt ett antal, i dagsläget 38 stycken, licenser för mjukvara. Dessa licenser specificerar förhållandet mellan den eller de som skapat mjukvaran och användarna. Bland annat bestämmer licensen hur mjukvaran får spridas, kopieras och ändras. Ett antal grundläggande kriterier måste dock uppfyllas för att licensen skall bli godkänd av OSI och anses överensstämma med OSIs definition av öppen källkod (OSI, 2002).

Innan OSI lanserade sin version av OSD var troligen Richard Stallmans och FSF:s GPL-licens den vanligaste licensen. Bakom tankarna kring GPL-licensen fanns ett koncept kallat Copyleft, ”All rights reversed”, som innebar att man fokuserade snarare på användarens rättigheter snarare än upphovsmannens. Initiativtagarna till OSD såg dock Copyleft-licensen som en alltför politisk licens och var oroliga för att detta skulle begränsa OSS möjligheter att penetrera den kommersiella mjukvarumarknaden. OSD var ett sätt för OSI att avpolitisera Open Source, även om den av OSI godkända Gnu Public Licens, GPL (se kapitlet ”Licens eller virus”), i princip är det samma som Copyleft (DiBona, Ockman & Stone, 1999).

Nedan följer ett sammandrag av de viktigaste punkterna, baserat på Open Source Definition (version 1.9, 2002; Feller & Fitzgerald, 2002; DiBona, Ockman & Stone, 1999). Det bör dock noteras att definitionen (OSD) i sig inte är någon licens utan snarare riktlinjer för hur licenser av Open Source-typ

skall se ut (ibid), sålunda ger den en bra överblick över vad som kan anses vara Open Source-mjukvara och vad som inte är det.

Mjukvaran skall få distribueras fritt. Med detta menas att mjukvaran är fri att använda som del av en större mjukvarudistribution.
I praktiken: vem som helst får göra hur många kopior av mjukvaran som helst, ge bort eller sälja den samma. Allt detta utan kostnad.
Källkoden till mjukvaran måste finnas tillgänglig. I de fall då källkoden inte medföljer programmet måste det tydligt anges var ifrån källkoden kan fås.
I praktiken: tanken är att källkoden, som är nödvändig för att kunna göra förändringar och nya versioner av mjukvaran, alltid skall följa med programmet eller program som skapats utifrån originalet.
Förändringar av källkoden måste vara tillåten. Även mjukvara baserad på originalet måste tillåtas.
I praktiken: utan rätten att göra förändringar faller hela tanken med Open Source, därför måste det alltid vara tillåtet att göra förändringar i källkoden.
Alla ska få använda mjukvaran. Användandet får inte begränsas till en viss grupp eller en viss person.
I praktiken: det är inte tillåtet att begränsa tillexempel kommersiell användning av programvaran.
Licensen måste gälla i alla sammanhang. Den får inte begränsa i vilket sammanhang licensen är giltig och den måste även gälla om programvaran levereras separat, utan andra produkter som den först kan ha levererats med.
I praktiken: licensen får inte kräva att programmet endast levereras tillsammans med andra Open Source-produkter.

3.3.2 Användarens frihet

OSD inbegriper i första hand rättigheter för användaren av mjukvaran, inte för den som skapat den. De olika delarna av OSD garanterar användaren friheten att ändra programvaran och sprida den vidare. OSS är inte nödvändigtvis fri i meningen gratis/kostnadsfri. Samtidigt skall den finnas fritt tillgänglig, till exempel på Internet, vilket innebär att priset för att köpa programvaran från något företag aldrig kan bli särskilt högt (DiBona, Ockman & Stone, 1999). Bruce Perens, som är en av författarna till den första Open Source-definitionen förklarar de ekonomiska aspekterna på följande sätt:

[...] the economics of information are fundamentally different from those of other products. There is very little cost associated with copying a piece of information like a computer program. The electricity involved costs less then a penny, and the use of equipment not much more. In comparison, you can't copy a loaf of bread without a pound of flour.
Bruce Perens i DiBona, Ockman & Stone, 1999, sid. 172

Open Source-mjukvara misstags ofta för att vara ”shareware” eller ”freeware” men som kan ses i taxonomin nedan finns det signifikanta skillnader i dessa licensmodeller (Cohen & Valloppillil, 1998).

	License Feature:						
Software type:	Zero Price Avenue	Redistributable	Unlimited Usage	Source Code Available	Source Code Modifiable	Public "Check-ins" to core codebase	All derivatives must be free
Commercial	-	-	-	-	-	-	-
Trial Software	X (Non-full featured)	X	-	-	-	-	-
Non-commercial use	X (Usage dependent)	X	-	-	-	-	-
Shareware	X (Unenforced licensing)	X	-	-	-	-	-
Royalty-free (freeware) binaries	X	X	X	-	-	-	-
Royalty-free libraries	X	X	X	X	-	-	-
Open Source (BSD-style)	X	X	X	X	X	-	-
Open Source (Apache-style)	X	X	X	X	X	X	-
Open Source (Linux/GNU style)	X	X	X	X	X	X	X

Tabell 1: ”Software Licensing Taxonomy” ur Cohen & Valloppillil (1998):

I tabellen ovan kan man utläsa att den största skillnaden mellan Open Source och de övriga licensformerna främst är tillgången till källkoden. Även rätten

att modifiera programmen och använda dem fritt utgör en markant skillnad. I tabellen kan man också se att de olika Open Source-licenserna skiljer sig åt inbördes på de tre sista kolumnerna i tabellen. Medan BSD- och Apache-licensen inte kräver att modifieringar av programmen skall släppas under Open Source-licens kräver GPL (ovan kallad Linux/GNU style) detta

3.3.3 Licens eller virus

Den i särklass vanligaste licenstypen som överensstämmer med OSIs riktlinjer för Open Source är GNU Public Licens (GPL). Av totalt nästan 40 000 projekt på Open Source-projektplatsen sourceforge.net är närmare 18 000 klassificerade som GPL (SourceForge, 2002). När man stöter på Open Source-programvara är den sålunda oftast distribuerad under GPL. Till skillnad från en del andra OSD-kompatibla licenser som Apache och BSD-licensen tillåter inte GPL att mjukvaran byter licens. Nya program som skapas med hjälp av källkod från ett GPL-program måste spridas vidare under GPL (DiBona, Ockman & Stone, 1999). Denna egenskap hos GPL är inget krav för att överensstämma med OSIs riktlinjer, som tillåter att programvara byter licens (Learner & Tirole, 2000)

GPL har varit föremål för mångas ifrågasättande och det var just denna licens som Steve Ballmer, VD för Microsoft, åsyftade när han uttalade att Linux är en cancer som sätter sig fast vid allt det vidrör. Linux är en av de programvaror som sprids under GPL och får därför inte modifieras för att spridas under en annan, till exempel rent kommersiell, licens.

GPL har också fått kritik för att den ska vara för politisk med många formuleringar om frihet för användaren. Licensen anses också skrämma iväg i synnerhet kommersiella användare (DiBona, Ockman & Stone, 1999; OSI, 2002c). Trots detta rekommenderar flera ledande personer i framförallt Free Software Foundation att man använder denna licens med skälet att programvaran, genom GPL, förblir fri (se till exempel Raymond, 2001).

3.3.4 Licensernas vara eller icke vara

Open Source-licenserna i stort sett förenliga med svensk lagstiftning och sträcker sig snarare längre med avseende på vilka rättigheter de ger

användaren. I vissa fall av distribution via till exempel återförsäljare kan det vara tveksamt om licensen skulle gälla. I dessa fall skulle då, om användaren väljer att inte följa den aktuella Open Source-licensen, svensk upphovsrätt gälla. Svensk upphovsrätt är avsevärt mer restriktiv vad gäller rätten att förändra mjukvaran, så det är inte troligt att någon konflikt skulle uppstå på grund av detta mellan företag, som väljer att distribuera mjukvara under Open Source-licens (Andersson, 2000; Pawlo, 2002).

Det bör dock noteras att till exempel GPL kan orsaka problem för företag som utvecklar programvara. Om ett sådant företag skulle utveckla nya program baserade på program spridda under GPL måste dessa nya program spridas under GPL. Likaså kan det uppstå problem om man skulle kombinera delar av Open Source-programvara som har icke-kompatibla licenser till en ny programvara (Pawlo, 2002).

3.3.5 Löst organiserat – snabbt producerat

Bortom licenserna som sätter den formella etiketten på vad som är OSS eller inte finns olika metoder för hur Open Source-rörelsen ofta organiserar sitt arbete. Denna del av Open Source-rörelsen är troligen en av de mest studerade och analyserade. En av anledningarna till den grundliga analysen är de problem som den traditionella mjukvaruindustrin dragits med i flera år: utvecklingen av nya program tar för lång tid, kostar för mycket och ger inte de kvalitativa resultat som är önskvärda. Open Source-rörelsens utvecklingsmetoder har setts som ett potentiellt sätt att lösa dessa problem (Feller & Fitzgerald, 2002), se även ”Linus lag möter Brooks lag”. Utvecklingsmetoden som oftast används inom Open Source-projekt beskrivs på följande sätt av Eric Raymond (2001), med Linux som konkret exempel:

”Linus Torvalds [skaparen av Linux, förf. anm.] utvecklingsmetod – att publicera snabbt och ofta, att delegera så många uppgifter som möjligt, att vara öppen till promiskuitetens rand – kom som en överraskning. Här var det inte tal om lugnt och pietetsfullt katedralbyggande – Linuxgänget liknade snarare en stor, sorlande basar av människor med olika mål och arbetsmetoder (Linuxarkiven är en passande symbol, för de tar emot bidrag från vem som helst).”
Raymond (2001) sid. 26

Om man skall sammanfatta karakteristika för Open Source projekt skulle denna kunna kondenseras till följande fem punkter:

Utvecklarbasens sammansättning:
Många löst sammankopplade utvecklare. De inblandade i ett utvecklingsprojekt återfinns inte sällan i andra Open Source-projekt. Projekten leds oftast av en eller flera ansvariga, som snarare ansvarar för snarare koordinering än programmering. Det finns ofta en stor geografisk spridning bland utvecklarna och många projekt involverar människor från såväl USA och Europa som Asien (Feller & Fitzgerald, 2002; Josh & Valloppillil, 1998; Raymond, 2001).
I nästan alla projekt sker kommunikationen mellan projektmedlemmarna över Internet, via e-postlistor och diskussionsforum. Den snabba kommunikationen är på många sätt en förutsättning för att det decentraliserade arbetet skall kunna överblickas (Feller & Fitzgerald, 2002; Moody, 2001).

Parallell utveckling:
Många Open Source-projekt karakteriseras av parallell utveckling vilket innebär att flera utvecklare jobbar med delprojekt som har liknande mål varpå den bästa lösningen inlemmas i projektet (Feller & Fitzgerald, 2000, Raymond, 2001). I den parallella utvecklingen ligger också att projekten är uppdelade i olika delar, som utvecklas var för sig och sedan sammanförs till en fungerande helhet. Detta medger att flera olika utvecklare enskilt kan arbeta med projektet utan att beröra varandras delprojekt (Jones, 2000; Raymond 2001).

Felsökning/buggtestning:
Oftast opererar Open Source-projekten efter vad som kallats en del av Linus lag (efter Linus Torvald, upphovsmannen till Linux): ”Given enough eyeballs, all bugs are shallow” (Feller & Fitzgerald, 2002; Himanen, 2001; Raymond, 2001). Vilket fritt översatt får innebörden att om tillräckligt många ögon söker efter buggar, fel i mjukvaran, så hittar man de flesta. Detta anses delvis förklara den goda kvalitet OSS ofta uppvisar (ibid.).

Projekten har ofta en stor skara buggtestare som, ibland utan att själva

programmera eller systemera, hjälper till att utveckla projektet genom att rapportera fel och ibland ge förslag på lösningar. Många buggtestare är ofta mycket kompetenta och kan ge precis bugginformation och inte sällan direkta förslag på lösningar (ibid). Detta anknyter till nästa punkt.

Användarna är utvecklare:
Traditionellt sett har användarna av OSS själva varit utvecklare, vilket delvis är förklaringen till det fokus OSS haft på programmeringsverktyg och liknande, snarare än ”kontorsmjukvara” som ordbehandlingsprogram och dylikt (Feller & Fitzgerald, 2002; Moody, 2001). Detta är möjligtvis också en anledning till den kultur av gratissupport som man finner inom Open Source-rörelsen, där utvecklare hjälper användare med problem men även tvärtom (Lakhani & von Hippel, 2000).

Projekten släpper ofta nya versioner av programvaran:
Detta anses underlätta buggtestning och vidare utveckling. Denna metod ger också de som bidrar till projekten snabb återkoppling på att deras arbete kommer till nytta. Den snabba utvecklingen grundar sig i inkrementell utveckling, det vill säga att man bygger vidare på tidigare versioner av programvaran, vilket underlättas avsevärt av att källkoden finns tillgänglig (Feller & Fitzgerald, 2002; Jones, 2000; Jørgensen, 2001; Raymond, 2001).

Till ovanstående kommer dessutom ett viktigt inslag i organisationen kring Open Source: lejonparten av de inblandade utvecklarna arbetar ideellt. Detta ideella engagemang grundar sig, enligt flera bedömare, i en allenarådande altruism inom rörelsen, men andra minst lika viktiga faktorer är möjligheten att utveckla sitt eget kunnande och vinna erkännande för sina kunskaper (Feller & Fitzgerald, 2000, Jørgensen, 2001; Learner & Tirole, 2001, Raymond 2001). Det sistanämnda faktorn är något som Raymond (2001) beskriver som ”egoboo”, boosting ones ego, fritt översatt att få en egotripp eller stärka jaget.

Kapitel 4 – Teori och tidigare forskning

I tidigare forskning kring Open Source finns förhållandevis lite litteratur som fokuserar på affärsmodeller och -idéer. Den litteratur som behandlar Open Source delas av Feller och Fitzgerald (2002) upp i tre kategorier: journalistisk (till exempel Moody, 1997; Wayner, 2000), ambassadörsmässig, det vill säga propagerande (till exempel Himanen, 2001; Raymond, 2002; Young & Rohm, 1999) och slutligen ideologisk (till exempel Stallman, 1993). Vissa av dessa författare, debattörer och inte sällan aktiva inom Open Source-rörelsen har blivit något av symboler och språkrör för rörelsen. Detta gäller i synnerhet Eric Raymond, som ofta kallas rörelsen kulturhistoriker och antropolog (Raymond, 1999), Linus Torvalds som genom skapandet av Linux på många sätt blivit symbolen för hackers (Himanen, 2001) och Richard Stallman som en av de ledande Open Source-ideologerna.

Denna litteratur erbjuder i många fall en inblick i Open Source-rörelsen och deras vikt som bakgrundsmaterial bör kanske inte underskattas. Anekdotiska och narrativa inslag i litteraturen gör att den dock inte kan betraktas som forskning på området, utan snarare en del av Open Source-rörelsens egna historieskrivning.

Flera olika försök har dock gjorts för att presentera en teori för att förstå Open Source-mjukvaruutveckling. I dessa teorier finns ett antal fokus, bland annat ekonomisk forskning. Denna forskning har huvudsakligen varit fokuserad på frågan kring varför personer väljer att bidra med sin tid till Open Source-projekt utan att erhålla någon direkt ekonomisk ersättning.

På frågan kring grunden för denna icke-ekonomiska motivation finns huvudsakligen två förklaringar. Den ena förklaringen är en slags inter-individuell solidaritet (Dalle & Jullien, 2002), eller altruism (Raymond, 2002) inom Open Source-rörelsen som presenteras bland annat av Lakhani och Von Hippel (2000) samt Harhoff, Henkel och Von Hippel (2000). Den andra förklaringen är en mer "vanlig ekonomisk" förklaring som grundar sig i att de inblandade gör bidrag av karriäranledningar, för att stärka sin position i konkurrensen om bra arbeten. Denna synvinkel representeras bland annat av Learner och Tirole (2000) som också studerat hur ledarskap ser ut inom Open

Source-projekt. Learner och Tirole (2000) presenterar bilden av Open Source-projektledaren som den som står för visionen om hur mjukvaran skall se ut.

Open Source-rörelsen har också förklarats som en gåvoekonomi av bland andra Bergquist och Ljungberg (2001), Ljungberg (2000) och Raymond (2002). Bergquist och Ljungberg har i artikeln "The Power of Gifts" genom empiriska studier visat hur systemet med gåvor resulterar i en kultur där gåvorna, företrädelsevis arbetsinsatser i form av till exempel programmering, till rörelsen ger prestige och makt till de som bidrar. Likaså skapar dessa gåvor en viss hierarki inom rörelsen (Bergquist & Ljungberg, 2001).

Ett annat fokus är det på själva rörelsen och de faktorer som styr dess utveckling och interna samspel. En inte ovanlig syn som finns att finna i bland annat von Hippel (2001) och Edwards (2001) är Open Source-rörelsen som en community. Communites/gemenskaper formulerar enligt författarna gemensamma referensramar, mål och visioner för Open Source-projekten. von Hippel benämner Open Source-rörelsen som en "user innovation community" (Hippel, 2001, sid 4) och drar paralleller till andra gemenskaper som också uppvisar liknande beteende. Han menar vidare att dessa gemenskaper kan frodas när (1) åtminstone några användare har tillräklig motivation för att uppfinna, (2) åtminstone några användare har motiv och möjlighet att bidra med sina uppfinningar till gemenskapen, och (3) spridningen av användarnas uppfinningar kan konkurrera med kommersiell produktion och distribution (ibid).

Edwards (2001) presenterar synen på Open Source-projekt som epistemologiska communities, det vill säga gemenskaper baserade på kunskap. Han behandlar också vad han kallar för socialisering av nya medlemmar i gemenskapen och hur värderingar befästs inom gruppen av utvecklare.

I detta teorikapitel kommer jag dels att närmare gå in på Bergquist och Ljungbergs gåvoekonomi, då detta troligen anknyter starkt till Open Source-företagens spelregler på marknaden. Jag kommer också att presentera två teorier om mjukvaruutveckling som har stark koppling till synen på OSS.

Slutligen behandlar jag generell teori kring affärsmodeller och även specifikt sådana modeller som är applicerbara på OSS samt styrkor och svagheter hos OSS.

4.1 Teorier kring mjukvaruutvecklingen

I detta kapitel presenterar jag två teorier för mjukvaruutveckling som i hög grad påverkat synen på Open Source som utvecklingsmodell: den klassiska Brooks lag och den nyare Linus lag.

4.1.1 Brooks lag

1975 introducerade Frederik Brooks något som han kallade Brooks lag som lyder ”Adding manpower to a late software project makes it later” (Brooks, 1995, sid. 25), fritt översatt: att addera mer mankraft till ett försenat mjukvaruprojekt försenar det än mer. I praktiken innebär detta att man inte skyndar på ett utvecklingsprojekt genom att engagera fler människor i det.

Utvecklare som redan är insatta i projektet måste reducera sin arbetsinsats för att lära upp de nya utvecklarna. Allt eftersom projektet växer måste också allt mer resurser läggas på kommunikation. Brooks menar att dessa kommunikationsförluster är proportionella mot n(n-1), där n är antalet nya utvecklare i projektet. Detta skulle ge avsevärda tidsförluster i större projekt eftersom komplexitet och kommunikationskostnader ökar med kvadraten av antalet utvecklare. Till detta kommer att projektet blir också svårare att styra, koordinera och överblicka (Brooks, 1995).

Brooks bok ”The Mythical Man-Month” är en av vår tids mest tongivande böcker kring mjukvaruutvecklingsteorier. Brooks förutspådde att ingen ny utvecklingsmetod, ”No Silver Bullet”, skulle lösa de problem som Brooks lag implicerar (Brooks, 1995). En sammanfattning av dessa problem resulterar i tre huvudproblem som drabbar mjukvaruutveckling: projekten tar för lång tid, kostar för mycket och producerar inte tillräckligt kvalitativa program (Feller & Fitzgerald, 2002).

En vidareutveckling av Brooks lag gjord av Abdel-Hamid och Madnik i ”Software Project Dynamics: An Integrated Approach”, utvecklar Brooks lag till att lyda ”Adding more people to a late project always makes it more costly, but it does not *always* couse it to be completed later.” De menar att desto senare man tillför resurser till ett projekt desto mer riskerar de tillförda personalresurserna att försena projektet (Brooks, 1995).

4.1.2 Linus lag

Linus lag, myntad av Raymond (2001), lyder ”Har man bara tillräckligt många ögon är alla buggar små” (Raymond, 2001, sid 33). Linus lag innebär att man genom öppenhet gör det möjligt för många utvecklare och/eller användare att hitta buggar och även ge förslag på lösningar. På detta sätt menar Raymond att avlusningsprocessen inte blir lika krävande som i slutna, propeitära, mjukvaruprojekt (Raymond, 2001).

Raymond anser att ”Linus lag förkroppsligar […] den stora skillnaden mellan katedral- och basarmetoderna. Ur katedralbyggarnas synvinkel är buggar och programmeringsproblem knepiga, försåtliga, svårfattliga fenomen.” (ibid, sid 33). Med katedralen avser Raymond den traditionella mjukvaruindustrin och med basaren Open Source.

Linus lag anses delvis falsifiera Brooks lag då det visat sig att den ofta inte är applicerbar på Open Source-projekt. Det finns flera anledningar att finna till att Brooks antagande inte stämmer för Open Source-utveckling. Feller och Fitzgerald (2002) anger den snabba Internetbaserade kommunikationen mellan olika utvecklare med stöd av att användarna ofta ses som utvecklare och att bidrag till projekten tas emot från vem som helst. Jones (2000) menar arr undantaget från Brooks lag beror på att Open Source-projekten ofta saknar de fasta tidsramar som återfinns i den proprietära mjukvaruutvecklingen och sålunda utvecklar programkod efter en vision om hur programvaran skall se ut, snarare än efter ett tidsschema.

4.2 Open Source som en gåvoekonomi

En teori som presenterats av bland annat Raymond (2001) är synen på Open Source som en gåvoekonomi. Jag anser att teorin kring Open Source som gåvoekonomi är intressent att sätta i relation till den studie av affärsidéer jag kommer att presentera. I någon mån är det kanske möjligt att se en konflikt mellan tanken på en gåvoekonomi och kommersiell verksamhet. I presentationen av teorin utgår jag från forskning utförd av Bergquist och Ljungberg (2001). Trots att synen på Open Source som gåvoekonomi inte är ovanlig är Bergquist och Ljungbergs forskning den enda jag funnit som grundar sig i empiri.

Bergquist och Ljungberg menar att ett karakteristiskt drag för gåvoekonomin är att det råder ett överflöd av de varor som ges bort. Detta menar de är det drag som huvudsakligen skiljer gåvoekonomin från bytesekonomin som bygger på en brist på varorna i fråga. Bergquist och Ljungberg hänvisar också till Raymonds (2001) beskrivning av den mekanism som styr den sociala statusen i en gåvoekonomi: människors status styrs av vad de äger i form av idéer, kunskap och kreativitet och vad de delar med sig av. Först genom att dela med sig av det de skapat eller äger kan de bedömas utifrån detta tillskott till ekonomin. En lysande programmerare kan till exempel inte vinna ryktbarhet förrän han eller hon låtit andra ta del av det som programmerats.

I gåvoekonomin är sällan monetär kompensation den centrala mekanismen som motiverar inblandades vilja att ge. Bergquist och Ljungberg (2001) för fram, och problematiserar här Marcel Mauss beskrivning av gåvoekonomin, att gåvorna väcker förväntan att få någonting tillbaka. Gåvorna skapar sålunda ett socialt förhållande som ligger till grund för de sociala strukturerna i gåvokulturen.

Synen på Open Source som en gåvoekonomi verkar dock inte vara helt okontroversiell. Richard Barbrook beskriver i sin artikel ”The High-Tech Gift Economy” gåvoekonomin som en anarko-kommunistiskt fenomen (vilket i sig förvisso borde vara en motsägelse, författarens anm).

”they usually prefer to circulate gifts amongst each other. Net users will always obtain much more than will ever be contributed in return. By giving away

something which is well-made, they will gain recognition from those who download their work. For most people, the gift economy is simply the best method of collaborating together in cyberspace. Within the mixed economy of the Net, anarcho-communism has become an everyday reality."
Barbrook, 1998

Oavsett vad man anser om hans politiska syn på gåvoekonomin hävdar han, i likhet med Bergquist och Ljungberg (2001), att de som bidrar till gåvoekonomin får en form av ersättning i form av berömmelse och respekt inom gemenskapen i gåvokulturen.

Bergquist och Ljungberg (ibid) menar också att gåvorna har effekten att mottagaren blir underställd givaren och i förlängningen förväntas mottagaren bidra tillbaka till gemenskapen.

För att återkoppla till de studerade företagens förhållande till Open Source vill jag avsluta med att nämna den syn Bergquist och Ljungberg (ibid) presenterar. De menar att Open Source-företagen bidrar till Open Source-rörelsen genom att hjälpa till att skapa den "kritiska massa" av användare som är nödvändig för att OSS ska få uppmärksamhet som ett attraktivt alternativ till proprietär mjukvara. Även Osterhout (1999) framför denna åsikt och menar att företag kan bidra till att stärka OSS genom att erbjuda extra tjänster kring och programvara anpassad till OSS.

4.3 OSS – styrkor och svagheter

För att underlätta en analys av OSS ur ett kommersiellt perspektiv beskriver jag nedan de styrkor och svagheter OSS anses ha. Utifrån dessa styrkor och svagheter finns även eventuellt möjligheten att dra paralleller till var kommersiella intressen har störst möjlighet att lyckas. Läsning av kapitel 3.3.5 rekommenderas också, då det delvis sammanfaller med detta kapitel. Det bör poängteras att nedanstående är ett urval av de för uppsatsen mest relevanta punkterna och för vidare fördjupning rekommenderas de referenser som finns i kapitlet.

OSS sägs ha en god potentiell tillväxt eftersom OSS huvudsakligen utvecklas med Internet som bas av programmerare runt om i världen. I och med att

tillgången till Internet ökar är skulle det då vara rimligt att tänka sig att antalet OS-utvecklare också ökar (Josh & Valloppillil, 1998; Mantarov, 1999). Ett exempel som skulle kunna verifiera detta påstående är Tuomis studie av utvecklingen av Linux-kärnan (Toumi, 2001). I sin studie påvisar Toumi att utvecklingen varit mycket snabb och att tillväxten ökat sedan starten av Linux 1991. Även Dempsey, Greenberg, Jones och Weiss (1999) har genom kvantitiva studier av Linux-arkiv på Metalab i USA visat på ett mycket stort deltagande i utvecklingen av bland annat Linux.

Om tesen ovan stämmer kommer det också att bli lättare att utveckla OSS och även perifer OS-programvara. Som följd av detta skulle då Open Source-programvara utgöra ett allt mer komplett alternativ till proprietär programvara (Josh & Valloppillil, 1998; Mantarov, 1999). Detta till trots att alla som bidrar inte är kompetenta utvecklare. Feller och Fitzgerald (2002) menar att så många som en av fem utvecklare är negativa bidragsgivare, det vill säga deras bidrag ger de övriga utvecklarna mer jobb i form av administration och testning av programkod som inte är tillräkligt kvalitativ.

Vidare talar också argumentet ”användarna är utvecklare” för Open Source enligt flera källor (se till exempel Dalle & Julien, 2002; Feller & Fitzgerald, 2002; Moody, 2001). Denna egenskap innebär att de som utvecklar programvaran ofta är de samma som använder den. Flera källor pekar på detta som en fördel för Open Source-programvaran (bland andra referenserna ovan) då programvaran till exempel blir utförligt testad. Till detta kan man eventuellt lägga Raymonds (2002) åsikt att Open Source-utvecklare kommer från toppskiktet bland världens programmerare. Detta är givetvis omöjligt att bevisa, dock är frontfigurerna inom Open Source-rörelsen några av de mest ansedda programmerarna enligt bland annat Cook (2001).

Sammanknutet med de två föregående punkterna är återanvändning av kod och kumulativ intelligens. Återanvändningen av kod är en fördel som enligt bland annat Raymond (2001) följer med att källkoden finns tillgänglig och sålunda slipper utvecklare uppfinna hjulet igen (bland annat Feller & Fitzgerald, 2002). Den kumulativa intelligensen sägs vara ett resultat av att utvecklare med olika bakgrund och kunskaper ”träffas” och sammanför sina

kunskaper efter intresse (Feller & Fitzgerald, 2002; Porter & Schmidt, 2001; Mantarov, 1998)

Som nämndes under kapitlet ”Linus lag” är parallell utveckling och felsökning Open Source-projektens kanske viktigaste karakteristika. Med en stor skara buggtestare anses (se bland annat Feller och Fitzgerald, 2000 och DiBona et al, 1999) chansen öka att dels hitta fel i programmen och dels att lösa dem. Denna buggtestning är dessutom i hög grad accepterad som en del av processen. Kommersiella företag sägs enligt till exempel Raymond (2001) ofta ha svårt att uppnå denna grad av parallell felsökning.

Gentemot slutanvändare uppvisar OSS enligt bland andra Porter och Schmidt (2001) ett viktigt drag: den låga inköpskostnaden. Detta är en måhända uppenbar följd av att programvaran oftast är fritt tillgänglig som påpekas från många håll (se bland annat Raymond, 2001; Feller & Fitzgerald, 2000; 2002, mfl).

Vad gäller OSS svagheter finns huvudsakligen tre punkter som påvisas i litteraturen: avsaknad av användarorientering, ansvarsfrågan och support. Avsaknad av användorientering innebär enligt bland andra Behlendorf (i DiBona et al, 1999) att Open Source-utvecklarna sällan är talangfulla utvecklare av grafiska gränssnitt alternativt att de saknar intresse för aspekter av denna typ. Detta leder till att programmen blir svåranvända för icke-tekniker. Möjligtvis kan det också vara en följd av att programmen ofta tillkommer som följd av att en utvecklare har ett eget behov av programmet (”scratching a developer's personal itch”, enligt Eric Raymond) och sålunda inte fokuserar på andra användare än sig själv (Osterhout, 1999). På senare tid har dock ett antal initiativ för att göra programvaran mer lättanvänd växt fram. Exempel på detta är grafiska användargränssnitt som KDE och Gnome.

Det faktum att det inte finns någon att ställa till svars för programvaran anses också som negativt av flera bedömare (se bland annat Hubley, 2001 och DiBona et al, 1999) Andemeningen i detta är att användaren till exempel inte säkert kan veta hur programvaran kommer att utvecklas vilket leder till osäkerhet (Hubley, 2001).

Support anses ofta vara svårtillgänglig för vanliga användare (Osterhout, 1999). Ett exempel som studerats av von Hippel och Lakhani (2000) är webbservern Apache där utvecklarna uttryckligen undanber sig frågor från användarna och istället hänvisar till nyhetsgrupper. Det är tydligt att detta är det vanligaste sättet att hantera support på och supporten är enligt flera (se till exempel Porter & Schmidt, 2001) god men troligtvis svårtillgänglig för företagsanvändare.

4.3 Affärsmodeller

”Öppna system är ett allvarligt hot mot Microsofts affärsmodell och kan tvinga dem att sänka priserna på sina program”, skrev Computer Sweden (2003) nyligen i en artikel om Microsofts senaste kvartalsrapport till den amerikanska handelskammaren. Open Source skulle sålunda ses som ett hot mot en befintlig affärsmodell, vilket i sin tur borde innebära att företag som arbetar med Open Source-mjukvara använder sig av andra affärsmodeller en befintliga. För att kunna svara på denna uppsats fråga måste begreppet affärsmodell definieras samt några exempel på traditionella affärsmodeller ges. I slutet av detta kapitel ges dessutom exempel på ett antal affärsmodeller som ansetts vara tillämpliga på just företag som sysslar med Open Source.

Det finns ett flertal olika definitioner av vad som är en affärsmodell. Den jag valt är Paul Timmers (2000), då jag finner den tillräkligt generell för att fungera just som en övergripande definition av begreppet. Timmers (ibid.) definition av vad som krävs för att ha en klar affärsmodell består av tre huvudpunkter som lyder enligt följande:

- En arkitektur för produkt, service och informationsflöden, inklusive en beskrivning av de olika affärsaktörerna och deras roller.
- En beskrivning av de potentiella fördelarna för de olika affärsaktörerna.
- En beskrivning av inkomstkällorna.

Timmers menar dock att affärsmodellen i sig inte beskriver hur den bidrar till att hjälpa de inblandade aktörerna att uppnå företagsmål och –visioner. För att få veta detta hävdar Timmers att det förutom affärsmodellen behövs

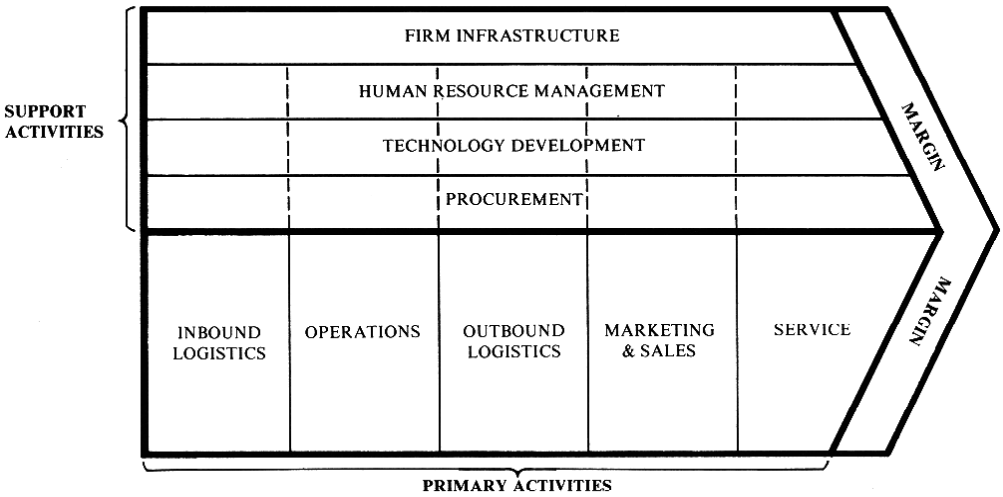
en marknadsföringsmodell som utgörs av affärsmodell samt en strategi för hur marknadsaktören skall hantera sin relationer till andra aktörer (ibid).

Jag kommer inte närmare att gå in på Timmers marknadsföringsmodell då affärsmodellen står i fokus i denna uppsats. Istället följer här istället en beskrivning av en modell för att analysera företags aktiviteter och affärsmodeller samt några exempel på grundläggande affärsmodeller som sedan kontrasteras med ett antal modeller som utländska Open Source-företag idag använder sig av.

4.3.1 Porters Value Chain

Företags verksamhet kan delas upp i ett antal aktiviteter och analyseras ur ett processperspektiv. Den kanske vanligaste metoden för att göra detta är Michael E. Porters (1985) ”Porter’s Value Chain”. I korthet består Porters analysmodell i att dela in företagets verksamhetsdelar i två huvudgrupper: primära aktiviteter och supportaktiviteter.

De primära aktiviteterna består enligt Porter (ibid.) av delar som produktion, leverans, marknadsföring och försäljning. De såkallade supportaktiviteterna består av bland annat human resourcing, teknisk utveckling samt finans- och planeringsverksamhet. Genom att utföra de olika aktiviteterna skapar företaget värde för sina kunder och värdet mäts genom den mängd pengar som kunderna är beredda att betala för slutprodukten.



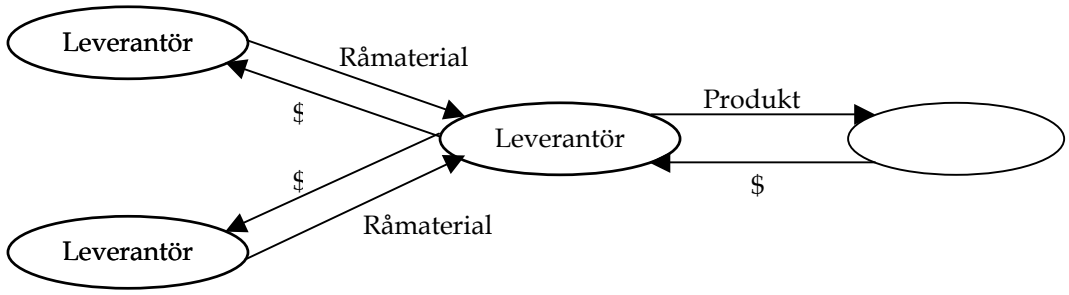
Grundläggande ”Value Chain” enligt Porter (1985).

4.3.2 Grundläggande affärsmodeller

Porters modell, som presenterats i föregående kapitel, är i hög grad lämpad för analys av ren tillverkningsindustri (Karlsson & Lund, 2000) och får i denna uppsats mer ses som ett sätt att beskriva vad en affärsmodell kan anses vara uppbyggd av. Med anledning av detta kommer jag här nedan att komplettera med ett antal grundläggande affärsmodeller baserade på MIT Sloan School of Managements definitioner.

Producera som skapande företag

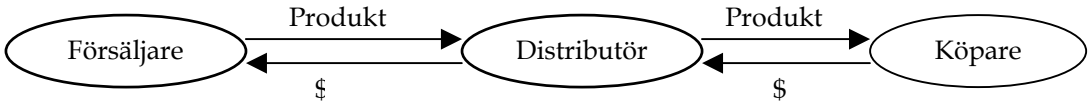
Enligt denna modell skapar företaget en produkt utifrån råmaterial från underleverantörer. Den egentliga verksamheten består alltså av att sammanfoga olika komponenter till ett för kunden funktionellt objekt. Inkomsterna för företaget genereras genom att kunden betalar produkten. (MIT, 2003)



Producera som skapande företag. (MIT, 2003)

Producera som distribuerande företag

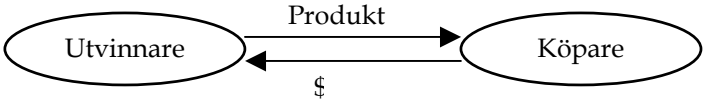
Skillnaden mot den föregående affärsmodellen är att distributören inte förädlar råmaterial och säljer vidare en produkt. Kärnan i distributörens modell är att en eller flera befintliga produkter köps in och att extra värde sedan adderas i form av till exempel support, kundanpassning eller ompackning. Inkomsterna från denna modell genereras i likhet med den föregående av att företagets kunder betalar distributören för den slutliga varan. (MIT, 2003)



Producera som distribuerande företag. (MIT, 2003)

Producera som utvinnande företag

Denna modell syftar främst på industriföretag som tillexempel gruv- och fiskeindustri. Dock menar MIT (2003) att affärsmodellen också används i tillexempel reklamfinansierad media och liknar medieprodukterna med fiskarens fiskenät. Genom att sprida produkten gratis till konsumenterna kan medieföretagen tjäna pengar på ”bieffekterna” av spridningen i form av reklamintäkter och den produkt som genererar inkomsterna är annonser.



Producera som utvinnande företag. (MIT, 2003)

Beståndsdelar i de olika grundläggande affärsmodellerna

Gemensamt för de olika modellerna är att de alla innehåller ett antal än mer grundläggande beståndsdelar:

- Köp, som inträffar när en kund betalar för att få en tjänst eller vara. För att ett köp skall kunna ske har säljaren bland annat identifierat potentiella kunder såväl som leverantörer.
- Skapande, vilket innebär att den säljande parten skapar en vara eller tjänst som sedan kan säljas. När inte varan/tjänsten skapas från grunden, utan består av modifieringar eller sammanfogningar av inköpta varor/tjänster, ingår design och utveckling av produkter och processer i denna beståndsdel.
- Försäljning, det vill säga när kunden köper varan eller tjänsten från leverantören. Detta steg består i sin tur av delmoment som att säljaren identifierar kunder och deras behov samt kommer i kontakt med dessa. Även betalning, leverans och hantering av kundrelationer är delmoment i försäljning enligt MITs (2003) synsätt.

4.3 Internationella exempel på OSS-affärsmodeller

Det teoretiska underlaget där man identifierat olika affärsmodeller bland OSS-företag är enligt vad jag kunnat se begränsat. Man kan dock utläsa ett antal olika modeller med ledning av det som Hecker (2000), Ljungberg (2000), Moody (2001), Raymond (2001) och Wayner (2000) skrivit. Här nedan följer fem stycken modeller som är applicerbara på ett antal olika Open Source-företag.

Kringtjänster, service och support
Inom ramen för denna affärsmodell ryms företag som sysslar med kundanpassning av OSS, installation och utbildning. Intäkter av denna modell skapas främst på två sätt: fysiska produkter i form av till exempel dokumentation och anpassade CD-skivor med program samt tjänster som till exempel teknisk support. Exempel på företag är Red Hat och MandrakeSoft vilka båda har egna Linux-distributioner och sålunda utvecklar viss egen programvara men släpper även denna under Open Source-licens. Min tolkning av MITs grundläggande affärsmodeller är att denna modell till största del ryms inom ”Producera som distribuerande företag” men även till viss del inom ”Producera som skapande företag”, då exempelvis Red Hat och MandrakeSoft skapar viss programvara själva. I många fall handlar det dock om att ta befintliga produkter och anpassa dessa till kundens behov, till exempel genom ökad användarvänlighet eller genom att lägga till supporttjänster.

Hybridmodeller
Under denna kategori finns det som på engelska ofta kallas ”Loss Leader”. Genom att släppa en del av mjukvaran eller relaterad mjukvara fri kan företag öka intresset för sin kommersiella programvara och få fler användare. Exempel på företag som använder denna affärsmodell är Sendmail Inc. som distribuerar och utvecklar sitt program Sendmail under Open Source-licens men även erbjuder kommersiell, proprietär, mjukvara som gör Sendmail mer användarvänlig. Ett annat exempel är Cygnus Solutions som finns omnämnt tidigare i uppsatsen. Inkomsterna kommer uteslutande från den kommersiella programvaran och eventuellt kombineras

modellen med support- och servicetjänster. Denna modell har klara likheter med MITs ”Producera som utvinnande företag” då man just sprider produkter gratis och sedan tjänar pengar på de kommersiella produkterna som utvecklats som komplement till Open Source-programvaran. OSS fungerar här till viss del som reklam för de produkter som säljs, på samma sätt som i exemplet med medieföretagen under stycket ”Producera som utvinnande företag” i förra kapitlet.

Open Source som stöd för hårdvara
På engelska kallat ”Widget Frosting” och innebär att hårdvaruutvecklande företag använder Open Source som modell för att sprida till exempel drivrutiner som krävs för att man skall kunna använda deras hårdvara. Inkomsterna från denna affärsmodell kommer i första hand från hårdvaran och mjukvaran är en extra kostnad för företaget. Genom att sprida den tillhörande mjukvaran fritt kan dra fördel av användarnas bidrag till mjukvaran och förhoppningvis hålla nere sina egna kostnader för mjukvaruutveckling.

En annan variant av ”Widget Frosting” är att sälja hela datorer, till exempel serverlösningar, utvecklade för att använda Linux , BSD eller andra Open Source-operativsystem. Här kan Open Source användas dels för att hålla nere kostnaderna och dels som draghjälp för hårdvaran. Exempel på företag som använder sig av denna modell är IBM.

Denna modell kan liknas vid ”Producera som skapande företag” och Open Source-programvaran kan snarare sägas vara en del av marknadsföringen för produkter. Användandet eller producerandet av Open Source-produkter kan helt enkelt ses som en ett sätt att öka den potentiella marknaden för själva produkten.

Tillbehör, från t-shirts till böcker
Denna modell kan kanske verka marginell men i Open Source-fallet är den inte det. Genom att sälja mer eller mindre nödvändig ”kringutrustning” som appellerar Open Source-utvecklare får företagen intäkter. Exempel på ett sådant företag är O’Reilly & Associates som främst säljer böcker inom programmering, OSS och OSS-kultur. Ett annat känt företag är ThinkGeek

som säljer t-shirtar, böcker och livsstilsartiklar. Närmast till hands för att klassificera denna modell enligt MITs grundläggande modeller ligger kanske distribuerande företag men i vissa fall skulle man kunna se dem som utvinnande företag. Skillnaden här mot till exempel ”Hybridmodeller” är att företagen som säljer livsstilsprodukter och dylikt är att de snarare drar nytta av de affärsmöjligheter som Open Source-produkterna genererar och inte själva producerar några produkter att sälja tillbehör till.

Servicefrämjare
Modellen går ut på att tjäna pengar på Internettjänster och sprida den nödvändiga klientprogramvaran under Open Source-licens. Netscapes Netcenter är ett sådant exempel där företaget genom fri programvara ger tillgång till tjänster som genererar intäcker genom reklam och liknande. Även filbytarprogramvara som den Napster-liknande Limewire kan sägas tillhöra denna kategori. I likhet med hybridmodellen som beskrivits tidigare i detta kapitel anser jag att denna modell kan sägas vara av typen ”Producera som utvinnande företag”. Precis som hybridföretagen producerar företagen enligt denna modell dels gratis Open Source-programvara men pengarna tjänar man på till exempel serverprogramvara som säljs, oftast utan källkod.

Förutom de affärsmodeller som finns representerade ovan har bland annat Hecker (2000) beskrivit ytterligare ett antal som han anger som teoretiskt möjliga bland annat olika varianter av varumärkesbaserade modeller. I dessa modeller förutsätts ett såpass starkt varumärke att programmen kan släppas under Open Source-licens och företaget kan få tillräkliga intäkter enbart genom sitt starka varumärke och försäljning relaterat till detta. Jag har valt att inte nämna dessa här då de saknas företag som tillämpar dessa modeller idag.

Kapitel 5 – Respondenternas utsagor

I detta kapitel presenteras ett urval av den information och de erfarenheter som framkommit under intervjuerna med respondenterna. Detta ligger, tillsammans med tidigare delar av uppsatsen, till grund för diskussion och slutsatser. Under första delkapitlet ges en kortare beskrivning av företagen, huvudsakligen utifrån respondenternas egna ord. Andra delkapitlet fokuserar på resultatet av intervjuerna, kopplat till syfte och frågeställningar.

5.1 Beskrivning av de studerade företagen

I detta kapitel beskrivs de studerade företagen och deras bakgrund kort utifrån hur de själva presenterat sig, dels i de intervjuer som gjorts med företagens företrädare men också i material hämtat från deras hemsidor.

Cendio Systems AB, Linköping
Cendio Systems startades 1992 av studenter vid Linköpings universitet. Företaget hade när det startades, enligt Inge Walling vice VD för Cendio Systems AB, en tydlig ideologisk grund. Han säger att ”det tilltalade min etik att man delade med sig och så tänkte jag att det skulle man kunna tjäna pengar på det.” Open Source var sålunda en viktig faktor till att han och de övriga grundarna bestämde sig för att starta företaget. Företagets affärsidé i korthet är ”Att hjälpa företag och organisationer att använda Linux och Open Source för att stärka deras konkurrenskraft.”

Cendio skriver att de ”erbjuder service och lösningar baserade på Linux och Open Source” (Cendio Systems, 2002a). Enligt företagets hemsida har de ”specialistkompetens inom systemintegration” (ibid) och ”arbetar med högkvalitativ utveckling, integration och service baserat på Open Source Software” (Cendio Systems, 2002b).

I dagsläget har Cendio Systems 39 anställda och en omsättning på 22 miljoner kronor (Affärsdata, 2003).

Codefactory AB, Umeå
Företaget startades, enligt CEO:n Patrik Fransson, av ”två personer som varit djupt involverade i ett antal Open Source-projekt under många, många år och hade ett ganska brett kontaktnät [inom Open Source-rörelsen]”. Idag är företaget delägt av bland andra WM-Data.

CodeFactorys paroll är “Bringing Free Software to The Enterprise” (CodeFactory, 2002) och de beskriver sig själva som det ledande Open Source-konsultföretaget i Skandinavien. Vidare skriver de att de utvecklar, sprider och anpassar Open Source-mjukvara samt hjälper företag och organisationer att förstå och utnyttja fördelarna med Open Source-mjukvara (ibid).

Codefactory har 18 anställda och omsatte 6,2 miljoner kronor under 2001 (Affärsdata, 2003).

Raditex AB, Stockholm
Företaget startades 1987 av nuvarande VD:n Göran Hasse och arbetade i början huvudsakligen med olika Unix-system men har under 90-talet och framåt arbetat mer och mer med Linux och BSD (Hasse, 2002).

Raditex presenterar sig med att ha tre olika inriktningar: ”Utbildning”, ”Konsult” och ”Control” (Raditex, 2002a). De skriver bland annat att de ”förser marknaden med kvalificerad kompetens inom området öppna system”(ibid) och att ”konsultverksamheten är ledande i landet inom Linux och FreeBSD” (ibid).

Raditex AB har idag tre anställda och omsätter drygt 3,3 miljoner kronor (Affärsdata, 2003).

South Pole AB, Stockholm
South Pole startades 1999 av ett antal personer med civilingenjörsutbildning vilket enligt VD:n Jakob Sandgren (2002) troligen gjort att de har en mer tekniknära utveckling än andra svenska Open Source-företag.

I sin presentation skriver South Pole att de ”kombinerar det bästa av två världar och erbjuder produkter och tjänster kring det fria operativsystemet Linux tillsammans med servrar baserade på standardkomponenter” (South Pole, 2002). Företaget presenterade även en egen serie datorer som ”är specialanpassad för Linux och levereras med Linux förinstallerat” (ibid). Vidare skrev de att de levererar ”nyckelfärdiga lösningar som till exempel Linuxkluster”, det vill säga en mängd datorer med Linux som operativsystem som kopplats samman för att öka beräkningskapaciteten (ibid).

South Pole AB har fyra anställda och en omsättning på 3,5 miljoner kronor (Affärsdata, 2003).

5.2 Företagens verksamhet och koppling till Open Source

I detta kapitel kommer jag att närmare granska de studerade företagens verksamhet med fokus på de eventuella särskiljande egenskaper som kan kopplas till Open Source.

Cendio Systems AB
Som nämndes i föregående delkapitel startades Cendio mycket på grund av att grundarna var intresserade av Open Source och ville arbeta med det yrkesmässigt. I dagsläget finns fokus på Open Source kvar inom de bägge verksamhetsgrenar som Cendio, enligt vice VD Inge Wallin (2002), har. Den ena verksamheten kallar man ”Service providing”, det vill säga installation av servrar, utbildning och planering av systemmiljöer. Den andra grenen är mer specialiserad och benämns av Wallin som design av inbyggda system. Inom bägge dessa verksamheter menar man att man drar nytta av Open Source.

När det gäller ”Service providing” för Wallin främst fram ekonomiska motiv för det egna företaget, respektive ekonomiska fördelar för slutkund, som ett tungt argument för Open Source. Han illustrerar sin bild av förhållandet som gör att OSS är att föredra utifrån Cendios ”Service Providing”-verksamhet (nedan kallat Solution provider):

	Mjukvaruutveckling	Distribution	Solution provider
Icke OSS	65%	5%	30%
OSS	0%	5%	95%

Tabell 2: Fördelning av kundens totalkostnad för installerad mjukvara enligt Wallin.

Essensen i tabellen ovan är att Wallin hävdar att företag som är beroende av inköpt proprietär programvara måste lägga mer resurser på mjukvaruutveckling, vilket i praktiken ofta är likställt med inköp av programvara. Om man förutsätter samma pris mot slutkund för en OSS-lösning som för en icke OSS-lösning menar Wallin att Solution providern, i OSS-fallet, kan ha avsevärt större marginal då kostnaden för programvaran är liten eller obefintlig. Skall man hårdra detta innebär det att man i OSS-fallet kan sälja lösning baserad på programvara som inte kostat något i inköp. Detta medför enligt Wallin att man inom ramen för Solution providing även kan utveckla och integrera programvara och ändå göra en större procentuell vinst.

Likaså menar Wallin att detta förhållande är något de kan överföra till slutkunden:
”Om man jämför med en Microsoft-partner så kan ju vi kanske ta ut mer. Vi kanske kan få kunden att betala hälften istället för en tredjedel för lösningen och så kan vi ta ut lite mer för lösningen än de gör och det är ändå hälften så dyrt för kunden. Och då är ju konsten att hitta de tjänster som kunden är villig att betala för.”

Den andra verksamhetsgrenen, inbyggda system, har man i dagsläget brutit ut från kärnverksamheten genom att starta ett eget företag för denna typ av verksamhet. Anledningen till uppdelningen i två företag var att man bättre ville kunna fokusera verksamheten inom respektive företag. Cendio Systems sysslar alltså i dagsläget främst med det som Wallin kallar ”Service providing”. Wallins resonemang kring företagets fördelar när det gäller inbyggda system skiljer sig dock något från det kring ”Service providing”. Fördelen som Cendio har inom inbyggda system, genom sitt val av OSS, är till stor del den att de kan återanvända/vidareutveckla redan utvecklad

programvara och dra nytta av denna istället för att utveckla helt ny programvara.

Wallin summerar hur han ser på utvecklingen av Open Source, licenserna och hur utvecklingen går framåt: ”[…] i och med att det är Open Source, det vill säga att källkod alltid följer med, så kan man ju ta tag i det där problemet och det är ju en delmängd av folk som gör det. Så de lägger till lite grand och då har ju de samma problem som ursprungspersonen här, vad gör man med det här tillägget då? Licensen på det här programmet är ju ofta skriven så att om du släpper någonting så måste du också släppa tilläggen under samma villkor som ursprunget var. Men det står ju å andra sidan inget skrivet om att du måste släppa det. Så du kan ju mycket väl hålla det här för dig själv. Problemet som du råkar ut för är att om det kommer nya versioner så måste du ofta uppdatera din uppdatering och det kan ju bli ett rätt stort underhållsarbete till sist. Så det vanliga är naturligtvis att släppa även det här och då tänker man sig att de här grejorna ska stoppas in som en extra feature i originalprogrammet och sedan alltid följa med i fortsättningen. Och det är nästan alltid vad som händer också.”

För att summera Cendios verksamhet är de alltså en ”Service provider” med viss aktivitet inom inbyggda system. Cendio tjänar dels pengar på att sälja tjänster kring OSS och dels genom att paketera och anpassa OSS så att det passar deras slutkunder. Man använder OSS inom alla sina verksamheter och har ett antal argument för att göra detta, bland annat pris, återanvändning av kod och kollaborativ programvaruutveckling.

Jag anser att man kan se ett antal faktorer som särskiljande för företaget genom deras användande av Open Source: bakgrund, kostnadsfördelning vid produktutveckling och företagets inriktning. Bakgrunden, företagets historia, är tätt knuten till Open Source-rörelsen och jag tycker mig skönja en ideologisk förankring inom Open Source-rörelsen. Denna koppling stärks ytterligare av att företaget kontinuerligt deltar i utvecklingen av ett antal Open Source-programvaror som till exempel Linuxconf (konfigurationsprogramvara för Linux) och Free/SWAN (säkerhetsprogramvara för Linux) (Gnuheter, 2001). Kostnadsfördelningen vid utveckling av mjukvara är något som Wallin under intervjun framhävt

som en skillnad mellan Open Source-företag och företag som sysslar med proprietär programvara. Slutligen verkar företagets inriktning, service providing, vara en följd av att de sysslar med OSS och sålunda får svårare att ägna sig åt försäljning av programvara.

Codefactory AB, Umeå

I likhet med Cendio Systems startades Codefactory av personer med ett intresse för OSS. Sedan företaget startade har man fått en klar uppdelning i vad de själva kallar för ”tre ben: strategi, teknik och collaboration” (Fransson, 2002). Strategidelen av verksamheten innebär oftast strategier för produktföretag ”som har en produkt och funderar på Open Source”. Codefactory arbetar i dessa fall med att utreda vilken eventuell nytta dessa företag skulle ha av att välja Open Source som modell för sina mjukvaruprodukter.

Den andra verksamhetsgrenen, teknik, beskriver VD:n Patrik Fransson med att de ”hjälper företag att bygga en infrastruktur” men även programutveckling och systemutveckling. Skillnaden mellan Codefactory och andra IT-företag är enligt Fransson den process som föregår själva utvecklingsarbetet. Han beskriver processen med tre steg: ”ett, har det här lösts någonstans förut? Två, får vi då använda det för våra syften? Tre, får vi då göra förändringar i programvaran? Och fyra, tycker vi att det är kvalitativt?”

Företagets tredje område är kanske det som är mest signifikant för Codefactory. Det innebär att ”[…] hjälpa företag och organisationer att jobba på det sättet som […] Open Source-projekten gör med systemutveckling. Alltså distribuerad systemutveckling eller collaboration eller vad man vill kalla det för.” Man försöker alltså att överföra det arbetssätt som Open Source-projekt ofta använder till företagsvärlden genom att anpassa till exempel arbetssättet med distribuerad systemutveckling från Open Source-projektens verklighet till de företag man har som kunder. Fransson menar att detta arbetssätt kan lämpa sig till exempel för företag som har utvecklare spridda på flera geografiska platser och behöver samordna utvecklingen. Värt att nämna är att detta område inte har direkt att göra med huruvida programvaran som utvecklas är Open Source eller inte, utan snarare

fokuserar på utvecklingsmetoderna som används inom Open Source-rörelsen.

Fransson sammanfattar beskrivningen med ”Open Source är vår kärnkompetens. Det är viktigt för oss att inte framstå som ett Linux-företag […] för oss är det Open Source, både teknik, metod och filosofi egentligen.” Han menar också att de försöker arbete långsiktigt för att Open Source skall få fäste och Codefactory är bland annat med och driver projekt som ”Linux i skolan” tillsammans med Skolverket. Fransson tilläger också att ”det är långsiktigt, det är inga affärer för oss imorgon eller så, utan det är mer så att bearbeta marknaden i viss riktning och öka medvetandet.”

Just fokus på de icke-tekniska faktorerna kring Open Source är något som utmärker Codefactory bland de intervjuade företagen. Samtidigt som man menar att Open Source genomsyrar företaget hävdar man att det inte är så speciellt egentligen: ”Generellt när vi är ute hos kunder […] så försöker vi avdramatisera det här. Vi tycker definitivt inte det är något speciellt med Open Source. Vi tycker inte det är speciellt spännande eller anmärkningsvärt eller konstigt eller någonting. Vi är ett normalt företag […] som fungerar på normala villkor som vilket företag som helst. Och det vi jobbar med är ganska vanliga saker, som vi vet att de är stabila och de fungerar bra och hela det paketet. Plus att vi tycker att , våran poäng är att det är en fördel för dig som kund. Det är så mycket bättre för dig som kund att gå på Open Source-produkter.”

De ideologiska undertonerna inom Open Source är inte heller något man lägger stor tonvikt vid: ”Vi jobbar för våra kunder, inte för någon ideologi eller filosofi, utan det vi baserar det här på är att vi tycker det här är bästa lösningen helt enkelt. […] Det är klart att alla i bolaget har ju i ryggmärgen det hära distribuerade och nätverkande sättet att jobba man behöver inte ses varje dag för att kunna jobba ihop i gemensamma projekt och vi har ju folk som sitter både i Stockholm och Göteborg jobbar med varandra dagligen och nästan aldrig lyft en telefon och pratat med varandra utan all kommunikation sker då genom nätet […].”

Franssons markerar samtidigt tydligt att man vill arbeta med alla de olika delarna av Open Source, inte enbart programvaran. Den uttalade kopplingen till dessa olika delar av Open Source-rörelsen är kanske det mest särskiljande för Codefactory. Även det fokus man har på att sprida metoderna och filosofin från Open Source-världen gör att man skiljer sig något från traditionella företag. Även när det gäller huruvida kunderna bör eller inte bör använda Open Source har man en tydlig linje: ”Vi försöker alltid styra kunden så att de driver det [utvecklandet av mjukvara] helt öppet. […] Men vi har inget problem med att de vill hålla det inom sina fyra väggar. […] Men oftast är det GPL eller liknande Open Source-licens som vi ändå utvecklar det här under, om de [kunderna] då tagit steget att visa upp det […]” (Fransson, 2002)

Codefactory är ansvariga för och driver utvecklingen av bland andra Open Source-produkter som MrProject, ett projektplanerings- och kalenderverktyg som liknar Microsoft MsProject, och RoadRunner, ett ramverk för utveckling av nätverksapplikationer. MrProject är ett av företagets flaggskepp och ingår bland annat i det populära grafiska gränssnittet Gnome för Linux.

Raditex AB, Stockholm

Raditex som företag skiljer sig något från de andra intervjuade företagen med hänseende på företagets ursprung. Medan de andra företagen mer eller mindre kommer ur ett direkt intresse för Open Source hos grundarna har Raditex växt fram ur den proprietär Unix-världen. VD:n, Göran Hasse, beskriver det med följande ord: ”[…] jag har inte haft någon utpräglad åsikt egentligen från början om att hålla på men källkodsöppna system bara sådär, utan det har mer blivit så. […] mer praktiskt betingat. Kunderna finns där idag.”

Även Raditex poängterar dock vissa av egenskaperna hos Open Source som avgörande för tankarna bakom företaget. En viktig sådan egenskap är öppenheten. Hasse beskriver den slutna källkoden, till skillnad från Open Source, med ”att säga: vi har gjort elinstallationer i ditt hus. Men du får inga elritningar utan det håller vi hemligt. Eller motsvarande då: Vi har byggt ett, ett VVS system i din fastighet men du får inga ritningar till det, för de vet

bara vi. Så när det blir problem med dina avlopp så får du ringa till oss så kommer vi. […] öppen källkod handlar om att leverera med ritningarna till det man har byggt också. […] egentligen är det en hederlighetssak. Det här har jag byggt åt er. Och såhär har jag byggt det."

En viktig faktor enligt Hasse är också det han kallar för starthöjden, det vill säga på vilken nivå man börjar utveckla datorprogram idag. Han exemplifierar med: "Man startar med en SQL-databas istället för att börja skriva drivrutiner för att hantera datalagringen. [...] Det är bara starthöjden som har ändrats. Sen därifrån så är verksamheten som förut och det är väl egentligen väldigt sund och väldigt naturlig bit på utvecklingen." Han menar att Open Source spelar en viktig roll här eftersom det driver utvecklingen framåt med inkrementell utveckling.

Även Göran Hasse för fram kostnadsargumentet som positivt för Open Source men även konkurrensfaktorer. Han menar att Open Source inte betyder något direkt för slutkonsumenten. I det ledet är det snarare användarvänlighet och integration som betyder något. Priset anser han dock vara en genomgående viktig faktor för att marknadsföra Open Source-produkter: "det sticka i ögonen att fan vad dyrt det har blivit ändå, jag kan köpa SuSE-linux här för 700 kronor. Och installera det på 5 datorer. […] Men det tror jag inte har med öppen källkod att göra utan det har med prisbilden bara att göra. Dom är lite greedy där i, alltså Microsoft, dom vet om att dom sitter i en väldigt stark position så utnyttjar dom den också då och det är klart då märks det ju naturligtvis."

I övrigt är dock Raditex det företag som kanske i lägst grad pekar på särskiljande faktorer i sin verksamhet som går att relatera till Open Source. Hasse menar att verksamheten inte ändrats nämnvärt för att man numera främst sysslar med Open Source, framför olika proprietära Unix-varianter. Han pekar dock på faktorer som utebliven licenshantering, genom att Open Source-licenserna inte innehåller några klausuler om kopieringsförbud och liknande, samt minskade kostnader för programvara som en positiv följd av att företaget använder Open Source inom konsult och utbildningsverksamheten.

South Pole AB, Stockholm

När South Pole startades var det i mångt och mycket ett tekniskt intresse och ett intresse för operativsystemet Linux som låg bakom. VDn för South Pole, Jakob Sandgren, berättar om anledningen till att de valt Linux och Open Source som bas för verksamheten: ”Jag är i grund och botten tekniker, så intresset har väl alltid funnits där. Sedan upptäckte man väl någon gång för tre år sedan att nu börjar det lossna [...] Det var väldigt mycket Linux ett tag och det gjorde väl även att många inom industrin [...] fick upp ögonen för det och såg att det faktiskt var något som det kan bära sig att jobba med. Så det var därför som vi startade.”

I South Pole:s fall spelade insynen i de tekniska lösningarna som den öppna källkoden för med sig en stor roll för det egna utvecklingsarbetet. Då företaget idag ägnar sig till stor del att specialanpassa drivrutiner för Linux och bygga avancerade klustersystem baserade på Linux är insynen i källkoden viktig för företagets möjligheter, enligt Jakob Sandgren. ”Även om vi har fokus på Linux så känner vi att har fokus på ett visst område där och det är ju djupt nere i Linuxkärnan med avancerad utveckling av drivrutiner och kärnan i sig”, menar Sandgren.

Förutom de rent tekniska fördelarna som Sandgren hävdar, ser han också fördelar i personalrekrytering. Han menar att det är lättare att hitta eldsjälar inom Open Source- och Linux-världen än vad det är inom den proprietära mjukvaruvärlden. Även om han tror att liknande personer går att hitta inom Windows-världen tror han att om ”man [ser] inom Linux och Open Source-sidan så är det ju betydligt enklare.”

I övrigt ser Sandgren liten skillnad mellan South Pole och andra företag, verksamheten skiljer dock lite tror han: ”[...] eftersom vårt arbete baseras ju egentligen mer på andras insatser än på våra egna [skrattar] om man ska se det krasst. Utan Linux skulle vi inte hållit på med det vi gör idag. […] Som företag och företagsstruktur och så tror jag nog inte att det [skiljer sig], inte som man kan härhöra till själva Open Source-biten i alla fall. Det […] ser väl ungefär likadant ut som hos de andra [dataföretagen].”

South Pole utvecklar ingen egen mjukvara som marknadsförs externt. Dock har man en datorserie som är ”specialanpassad för Linux och levereras med Linux förinstallerat”.

Sammanfattning

Om man kort skulle summera de fyra företagens relation till och användning av Open Source har jag i tabellen nedan fokuserat på ett antal kategorier. Nedan har jag ordnat dem efter ett antal karakteristika som framkommit i intervjuerna. Faktorerna grundar sig på hur respondenterna sett på ett antal frågor som behandlats i intervjuerna.

	Cendio Systems	Codefactory	Raditex	South Pole
Ursprung	OS	OS	Ej OS	OS
Endast OSS	Nej	Nej	Nej	Nej
OSS-argument	Kostnadsfokus och inkrementell utv	Kostnad, arbetssätt och inkrementell utv	Öppenhet, kundbas inkrementell utv	Stabilitet, integration och inkrementell utv.
Rekrytering	-	Inom rörelsen	-	Större tekniskt intresse

Tabell 3: Kort sammanfattning per företag

Ursprung: hur företaget startades. Av personer med stort intresse för Open Source (OS) eller på annat sätt (ej OS).
Endast OSS: sysslar företaget uttalat endast med Open Source Software.
OSS-argument: de argument som företaget har för användning av Open Source, såväl internt mot anställda som externt.
Rekrytering: hur har rekrytering av anställda skett, alternativt på vilket sätt underlättar arbetet med Open Source rekrytering av personal.

5.3 Analys av affärsmodeller

Utifrån de affärsmodeller som beskrivits i kapitel 4.3 kan de fyra företagens affärsidéer huvudsakligen kategoriseras inom modellen ”Kringtjänster, service och support”. Alla fyra inriktar sig mer eller mindre på att erbjuda IS/IT-lösningar baserade på Open Source-programvara. Och även om inriktningarna ändå skiljer sig något kan sägas att de sysslar med systeminstallation, vidareutveckling/anpassning av programvara, utbildning och support. I ett fall arbetade företaget även med något som kan anses falla under ”widget frostning”, det vill säga Open Source som stöd för hårdvara. I praktiken innebör detta att de erbjuder specialkomponerade datorer där de garanterar att hårdvaran fungerar med viss utvald Open Source-programvara.

Här följer en kort genomgång företag för företag:

Cendio Systems: uttalat service- och supportfokus. I intervjun framställs Cendio främst som en ”Solution provider”, det vill säga man tillhandahåller mervärde till OSS genom att paketera, installera och designa lösningar baserade på OSS.

Codefactory: i likhet med Cendio har Codefactory ett uttalat service- och supportfokus. Till detta kommer ett större fokus på egenutvecklad programvara som MrProject vilket teoretiskt skulle kunna möjliggöra en hybridmodell där företaget gör delar av programvaran tillgänglig under OS-licens. Under intervjun har dock denna utveckling avvisats med hänvisning till att företaget snarare skulle förlora på att ägna sig åt icke OSS-utveckling.

Raditex: även Raditex faller inom ramen för ”Kringtjänster, service och support”. De har dock ett kraftigare fokus på utbildningsdelen av modellen i det som framkommit under intervjun.

South Pole: förutom ”Kringtjänster, service och support” kan South Pole även sägas ägna sig åt ”Widget Frosting”. Även om de inte har egen hårdvaruutveckling så utvecklar de en egen datorserie med komponenter som de vet fungerar bra med olika Open Source-programvaror. Förutom den

egna datorserien tillämpar South Pole samma arbetssätt när de säljer klusterlösningar.

I stort kan alltså sägas att företagens affärsinriktning överensstämmer med de som finns beskrivna i litteraturen och där igenom påvisats internationellt. De sysslar främst med det som ryms inom beskrivningen ”Kringtjänster, service och support”, en kategori dit till exempel RedHat räknats i litteraturen. Bland företagen i denna studie verkar dock den egna mjukvaruutvecklingen till stor del vara sekundär och de bygger främst verksamheten på programvara utvecklad av andra. Om man jämnför med företaget RedHat som omnämns under kategorin ”Kringtjänster, service och support” (Hecker, 2000) är RedHat känt mest för sin produkt RedHat Linux medan de studerade företagen rent generellt har en mindre produktorienterad strategi. Istället för att ha fokus på en huvudsaklig produkt arbetar de med support och utveckling av flera mindre produkter samt tillhandahåller specialdesignade lösningar med olika Open Source-programvaror. I sken av detta vore det kanske mer lämpligt att dela kategorin ”Kringtjänster, service och support” i två olika delar: dels de företag vars arbete huvudsakligen är fokuserat på en produkt och dels de företag som med hjälp av flera, inte nödvändigtvis egenutvecklade, mindre produkter konstruerar kundanpassade lösningar.

En möjlig förklaring till denna skillnad mellan de studerade företagen och de som beskrivits i litteraturen är deras storlek. RedHat är ett relativt stort, börsnoterat företag, med tillräkliga resurser för att kunna marknadsföra en enda produkt och förvänta sig göra en nettovinst på detta. De företag jag studerat är relativt små och har därmed inte dessa resurser, samtidigt som de arbetar i högre grad med traditionell IT-konsultverksamhet och löser av kunden specificerade problem med hjälp av OSS.

En faktor som möjligtvis skiljer de studerade företagen från företag som sysslar med proprietär programvara är den att de ägnar mycket arbete åt att anpassa programvaran och bygger kompletta lösningar med hjälp av flera, ofta av företaget anpassade, mjukvaror. En tydlig tendens bland de studerade företagen är att man vill vara effektiva genom att inte ”uppfinna hjulet igen”, det vill säga utveckla programvara som redan existerar under

Open Source-licens. Finns programvaran väljer man istället att anpassa och utöka den efter behov för att sedan i enlighet med Open Source-traditionen göra ändringarna tillgängliga för andra.

En möjlig anledning till att företagens verksamhet och affärsidé i hög grad överensstämmer med de i litteraturen beskrivna affärsmodellerna är att det helt enkelt är dessa modeller som är de mest logiska resultaten av att som IT-företag syssla med Open Source. Då de sysslar just med OSS försämras förutsättningarna radikalt för att kunna ta betalt för själva programvaran. Sålunda måste företagen inrikta sig på kringtjänsterna i form av support, anpassning och service.

Ett exempel som jag funnit hos Codefactory finns dock inte, mig veterligen, tidigare beskrivet i litteratur som en möjlig affärsmodell/-idé. Denna affärsidé är innebär att sälja utvecklingssättet som används i många Open Source-projekt (se tex. kapitel 3.3.5) till företag. I praktiken kan detta innebära att företag förser kunden med lämplig programvara och utbildar personal i sättet att utveckla såväl som i programvaran.

Affärsidéen skulle kunna summeras med att sälja kunskap om utvecklingsmetoder och sätt att samarbeta som kan tillämpas över stora geografiska avstånd. Många Open Source-projekt har utvecklare spridda över flera världsdelar och skulle därmed kunna jämnföras med transnationella företag i sin struktur med avseende på utvecklarbas. En annan metod som kan säljas är den att låta externa aktörer bidra till utveckling av företagets programvara. Hårdvaruföretag som önskar stöd för sina produkter inom fler operativsystem är exempel på sådana företag och genom att uppmuntra bidrag från frivilliga utvecklare kan företaget nå ett större genomslag för sin produkt.

Kapitel 6 – Diskussion och sammanfattning

6.1 Företagens verksamhet

Genomgående för de intervjuade företagen är att man valt Open Source som grund för verksamheten av den enkla anledningen att man ser ekonomisk bärkraft i de arbetssätt och produkter som tidigare i denna uppsats beskrivits som Open Source. Det handlar alltså inte, i första hand, om ett ideologiskt val utan ett strikt företagsekonomiskt.

I bakgrunden finns dock ofta en mer ideologisk bild, där man hävdar stora fördelar med Open Source. Fördelarna som förts fram av de intervjuade företagen överensstämmer i stort med de fördelar som kan läsas i kapitel 4.3, till exempel tillväxtmöjligheter, utvecklingstakt och återanvändning av kod. En faktor som dock poängterats extra tydligt av flera av företagen är öppenheten gentemot kund. Att man skickar med programkoden till det man byggt, även om det handlar om en färdig ”produkt” som företaget tillhandahåller. Det är i och för sig inget revolutionerande då den öppna koden är en grundsten i Open Source

Nedan följer en sammanfattning av de aspekter som framkommit i intervjuerna och som intervjupersonerna hävdat som mer eller mindre särskiljande för sina respektive företag.

6.1.1 Öppenhet

Flera av företagen i studien talar om ”öppenhet mot kunden”. Detta begrepp kan sägas innefatta bland annat synen att man genom att arbeta med öppen källkod ger kunden större valfrihet. Man menar att kunden inte köper en sluten lösning i den meningen att den inte kan eller får gå in och ändra i källkod till program och konstruktion av system. Likväl är inte kunden knuten till en leverantör eller utvecklare vilket verkar vara ett viktigt marknadsföringsargument för flera av de intervjuade företagen.

Argumentet med öppenhet kan sägas vara kärnan i mycket av Open Source-retoriken och att detta anammats av företagen är kanske inte helt oväntat. De intervjuade har i flera fall jämfört den proprietära programutvecklingen med

en ”black box”, en jämförelse som syftar på ett system som förvisso fungerar men där man inte vet hur.

6.1.2 Ekonomiska argument gentemot slutkund

Genomgående för alla företagen är att man för fram TCO, Total Cost of Ownership, såväl som initieringskostnader som ett argument för OSS när man marknadsför sig gentemot kunder. Dessa argument kan delas upp i två delar:

Dels menar man att slutkunden genom låga eller inga inköpskostnader för standardprogramvara kan sänka den totala systemkostnaden. När det gäller specialutvecklad programvara hävdar man en fördel genom att man i utvecklingsarbetet kan återanvända delkomponenter som är Open Source. Som exempel nämndes i ett fall utveckling av databaslösningar där man istället för köpa ett databassystem, eller nyutveckla, kan använda OSS som grund och på så sätt minska kostnaderna för den totala utveckling.

TCO kan, enligt flera respondenter, minska genom att kunden bland annat har möjlighet att välja andra leverantörer än den ursprungliga och sålunda i högre grad konkurrensutsätta IS/IT-verksamheten. Grunden för detta skulle vara öppenheten. Eftersom utvecklingen skett som Open Source kan en annan leverantör ta vid. Man menar också att en stor del av TCO utgörs av inköpskostnad för program. Dessa besparingar skulle då kunna användas för att stärka den interna kompetensen.

6.1.3 Garanterad kontinuitet

Ett argument som förts fram som positivt för Open Source av respondenterna är kontinuiteten i projekten, eller åtminstone möjligheten till den samma. Man menar att så länge det finns intresse för en programvara eller ett system kommer det att fortsätta att utvecklas. Man menar också att den öppna källkoden är en förutsättning för att detta skall fungera på ett tillfredställande sätt. Genom möjligheten till nya grenar i utvecklingsprojekt kan nya utvecklare ta del av den befintliga källkoden och vidareutveckla denna i en ny riktning, eller fortsätta på den fastslagna linjen för den del. Genom transparensen i projekten blir inte programvara beroende av ett företag eller en begränsad skara utvecklare. Även detta argument menar man kan användas i marknadsföringen mot kund då man kan peka på att

mjukvaran inte kommer att försvinna från marknaden alternativt upphöra att utvecklas.

6.1.4 Inkrementell utveckling

Detta anknyter till argumenten med de ekonomiska fördelarna för slutkund. Genom att programkoden får modifieras och användas av vem som helst kan lösningar som tidigare använts med framgång återanvändas i nya programvaror. Om man skulle hårddra respondenternas syn på den inkrementella utvecklingen skulle man kunna beskriva det som att man ”slipper återuppfinna hjulet”. Genom att återanvända tidigare utvecklad programkod hävdar de intervjuade företagen att man kan få lägre utvecklingskostnader vilket i sig dels är positivt för företaget självt men också ett försäljningsargument. Huruvida den inkrementella utvecklingen är framgångsrik eller inte beror givetvis på om någon tidigare utvecklat liknande programvara eller programvara som kan ingå som en del i den nya produkten.

6.1.5 Större utvecklarbas

Flera av respondenterna hävdar att Open Source har fördelar genom att företag och individer kan dra nytta av varandras framsteg och hjälpa varandra med utveckling av programvara. Ett företag som verkar på en avgränsad marknad kan släppa sin programvara under Open Source och på så sätt vinna utvecklingshjälp från andra företag eller privatpersoner som verkar på andra marknader. Genom att vara huvudutvecklare av programvaran kan dock det ursprungliga företaget hävda ett kunskapsmässigt övertag gentemot konkurrenter när det gäller till exempel specialanpassningar av programvaran. Genom att dela med sig skulle man alltså vinna konkurrensfördelar samtidigt som man får hjälp att bära utvecklingskostnader. Om flera behöver en lösning är det helt enkelt, enligt respondenterna, ekonomiskt fördelaktigt att dela på utvecklingsinsatserna.

6.2 Företagens affärsidéer

Som redovisades i resultatkapitlet sysslar de studerade företagen främst med verksamhet som ryms inom beskrivningen ”Kringtjänster service och support”. Jag hävdar att detta är en logisk följd av att man inte kan ta betalt för mjukvaran i sig utan snarare för arbetet med att anpassa och utveckla den för specifika användningsområden.

Ska man hårdra grunden för att utveckla och sälja mjukvara grundar den sig i min mening på att ingen annan kan eller vill göra samma sak lika bra, samtidigt som det finns ett tillräkligt stort intresse för mjukvaran för att försäljningen skall kunna täcka utvecklingskostnaderna. I och med att mjukvara blir ett arbetsredskap som används av fler och fler samt att antalet duktiga programmerare hela tiden ökar får fler möjlighet att utveckla sin egen mjukvara. Att tala om att möjligheten att ”utveckla sin egen mjukvara” kan verka underligt men om tillräkligt många känner ett behov av en specifik mjukvara kommer någon av dem att börja utveckla den. Genom att komma i kontakt med andra utvecklare och samarbeta med dessa kan flera enskilda personer till slut få en fungerande programvara som täcker det behov de har.

När detta är verklighet försvinner en stor del av grunden för att sälja ”bruksprogramvara”, det vill säga programvara som många har ett behov av. Om så kommer att ske beror i min mening på vilken grad av datormognad gemene man når. Om man blir medveten om att Open Source-programvara finns kommer man kanske välja att använda denna och en del av dessa användare kommer att med tiden bli utvecklare.

Klokheten i att inför ett sådant framtidsscenario förlita sig på att kunna sälja programvara och ha detta som affärsidé kan diskuteras. De studerade företagen har istället valt en väg som innebär att man satsar på support, utbildning och specialanpassning av programvara. Det vill säga sådana delar som är oberoende av om mjukvaran är Open Source eller ej. Respondenterna hävdar att det är en fördel med öppen källkod när de i sin tur ska bedriva utbildning (de kan själva grundligt sätta sig in i hur programvaran är konstruerad och förmedla detta vidare), support (genom att själva kunna

avhjälpa problem som uppkommer i mjukvaran) och specialanpassning (genom att själva ha rätt att ändra i mjukvaran).

En faktor som är genomgående för de studerade företagen är att de, till skillnad från de internationella exempel som nämnts tidigare, i lägre grad fokuserar på enskilda produkter. Deras arbete är snarare inriktat på traditionell IT-konsultverksamhet där de arbetar med ett fokus på att lösa specifika problem med hjälp av Open Source-programvara. Detta må vara att passa ihop drivrutiner med viss hårdvara, såväl som att sätta upp e-postservrar eller sätta samman programvara för ett intranät. Istället för att förlita sig på en specifik produkt använder de olika mindre OS-produkter och kombinerar dessa efter behov.

6.3 Företagens koppling till Open Source-rörelsen

Respondenterna har i flera fall under intervjuerna pekat på deras företags ”bakgrund”, det vill säga deras historiska koppling till Open Source-rörelsen. Mitt intryck är att man i flera fall har ett behov av att inte hamna i dålig dager inom Open Source-rörelsen genom att driva företaget utan att bidra tillbaka till Open Source-rörelsen.

Min förklaring till detta är att man helt enkelt anser sig vara beroende av de utvecklare som runt om världen utvecklar fri programvara. Jag tror även att man, i viss mån, känner en förpliktelse att bidra till Open Source-utvecklingen för att stärka den samma. Flera av de intervjuade har hävdat att det råder ett slags Moment 22-tillstånd för OSS: utan företag som för fram OSS kan inte OSS växa och få fler användare och utvecklare. På samma sätt får företagen svårare att lyckas utan en stark Open Source-rörelse som kan hjälpa till att skapa bra program.

Open Source-företagen verkar ofta uppstå ur Open Source-rörelsen på ett eller annat sätt. Antingen startas de av personer direkt engagerade i Open Source-projekt eller av personer som kommit i kontakt med tankarna kring Open Source under sin tid på högskola och universitet.

6.4 Slutord

Som vidare forskning skulle det vara intressant att närmare studera den affärsidé jag funnit hos ett av företagen, Codefactory AB, som bygger på att sprida den utvecklingsmetod som används i många Open Source-projekt. Jag har inte, i litteratur, funnit några tidigare studier av detta och tanken att arbeta med att sprida själva utvecklingssättet är i mitt tycke intressant. Aspekter som vore särskilt intressanta att belysa vore vilken genomslagskraft utvecklingssättet får och även att undersöka hur detta eventuellt bidrar till att stärka Open Source-rörelsen.

Kapitel 7 – Slutsats

Affärsidéerna som de intervjuade beskrivit överensstämmer i stort med de som finns beskrivna i litteraturen för utländska Open Source-företag. I stort sett faller samtliga dessa företag inom ramen för ”Kringtjänster, service och support” och respondenterna har menat att företagen ersatt inkomster från egen proprietär programvara med inkomster just från detta område. En klar skillnad i arbetssätt jämnfört med till exempel RedHat, som nämns som exempel i litteraturen, är att de studerade företagen har mindre fokus på enskilda produkter och snarare kan betraktas som IT-konsulter som utifrån ett kundperspektiv försöker löser avgränsade problem med hjälp av ett flertal olika Open Source-programvaror snarare än med hjälp av en produkt.

Förutom detta har jag dock funnit ytterligare en affärsidé. Denna bygger på utbildning och konstruktion av infrastrukturella förutsättningar för utvecklingsarbete baserat på Open Source-liknande metoder. Här är strategin att genom utbildning och anpassning av mjukvara sälja nyttan i att organisera projektarbete i kundföretagen på ett sådant sätt som ofta görs inom Open Source-projekt.

Företagen uppvisar i samtliga fall en vilja att utgöra en del av Open Source-rörelsen, snarare än att ensidigt utnyttja dess framsteg. Detta sker genom att man på olika sätt bidrar till befintliga projekt eller startar och driver egna.

Om man skulle summera de konkurrensfördelar företagen ser med att använda Open Source i verksamheten kan dessa sammanfattas med:
Öppenhet gentemot kunder genom att källkoden är fri att ändra och vidareutveckla.
Lägre TCO och initiala kostnader i form av programvarukostnader.
Bättre kontinuitet genom att programvaran utvecklas av flera oberoende aktörer och sålunda kan leva vidare oavsett utvecklarbasens sammansättning.
Inkrementell utveckling, vilket innebär att man kan utveckla nya produkter med gamla som grund.
Större utvecklarbas genom att utvecklarbasen är oberoende av det egna företaget.

Referenser

Böcker

Backman, Jarl. (1998). *Rapporter och uppsatser.* Lund: Studentlitteratur.

Brooks, Frederik P., Jr. (1995). *The Mythical Man-Month.* Addison Wesley Longman Inc.

DiBona, Chris, Ockman, Sam, & Stone, Mark. (1999). *Open Sources. Voices from the Open Source Revolution.* Sebastopol, USA: O'Reilly & Associates.

Eriksson, Lars Torsten, & Wiedersheim-Paul, Finn. (1997). *Att utreda forska och rapportera.* Malmö: Liber Ekonomi.

Feller, Joseph, & Fitzgerald, Brian. (2002).Understanding Open Source Software Development. Harlow, England: Pearson Education Limited.

Holme, Idar Magne, & Solvang, Bernt Krohn. (1997). Forskningsmetodik. Lund: Studentlitteratur.

Moody, Glyn. (2001). Rebel Code. Linux and the Open Source revolution. London: Allen Lane, The Penguin Press.

Raymond, Eric S. (2001). *Katedralen och basaren.* Nora: Nya Doxa.

Repstad, Pål. (1993). *Närhet och distans.* Lund: Studentlitteratur.

Torvalds, Linus, & Diamond, David. (2001). *Just for fun. Mannen bakom Linux.* Stockholm: Alfabeta.

Wayner, Peter. (2000). *Free for All.* New York: HarperCollins Publishers Inc.

Young, Robert, & Goldman Rohm, Wendy. (1999). *Under the radar.* Scottsdale, Arizona: The Corolis Group.

Artiklar och avhandlingar

Andersson, Mattias. (2000). *Open Source – Ur ett praktiskt juridiskt perspektiv.* Juridiska institutionen vid Handelshögskolan, Göteborg universitet.

Barbrook, Richard. (1998). *The Hi-Tech Gift Economy.* First Monday, vol 3, nr 12. <http://www.firstmonday.dk/issues/issue3_12/barbrook>.

Bergquist, Magnus & Ljungberg, Jan. (2001). *The power of gifts: organizing social relationships in open source communities.* Blackwell Science Ltd, Information Systems Journal 11, sid 305-320.

Computer Sweden (2003). *Microsoft känner sig hotat av öppna system.* 2003-02-05.

Cook, Jonathan E. (2001). *Distributed Knowledge and the Global Organization of Software Development.* <http://opensource.ucc.ie/icse2001/cook.pdf>.

Dalle, Jean-Michael &, Jullien, Nicolas. (2002). *Open-Source vs. Proprietary Software.* Free/Open Source Research Community (Online Papers). <http://opensource.mit.edu/online_papers.php>. 2002-05-16.

Dempsey, Bert J., Greenberg, Jane, Weiss, Debra, & Jones, Paul. (1999). *A Quantitative Profile of a Community of Open Source Linux Developers.* Chapel Hill, North Carolina: UNC Open Source Research Team. <http://www.ibiblio.org/osrt/develpro.html>.

Edwards, Kasper. (2001). *Epistemic Communities, Situated Learning and Open Source Software Development* (Working paper). Free/Open Source Research Community (Online Papers). <http://opensource.mit.edu/online_papers.php>. 2002-05-16.

Feller, J., & Fitzgerald, B. (2000). *A Framework Analysis of the Open Source Software Development Paradigm*. I W. Orlikowski, P. Weill, S. Ang & H. Krcmar (red). Proceedings of the 21st Annual International Conference on Information Systems, Brisbane, Australia, December 2000.

Forge, Simon (2000). *Open Source: the economics of giving away stuff, and software as a political statement.* Camford info - the journal of policy, regulation and strategy for telecommunications information and media, Vol 2, No 1, 2000.

Harhoff, Dietmar, Henkel, Joachim, & Eric von Hippel. (2000). *Profiting from voluntary information spillovers: How users benefit by freely revealing their innovations.* Working Paper #4125, MIT Sloan School of Management.

Hubley, Mary. (2001). *The Pros and Cons of Open-Source Software and Linux*. Gartner Group Research, COM-13-5574.

Jørgensen, Niels. (2001). *Putting it all in the trunk: incremental software development in the FreeBSD open source project.* Information Systems Journal, 11, 321-336.

Lakhani, Karim, & von Hippel, Eric. (2000). *How Open Source software works: "Free" user-to-user assistance.* MIT Sloan School of Management Working Paper #4117.

Learner, Josh, & Tirole, Jean. (2000). *The Simple Economics of Open Source.* National Bureu of Economic Research. <http://papers.nber.org/papers/w7600>.

Learner, Josh, & Tirole, Jean. (2001). *The open source movement: Key research questions.* European Economic Review 45, 2001, s 819-826. Elsevier Science B.V.

Mantarov, Bojidar. (1998). *Open Source Software as a new business model.* University of Reading. <http://bmantarov.free.fr/academic/msc.htm>.

Moglen, Eben. (2000). *Free Software Matters: Free Software or Open Source?* Publicerad i LinuxUser. <http://emoglen.law.columbia.edu/publications/lu-07.pdf>.

Osterhout, John. (1999). *Free Software Needs Profit.* Communications of ACM, april 1999/vol 42, no 4.

Pawlo, Mikael. (2002). *Något om fri programvara och öppen källkod – nya licenstyper för datorprogram.* <http://harvard.pawlo.com/pawlo-nir.pdf>.

Raymond, Eric. (1999). *A Response to Nikolai Bezroukov.* First Monday, vol 4, nr 11. <http://firstmonday.org/issues/issue4_11/raymond/index.html>. 2002-05-16

Tuomi, Ilkka. (2001). *Internet, Innovation, and Open Source: Actors in the Network.* First Monday, vol 6, nr 1. <http://firstmonday.org/issues/issue6_1/tuomi/index.html>.

Internet

Affärsdata (2003). Företagsuppgifter avseende bokslutsperioden 200101-200112. <http://www.ad.se>. 2003-02-25.

BSA Sweden. (2002). *BSA Sweden - Antipirat-verksamhet.* <http://www.bsa.org/sweden/antipiracy/>. 2002-08-28.

Cendio Systems. (2002a). *Cendio Systems - Hem.* <http://www.cendio.se/>. 2002-03-05.

Cendio Systems. (2002b). *Cendio Systems - om Cendio.* <http://www.cendio.se/page.about>. 2002-03-05.

CodeFactory. (2002). *CodeFactory::Welcome to CodeFactory.* <http://www.codefactory.se/>. 2002-03-05.

Cohen, Josh, & Valloppillil, Vinod (1998). *The Halloween Documents.* Microsoft Corporation, confidential memorandum. <http://www.opensource.org/halloween/>. 2002-01-14.

Gates, Bill. (2002). Government Leaders' Conference. Remarks by Bill Gates. < http://www.microsoft.com/billgates/speeches/2002/04-17glc.asp>. 2002-04-21.

Gnuheter (2001). *Går det att tjäna pengar på Open Source?* <http://www.gnuheter.com/article.php?sid=542>. 2003-02-24.

History of the OSI. <http://www.opensource.org/docs/history.html>. 2001-01-09.

Jones, Paul. (2000). *Brooks' Law and open source: The more the merrier?* <http://www-106.ibm.com/developerworks/library/merrier.html>. 2002-04-29.

MIT. (2003). *MIT Process Handbook.* MIT Sloan School of Management. <http://process.mit.edu/eph/Info/eModels.asp>. 2003-03-31.

Netcraft Web Server Survey. <http://www.netcraft.com/survey/>. 2002-01-09.

Newbart, Dave. (2001). *Microsoft CEO takes launch break with the Sun-Times.* <http://www.suntimes.com/output/tech/cst-fin-micro01.html>. 2002-04-21.

News.com (2001). *Governments push open-source software.* <http://news.com.com/2100-1001-272299.html>. 2003-02-25.

Open Source Forum. (2002). *The Scandinavian Open Source knowledge base.* <http://www.opensource-forum.com/>. 2002-03-05.

Open Source History Interactive Timeline. <http://www.osdn.com/timeline/>. 2002-01-09.

Shankland, Stephen. (2001). *Linux growth underscores threat to Microsoft.* <http://news.com.com/2100-1001-253320.html>. 2002-05-06.

Stallman, Richard. (1993). *The Gnu Manifesto.* <http://www.gnu.org/gnu/manifesto.html>. 2002-05-06.

Raditex. (2002a). *Raditex AB.* <http://www.raditex.se/>. 2002-03-05.

Raditex. (2002b). *Företaget Raditex.* <http://www.raditex.se/raditex.html>. 2002-03-05.

Raymond, Eric. (2002). *The Jargon File, version 4.3.1.* <http://www.tuxedo.org/~esr/jargon/html/>. 2002-04-21.

Regan, Dave. (1999). *The Gospel of Tux (v1.0).* <http://www.ao.com/~regan/penguins/tux.html>. 2002-04-28.

SorceForge:Software map.
<http://sourceforge.net/softwaremap/trove_list.php?form_cat=14>. 2002-04-22.

South Pole. (2002). *South Pole AB - Leverantör av konsulttjänster och produkter baserade på Linux.*
<http://www.southpole.se/>. 2002-03-05.

Stallman, Richard (2001). *The GNU Project.* <http://www.fsf.org/gnu/the-gnu-project.html>. 2002-01-09.

The Open Source Definition, Version 1.9.
<http://www.opensource.org/docs/definition.html>. 2002-01-10.

OSI. (2002). *The Open Source Initiative: Licenses.*
<http://www.opensource.org/licenses/index.html>. 2002-04-22.

OSI. (2002b). *The Open Source Initiativ: Products.*
<http://www.opensource.org/docs/products.html>. 2002-04-24.

OSI. (2002c). *The Open Source Initiativ: The GNU General Public License (GPL).*
<http://www.opensource.org/licenses/gpl-license.html>. 2002-04-24.

OSI. (2002d). *The Open Source Initiative: History of the OSI.*
<http://www.opensource.org/docs/history.html>. 2002-05-06.

Appendix A – Intervjumanual

Bakgrund om företaget
När startades företaget?
Varför startades företaget?
Vilken var affärsidén från grunden?
Har affärsidén förändrats?
Hur ser affärsidén ut idag?

Vad är Open Source för er?
Vad betyder Open Source för företaget? Beskriv Open Source…

Varför Open Source?
Varför har företaget valt Open Source som en stor del av verksamheten?
Påverkar valet av Open Source verksamheten?
Tycker ni att Open Source-inriktningen gör att ni skiljer er från andra företag?

Marknadsaspekter
Hur ser ni på programvara som vinstgenerator?
Om programvaran är gratis vad inriktar ni er på då.
Ger valet av Open Source konsekvenser för företagets marknadsföring och dess sätt att tjäna pengar?

Appendix B – Ordlista

Apache: webbserverprogramvara distribuerad under Open Source-licens. http://httpd.apache.org/

BSD: Berkeley Software Distribution. En Unix-variant som utvecklades vid University of California/Berkeley och sprids under den såkallade BSD-licensen. Idag finns ett antal olika BSD-system, bland annat NetBSD och FreeBSD. http://www.bsd.org/

Copyleft: En ordlek med det engelska begreppet copyright. Syftar på en omvänd upphovsrätt som snarare skyddar användarens frihet än upphovsmannens. Ett exempel på program-varulicenser som kan sägas falla under copyleft är GPL.

FSF: Free Software Foundation. Stiftelse, grundad av Richard M. Stallman 1984, som bland annat driver GNU-projektet. http://www.fsf.org/

GNU: Står för ”GNU's Not Unix” (GNU är inte UNIX). Ett projekt styrt av FSF med målet att skapa ett Unix-liknande operativsystem som helt bygger på fri programvara. Startades 1984 av Richard M. Stallman. http://www.gnu.org/

GPL: GNU Public License. Den mest spridda licensen som är godkänd som en Open Source-licens av OSI.

Kompilering: omvandling av källkod till maskinkod, det vill säga ett program. Kompileringen utförs med hjälp av en kompilator, till exempel Gnu C Compiler, GCC.

Källkod: okompilerad programkod. Se kompilering.

Linux: operativsystem distribuerat under Open Source-licens. Utvecklades av Linus Torvalds.

OSD: The Open Source Definition. Definition utarbetad av OSI som används för att avgöra vad som är Open Source. http://opensource.org/docs/definition.php

OSI: Open Source Initiativ. Icke vinstdrivande företag som arbetar för att sprida bland annat OSD. Godkänner även Open Source-licenser. http://www.opensource.org/

OSS: Open Source Software. Programvara som distribueras i enlighet med någon av OSI-licenserna.

Tux: Operativsystemet Linux officiella maskot, en pingvin som enligt Torvalds är ”kramgo”. Tux utseende är, enligt Linus Torvalds beskrivning, typiskt för det stadie i vilket pingviner befinner sig när de är proppmätta av sill eller just haft sex.

Öppen källkod: Svensk översättning av begreppet Open Source.

Appendix C – Index