LUNDS UNIVERSITET
Ekonomihögskolan

Motivation i open source

Vilka faktorer ligger bakom utvecklares medverkan i open source?

Kandidatuppsats, 15 högskolepoäng, SYSK02 i informatik

Framlagd *2013-05-31*

Författare Erik Kronberg
Stina Qvarnström
Björn Svensson

Handledare Lars Fernebro

Examinator Markus Lahtinen
Paul Pierce

Titel:	Motivation i open source – Vilka faktorer ligger bakom utvecklares medverkan i open source
Författare:	Erik Kronberg
	Stina Qvarnström
	Björn Svensson
Utgivare:	Institutionen för informatik
Handledare:	Lars Fernebro
Examinator:	Markus Lahtinen
	Paul Pierce
Publiceringsår:	2013
Uppsatstyp:	Kandidatuppsats
Språk:	Svenska
Nyckelord:	Open source, motivation

Abstrakt:

På bara några få år har open source-mjukvaror som Mozilla Firefox tagit stora marknadsandelar från sina slutet utvecklade konkurrenter. Open source är ett snabbt växande och komplext socio-ekonomiskt fenomen som är beroende av en uppsjö av faktorer från en brokig skara ämnesområden. I denna uppsats kartlägger författarna vilka motivationsfaktorer som existerar hos medverkare i öppenkällkodsprojekt samt deras uppfattade relevans. Utifrån tidigare forskning och svarsdata från enkät sammanställdes ett teoretiskt ramverk som låg till grund för en ingående studie och rangordning av de innefattade motivationsfaktorererna i open source-gemenskapen samt på arbetsplatsen. Insamling av empirisk data gjordes på de virtuella samhällena Reddit och Hacker News. Den första enkäten var av en kvalitativ art och låg, tillsammans med tidigare forskning, till grund för det teoretiska ramverket samt den andra enkäten där respondenterna ombads rangordna faktorerna på en skala.
Författarna visar en tydlig skillnad mellan hur individer verksamma i branschen motiveras på arbetsplats och i open source. Ideologiska faktorer identifieras som de mest polariserande och engagerande motivationsverktygen, men över lag visas open source som en mer motivationsrik grogrund för innovation och utveckling. Betalning kvarstår som den viktigaste motivationsfaktorn på arbetsplatsen.

Innehållsförteckning

Figurförteckning

Tabellförteckning

1 Bakgrund

På de flesta företag skapas ett visst sätt att göra saker på. Hur man interagerar med varandra, mot kunder, vilka attityder som finns och hur arbete utförs, ibland även hur man tänker och känner inom en organisation. Är det en stor organisation så kan det skapas olika typer av kulturer på olika ställen i organisationen. Hur kulturen är på ett företag kan ha stor inverkan på hur framgångsrik verksamheten blir. Skillnaden mellan ett företag som presterar normala ekonomiska resultat och de som ligger i framkant kan vara hur kulturen ser ut internt (Barney, 1986). Att skapa en framgångsrik kultur är ingenting man gör i en handvändning och det finns inga enkla genvägar, men en viktig del i kulturen är att de anställda är motiverade och engagerade i hur det går för verksamheten. Två av de aspekter som den mycket välciterade undersökningen *In Search of Excellence* visar på är att de företag som är framgångsrika har är att de fostrar innovation och självstyre bland de anställda, samt att produktivitet kommer genom folket. De anställda ska behandlas som en källa till kvalitet (Waterman & Peters, 1982). Vi kan med andra ord konstatera att kulturen är viktig för hur en verksamhet presterar och att de anställda är en del i kulturen. Att de anställda trivs och är villiga att göra sitt arbete är betydelsefullt och kan influera hur framgångsrik organisationen är.

Det är ingen hemlighet att många IT-projekt misslyckas. Det finns mängder med exempel på projekt som inte blir inte färdiga i tid, innehåller mer eller mindre allvarliga fel och kostar ofta mer än vad de ursprungligen beräknades till. Miljoner läggs ut på utveckling av system som i värsta fall inte ens går att använda, eller skapar stora ekonomiska problem. Under början av 2000-talet försökte FBI skapa ett virtuellt filsystem kallat VCF. Runt 170 miljoner dollar av skattepengar lades under 5 år på utveckling i ett försök att lösa de gigantiska problem de hade med omodern hantering av data. Projektet övergavs officiellt år 2005 (Goldstein, 2005). På London Stock Exchange spenderade man minst 75 miljoner pund på ett system som fick överges och helt ersättas av ett annat under början av 90-talet (PLC Magazine, 1993). Under 2011 fick japanska Honda återkalla 2.49 miljoner bilar efter att en mjukvarubugg hittats vilken riskerade att skada växellådan (Woodall & Seetharaman, 2011). Även om problemen minskat allt eftersom så finns det fortfarande utrymme för förbättring i hur mjukvara utvecklas.

Mjukvara som utvecklas med öppen källkod är en växande och betydelsefull bransch idag. Tidningen Wall Street Journal rapporterade i september av 2012 att över 50 % av mjukvarubehovet över de kommande fem åren kommer att täckas av mjukvara som utvecklats med öppen källkod (King, 2012). Användandet av open source har spridit sig till mer än bara teknik- och it-företag; jättar som Sears, Chevron och MasterCard utnyttjar öppen källkod. När Chevron letar efter gas och olja i Mexikanska Gulfen analyserar de datan med hjälp av

databasramverket Hadoop, som är open source. Apache Web Server är den mest använda HTTP-servern i världen (Netcraft, 2012), Mozilla Firefox har i dagsläget cirka 20 % av marknaden för internetbrowsers (W3Counter, 2013), båda är utvecklade med öppen källkod. Open source-mjukvara finns nu utspridd över stora delar av världen och projektformen kan anses varit ett framgångsrikt sätt att utveckla på. I artikeln *Estimating the Total Development Cost of a Linux Distribution* gjorde McPherson et al. (2008) en uppskattning på hur mycket Linux skulle kostat om det var utvecklat på traditionellt sätt och kom fram till en uppskattad siffra på 1.4 miljarder dollar för kärnan. Men istället för att de summorna lagts ner av The Linux Foundation har utvecklare från andra håll bidragit med kod utan att få betalt. Open source-mjukvara utvecklas till stor del av utvecklare som inte får betalt för det de gör. Ofta sker arbetet på deras fritid. Till Apache har tusentals individer bidragit med rapporter om fel i mjukvaran och hundratals har bidragit med kod (Mockus et al., 2002). På senare år har även möjligheten att som utvecklare själv lägga ut sin kod öppen ökat genom tjänster som exempelvis GitHub, en webbsida där användare kan publicera sin kod och få kommentarer och andra typer av feedback från andra. GitHub har i dagsläget cirka 6 miljoner så kallade repositories, kodarkiv, och 3 miljoner användare (GitHub, 2013). Stora mängder kod produceras av många individer, det finns mycket innovation och gemenskapen växer.

I open source-projekten finns det med andra ord mängder av utvecklare som till synes är motiverade av annat än pengar. De bidrar frivilligt och är engagerade utan tvång. De skapar trots det kod av hög kvalitet, kanske till och med högre än kommersiella projekt (Black Duck Software & North Bridge Venture Partners, 2013; Coverity, 2012).

1.2 Problemformulering, syfte och avgränsning

Kulturen i en organisation påverkar dess prestation och mjukvarubranschen har små marginaler och stor konkurrens. Det är viktigt att ligga i framkant för att lyckas. Därför är kulturen och hur ens anställda presterar viktigt. Med rätt kultur kan man öka produktiviteten. Inom open source produceras framgångsrik mjukvara som sprider sig allt mer över världen. Orsaken till detta kan ligga i ett antal faktorer som projektform, utvecklingsmetodik och typen av personer som medverkar. Den aspekt vi valt att studera är motivation. Vi tror att det kan finnas något att lära sig av att titta på vad som motiverar utvecklare till att lägga sin fritid på open source-projekt. Genom att förstå motivationsfaktorer tror vi att man eventuellt kan förbättra sin egen mjukvaruutveckling när man förändrar processerna och arbetsplatsen så att utvecklarna blir mer motiverade och produktiva. I slutändan är det individer som skapar kod och de gör ett bättre jobb om de trivs med vad de gör.

Vår forskningsfråga lyder som följer:

Vad motiverar individer till att bidra till open source?

Självklart är det inte bara kulturen och de anställda som påverkar en verksamhet, många andra aspekter spelar in, men det är en faktor som är värd att studeras. Det är inte heller endast motivation som utgör kulturen, men vi har valt att avgränsa oss och bara fokusera på det i den här studien.

Det finns befintlig forskning inom det här området, men mycket av den har några år på nacken och därför tror vi att en ny studie skulle kunna tillföra information om huruvida det ser likadant ut nu, eller om faktorer tillkommit eller försvunnit.

Det finns inte heller en enskild motivationsfaktor för varje individ, utan antagligen motiveras en individ av flera faktorer. Därför tycker vi att det hade tillfört till området att rangordna motivationsfaktorerna, vilket inte verkar ha gjorts i stor utsträckning i tidigare forskning. Vi kommer även att koppla motivationsfaktorerna till vad som motiverar en individ på en arbetsplats och jämföra de båda för att se om det finns skillnader och lärdomar att dra.

Open sourcevärlden är en viktig del av nutidens IT-bransch och vi tror att det finns läxor att lära från den. Med den här uppsatsen hoppas vi att kunna kartlägga en aspekt av det mångfacetterade open source-fenomenet, nämligen motivationsfaktorerna hos individerna som ligger bakom det. Genom att sprida ljus över denna komplexa gemenskap hoppas vi kunna identifiera och jämföra motivationsfaktorerna både inom och utanför området.

1.3 Definitioner

I den här uppsatsen har vi valt att använda Open Source Initiative för att definiera vad open source är. De har publicerat en omfattande beskrivning som vi sammanfattar nedan (Open Source Initiative, 2013):

1 Mjukvaran ska vara fri att sprida

2 Källkod måste finnas tillgänglig

3 Måste tillåta förändringar

Vi kommer att använda översättningen “öppen källkod” för att beskriva open source. Vi har valt att använda begreppet open source i en bredare bemärkelse för att bland annat beskriva gemenskapen och projekten som finns runt koden. I den här uppsatsen betyder open source med

andra ord inte bara kod, utan även en rörelse, vilket stämmer väl överens med hur begreppets betydelse används i verkligheten och låter situationen och kontexten förklara vad vi menar, och på de ställen där det är oklart förklarar vi. Däremot inkluderar vi endast projekt som handlar om mjukvaruutveckling.

Vi har även valt att ibland använda det engelska ordet *community* för att beskriva de virtuella samhällen och gemenskapen som finns kring den öppna källkoden.

2. Litteraturgenomgång

2.1 Introduktion

I vår litteraturgenomgång kommer vi att presentera teorier inom motivationsforskning och hur de kan relateras till mjukvaruutveckling inom open source-miljön. Vi kommer att presentera ett flertal studier som har forskat i vad som motiverar individer att bidra till open source i olika kontext. Studierna kommer att granskas vad gäller deras metod, målgrupp, omfattning och resultat. Efter detta kommer vi att kategorisera studiernas motivationsfaktorer. Kategoriernas syfte är att gruppera likartade motivationsfaktorer, till exempel skulle de tänkta motivationsfaktorerna "intresse för programmering" och "intresse för Linux specifikt" kunna grupperas i en kategori som skulle heta "intresse". Efter kategoriseringen kommer det att finnas ett antal generella kategorier av motivationsfaktorer relaterade till mjukvaruutveckling inom open source. Dessa kategorier kommer att utgöra det teoretiska ramverk som vi kommer att använda för vår undersökning.

För att få ökad förståelse för dessa kategorier kommer vi att presentera motivationsforskning relaterad till varje kategori. Anledningen till detta är att få en djupare förståelse av kategorin för att sedan kunna använda denna i vår egen undersökning. Med hjälp av tidigare forskning kan vi förstå inte bara att ett svar bör falla under en viss kategori men också varför och vad som kan ligga bakom svaret.

Motivationsforskningen som vi kommer att presentera består av kända och välciterade teorier. Teorierna är kända men också från en annan tidsperiod. Vissa av teorierna kan gå tillbaka till en tid då fenomenet, mjukvaruutveckling inom open source, inte fanns. Teorierna är generella och deras applicering på vårt område, mjukvaruutveckling i open source-miljön, kommer att diskuteras med hjälp av nyare forskning relaterad till området.

Samtliga studier som används för att fastställa kategorierna av motivationsfaktorer har publikationsdatum från 2002 till 2010. Även den forskning som används för att diskutera studierna och att sätta motivationsforskningen i relation till open source-miljön är relativt nya (inte äldre än år 2000). Detta innebär att kapitel två blandar äldre, men välciterade källor för motivationsforskning, med nyare och mer aktuella källor för att beskriva problemområdet och hur den äldre forskningen kan appliceras.

2.2 Tidigare studier inom området

Eftersom att denna uppsats syftar till att undersöka förhållanden i den riktiga världen kräver detta ett befintligt teoretiskt ramverk. Detta teoretiska ramverk existerar inte sedan tidigare, utan kommer att konstrueras i detta kapitel i de nästkommande textstyckena. Ett teoretiskt ramverk behöver en grund att stå på och vi grundar vårt teoretiska ramverk på undersökningarna i tre studier som undersökt motivation relaterat till deltagande i open source. Det första kriteriet för urval var att studien på något sätt skulle behandla deltagande i open source; undersökningar i frivilligarbete inom andra områden är inte av intresse för denna studie. Studierna som valts ut har valts ut baserat på det område inom open source som de studerar, samt tidpunkten för studien som en indikation på dess relevans till dagsläget. Ett sista kriterium var svarsmängd och bortfall.

2.2.1 Benbya & Belbaly – Understanding Developers' Motives in Open Source Projects

Benbya & Belbaly (2010) studerade medverkande i open source-projekt hostade på internetsidan sourceforge.net, mer specifikt under kategorin "Software development frameworks and tools for enterprise application development". Författarna valde slumpmässigt ut projekt som uppvisade en viss nivå av aktivitet: att det hade skett arbete på projektet inom en viss tid. Inom de utvalda projekten tillfrågades 320 individer via internet-enkät varav 122 svarade på förfrågan, detta ger studien ett bortfall på 61,88 %. Författarna skapade ett multiteoretiskt ramverk byggt på tre teorier: målinriktning, förväntningar och socialt utbyte. Utifrån det valde de att undersöka sex potentiella incitament till varför utvecklare bidrar till open source-projekt:

- Ideologi, att individers åsikter och tankar om open source-mjukvara kan motivera dem att bidra till gemenskapen.
- Reciprocitet, att ge tillbaka till gemenskapen. Faktorn baseras på att individer som gynnats av andras arbete har en tendens att återgälda tjänsten (Grouldner, 1960 enligt Benbya & Belbaly, 2010).
- Förväntningar, en faktor som baseras på att individer handlar om de kan förvänta sig att deras handling leder till saker de vill ha (Vroom, 1964 enligt Benbya & Belbaly 2010).
- Valens, denna faktor beror på relevansen och betydelsen av de fördelar som individen kan förvänta sig.
- Resultatinriktning (performance), innebär att individer motiveras av att jämföra deras prestationer med andras. Individens status i relation till sin omgivning kan öka om han eller hon, på ett synligt sätt, presterar bättre än andra (Covington, 2000 enligt Benbya & Belbaly, 2010).
- Inlärningsinriktning (learning), syftar på incitamentet för individen att utveckla och förbättra sina kunskaper.

Diskussion av Benbya & Belbaly

Benbya & Belbaly (2010) studerar endast ett område inom open source, nämligen utvecklingen av enterprise-applikationer och ramverk. Detta innebär att studien kan misslyckas med att representera open source-gemenskapen som en helhet, författarna påpekar dock att området de valt utgör en stor andel av projekten på sourceforge.net. För övrigt har användningen av dessa applikationer ökat de senaste åren (IDC, 2009 enligt Benbya & Belbaly, 2010). Datainsamlingen för studien är relativt färsk jämfört med till exempel Hertel et al. (2003), Lakhani & von Hippel (2003) och Hars & Ou (2002). Bortfallet på 61,88 % är betydligt, men mindre än bortfallen i andra studier som vi uppsatsförfattare granskat, till exempel Lakhani & von Hippel (2003).

2.2.2 Baytiyeh & Pfaffman – Open Source Software: A community of altruists

Baytiyeh & Pfaffman (2010) inriktade sig på åtta olika grupper som bidragit till open source-projekt, bland annat Mozillautvecklare, utvecklare för OpenOffice och Moodle. Datainsamlingen skedde under året 2008. Undersökningen fick 110 svar varav majoriteten var från Moodle. Undersökningsmedlet, som i flera andra studier, var en internet-baserad enkät som respondenterna fick fylla i. Författarnas studie identifierade följande motivationsfaktorer:

- Altruism (altruism motivators) innebär att en individ agerar utan löfte om konkreta belöningar för sitt handlande. Istället får individen psykologiska sekundärbelöningar när de till exempel hjälper en annan individ (Ross-Ackerman, 1998 enligt Baytiyeh & Pfaffman, 2010). Altruism kopplas direkt till individens identifikation med gemenskapen, som en belöning för det altruistiska beteendet (Baytiyeh & Pfaffman, 2010).
- Skapandet av en publik artefakt (motivation to create) kan vara motiverande i sig självt, särskilt processen att bygga någonting från “scratch” och se artefakten slutföras (Lindenberg, 2001, enligt Baytiyeh & Pfaffman 2010). Detta kallas för konstruktionism, och härstammar från ett djupgående behov att skapa och innovera (Harel & Papert, 1991 enligt Baytiyeh & Pfaffman, 2010).
- Inlärning (motivation to learn) är också en faktor. Författarna gör antagandet att deltagare i open source-projekt har ett begär av kunskap och att lära sig (Dewey, 1915 enligt Baytiyeh & Pfaffman, 2010). Författarna menar även att individer vill involvera sig i processen att bidra till open source-projekt för att de ska dra lärdomar av de metoder som bidragandet utformas av.
- Programmering för nöjes skull (flow), som de benämner ”flow”. Uttrycket flow syftar på ett tillstånd som en individ kan befinna sig i (Nakamura & Csikzenthihalyi, 2003 enligt Baytiyeh & Pfaffman, 2010). Tillståndet uppstår när en individs färdigheter sätts på prov

eller när de är extremt koncentrerade och involverade i en uppgift (Baytiyeh & Pfaffman, 2010).

- Yttre faktorer (extrinsic motivators) exemplifieras som aktiviteter som arbete för att utveckla verktyg till att tillgodose sina egna behov eller för att stödja sitt arbete. En individs status i open source-gemenskapen kan även vara en yttre faktor, rykte och berömmelse är således också yttre faktorer.

Diskussion av Baytiyeh & Pfaffman (2010)

Studien av Baytiyeh & Pfaffman (2010) är relativt färsk till skillnad från studier som Hertel et al. (2003), Lakhani & von Hippel (2003), Hars & Ou, (2002) och har en vitt spridd målgrupp. Författarna anger dock ingen grad av bortfall för deras studie. Siffran 110 anges som antalet svar som inhämtades, inget nämns dock om antal respondenter tillfrågade eller antalet potentiella respondenter. Följden av detta blir att studiens förmåga att representera målgruppens åsikter och värderingar kan ifrågasättas.

2.2.4 Hars & Ou - Working for Free? Motivations for Participating in Open-Source Projects

Hars & Ou (2002) tillfrågade 389 personer som enligt författarna var involverade i open source projekt. Utav dessa svarade 81 personer (21 %) på undersökningen. Undersökningen skedde via en e-postlista som konstruerades av e-postadresser hämtade från olika open source-diskussioner på internet. Undersökningen kom fram till att relevanta motivationsfaktorer var

- Altruism (altruism) förklaras av Hars & Ou (2002) som en sorts inre motivation. Altruism är motsatsen mot självviskhet, det innebär att individen gör något för andras nytta på egen bekostnad.
- Att identifiera sig med gemenskapen (community identification) klassificeras som en form av altruism. Den är kopplad till grundläggande mänskliga behov av tillhörighet (Maslow, 1987 enligt Hars & Ou, 2002).
- Försäljning av produkter (selling products) innebär att en individ kan gynnas genom att sälja produkter och tjänster relaterade till sitt bidrag.
- Humankapital (human capital) syftar på en individs färdigheter och kunskaper. Eftersom att förbättring av humankapitalet kan leda till bättre framtida karriärsutsikter finns det ett motiv för individer att förbättra sitt humankapital (Becker, 1962 enligt Hars & Ou 2002).
- Marknadsföring av sig själv (self marketing) syftar till individens sätt att demonstrera sina färdigheter och kunskaper utåt.

- Erkännande från omgivningen (peer recognition) kan fungera som en motiverande faktor för en individ. I open source-gemenskapen tyder feedback på att utvecklarens bidrag faktiskt används, vilket ger utvecklaren ytterligare anledning att vidareutveckla sitt bidrag i förhoppning om mer positiv feedback.
- Personliga behov (personal needs) syftar på situationen då en utvecklare utvecklar mjukvara för att han eller hon behöver det för att understödja sitt egna dagliga arbete.
- Inneboende motivation (self determination) beskrivs som individers naturliga begär att bete sig på vissa sätt, till exempel begäret hos individen att skapa en relation till sin omgivning och att bekräfta sin kompetens (Deci, 1975 enligt Hars & Ou, 2002).

Diskussion av Hars & Ou

Studien saknar angiven tidsperiod för datainsamlingen, det kan alltså inte urskiljas om denna började till exempel 1995 eller år 2000 vilket kan ha inverkan på resultatet om åsikter och motiv såg annorlunda ut vid olika tidpunkter. Studiens bortfall på 79 % kan ses som problematiskt för resultatet eftersom att det bygger på svaren från endast en femtedel av den tillfrågade mängden. Studien begränsar sig dock inte till en liten del av open source-gemenskapen som till exempel är fallet med Lakhani & von Hippel (2003) och Hertel et al. (2003) vilket gör den lämplig för ett teoretiskt ramverk som inte ska inrikta sig specifikt.

2.3 Utformning av vårt teoretiska ramverk

Efter en studie av den tidigare forskningen kom vi fram till att det inte fanns en tydlig konsensus på hur man benämnde och delade in motivationsfaktorerna. Därför valde vi att konstruera ett eget teoretiskt ramverk som byggde på och sammanfattade de faktorer som de ovan nämnda studierna visade på. Vi kommer nedan att redovisa för vårt ramverk och hur vi valt att översätta, där det behövs, föregående motivationsfaktorer till våra egna. Vissa av studierna har faktorer som överlappar varandra och andra fattas helt faktorer som andra studier nämnt.

De faktorer som vi identifierat utifrån vår studie är som följer:

- Arbete mot betalning
- CV
- Att utveckla förmåga
- Scratch an itch
- Personlig njutning
- Ideologi
- Reciprocitet

Efter en kort beskrivning av vår tolkningsprocess och motivering av vår indelning följer tabell 2.2, vilken fungerar som en uppställning av vår faktoröversättning.

2.3.1 Arbete mot betalning

Vi har valt att kalla den här faktorn för *arbete mot betalning* och den återfinns under annat namn i Hars och Ou (2002). Författarna valde att kalla den för "inkomster av relaterade produkter och tjänster" och vi anser att passar väl in på vår benämning, men vi har förenklat namngivningen något.

2.3.2 CV

Att delta i open source-projekt för att förbättra sin resumé är något vi hittat i alla artiklar. Baytiyeh & Pfaffman (2010) nämner det under sin rubrik "yttre faktorer", men de inkluderar även en del andra faktorer där. Vi anser den vara så viktig att den inte ska klumpas ihop med andra faktorer och har därför valt att göra den till en egen. Benbya & Belbaly (2010) identifierar i sin studie att "karriärsrelaterade fördelar" är en viktig faktor bakom utvecklares bidrag till öppen källkod. De delar upp faktorn i två delar där den första är huruvida de har förväntningar på karriärsrelaterade fördelar och den andra hanterar hur relevanta och viktiga de fördelarna är. Precis som de gör i sin sammanfattning så väljer vi att slå ihop de två delarna till en, eftersom relevansen inte är viktig för vår forskningsfråga. Hars och Ou (2002) tar upp faktorn under sin rubrik "marknadsföring av sig själv".

2.3.3 Att utveckla förmåga

Att utvecklare motiveras av att förbättra sina kunskaper är något som alla de tre tidigare nämnda artiklarna tar upp, men de gör det på olika sätt. Hars och Ou (2002) har en faktor som de kallar "humankapital" och den har vi valt att innefatta i den här faktorn utan vidare översättning. För

Benbya & Belbaly (2010) finns faktorn "inlärning". Inlärning anser vi vara direkt översättningsbart till vår faktor "att utveckla förmåga".

2.3.4 Scratch an itch

Artiklarna nämner konceptet "scratch an itch" men lägger det i många fall under andra faktorer. Vi valde att bryta ut det till en egen då det av open source-gemenskapen uppfattas som en grundpelare för utvecklarengagemang. Konceptet kategoriseras som en "extrinsic motivation", en yttre motivationsfaktor av Baytiyeh & Pfaffman (2010).

2.3.5 Personlig njutning

Under denna rubrik återfinns alla faktorer som är kopplade till individens nöje och känsla av prestation. Dessa återfinns i olika hög grad i samtliga artiklar vi baserar vårt ramverk på. Benbya & Belbaly (2010) kallar faktorn för *prestation*, men detta anser vi vara begränsande och bortser från koncept som erkännande från gemenskapen, vilken Hars & Ou (2002) också fann. Baytiyeh & Pfaffman (2010) går ifrån dessa faktorer och identifierar istället *sociala motivationsfaktorer, motivationen att skapa* och *flödesmotivation*, vilka vi valt att placera under personlig njutning.

2.3.6 Ideologi

Samtliga artiklar kopplar på något sätt motivation till ideologi relaterad till open source. Benbya & Belbaly (2010) använder begreppet ideologi för att beskriva utvecklarens relation till gemenskapen. Vi anser att altruism är en sorts ideologisk drivkraft i open source-sammanhanget. Begreppet altruism används i Baytiyeh & Pfaffman (2010) för att beskriva en individs vilja att hjälpa andra människor. Även Hars & Ou (2002) tar upp altruism i relation till open source-gemenskapen.

2.3.7 Reciprocitet

Benbya & Belbaly (2010) identifierar att reciprocitet kan vara en motivationsfaktor och vi har valt att direkt använda oss av den faktorn i vårt ramverk.

2.3.8 Utgallrade faktorer

Vi har haft svårighet att placera motivationsfaktorn *valens* från Benbya & Belbaly (2010), främst då den inte heller hos dem kan ses som en faktisk motivationsfaktor, utan istället används för att indirekt rangordna andra faktorer efter uppfattat värde hos svarspersoner. Valens syftar till hur viktigt något anses vara och vi kan därför inte passa in denna i vårt ramverk.

2.3.9 Sammanfattning av teoretiskt ramverk

I nedanstående tabell (tabell 2.1) har vi sammanfattat de motivationsfaktorer vi skönjt i tidigare forskning och i vilken av artiklarna vi funnit dem.

Tabell 2. 1 Sammanfattning av motivationsfaktorer i tidigare forskning

Motivationsfaktor	**Författare**
Arbete mot betalning	Hars & Ou (2002), Baytiyeh & Pfaffman (2010)
CV	Baytiyeh & Pfaffman (2010), Benbya & Belbaly (2010), Hars & Ou (2002)
Att utveckla förmåga	Hars & Ou (2002), Baytiyeh & Pfaffman (2010), Benbya & Belbaly (2010)
Scratch an itch	Baytiyeh & Pfaffman (2010)
Personlig njutning	Hars & Ou (2002), Baytiyeh & Pfaffman (2010), Benbya & Belbaly (2010)
Ideologi	Hars & Ou (2002), Baytiyeh & Pfaffman (2010), Benbya & Belbaly (2010)
Reciprocitet	Benbya & Belbaly (2010)

I nedanstående tabell (tabell 2.2) har vi sammanfattat de motivationsfaktorer som vi identifierat i de artiklar vi refererar till. I den högra kolumnen anger vi till vilken faktor i vårt teoretiska ramverk som vi designerat den samt de faktorer som vi gallrat ut.

Tabell 2. 2 Sammanfattning av teoretiskt ramverk

Den tidigare forskningens faktorer	**Vårt ramverk**
Benbya & Belbaly (2010)	
Inlärning	Att utveckla förmåga
Prestation	Personlig njutning
Förväntningar	CV
Valens	Utgallrad
Reciprocitet	Reciprocitet
Ideologi	Ideologi
Hars och Ou (2002)	
Inneboende motivation	Personlig njutning
Altruism	Ideologi
Att identifiera sig med gemenskapen	Ideologi
Försäljning av produkter	Arbete mot betalning
Humankapital	Att utveckla förmåga
Marknadsföring av sig själv	CV
Erkännande från omgivningen	Personlig njutning
Personliga behov	Scratch an itch
Baytiyeh & Pfaffman (2010)	
Motivation att lära sig	Att utveckla förmåga

Motivation att skapa	Personlig njutning
Sociala motivationsfaktorer	Personlig njutning
Flödesmotivationsfaktorer	Personlig njutning
Altruismsmotiveringsfaktorer	Ideologi
Externa faktorer	CV, Scratch an itch

2.4 Litteraturstudie av motivationsfaktorerna

För att få en djupare förståelse för de faktorer vi identifierat har vi valt att nedan göra en studie av var och en av dem. Där kommer vi förklara vad de innebär i en bredare kontext och hur de appliceras på en open source-kontext. Detta kommer att hjälpa oss att få en stabil grund att stå på inför vår analys av det empiriska materialet. Vi kommer att redogöra för vetenskapliga, väletablerade teorier, som till stor utsträckning är identifierad i tidigare forskning, för att ge vidare insikter i hur individer generellt och utvecklare specifikt är motiverade.

2.4.1 Arbete mot betalning

Arbete mot betalning innebär att utvecklaren får betalt för sitt bidrag till open source. Med betalning avses monetär kompensation, alla psykologiska och andra sekundäreffekter som erhålls tas inte hänsyn till. Denna motivationsfaktor skiljer sig något från de andra i att individerna inte längre gör arbetet på sin fritid. Men det är en viktig del av branschen och därför viktigt att ta upp.

Hars & Ou (2002) argumenterar att det finns en begränsning för hur mycket utvecklare är villiga att arbeta gratis. Faktorer som mjukvarans komplexitet och hur pass beroende den är av existerande komponenter kan påverka dess status som open source-mjukvara. En mjukvara som är komplex, och som inte är till stor grad beroende av existerande open source-komponenter har större sannolikhet att inte läggas ut som open source (Hars & Ou, 2002).

Hars & Ou (2002) beskriver en motivationsfaktor bakom bidragande till open source som att sälja relaterade produkter och tjänster. Mer specifikt rör det sig om utbildning, distribution, support och implementationstjänster. Bidrag till en open source-produkt kan dock innebära, om denna förbättras, att behovet av de tidigare nämnda tjänsterna minskar, vilket skapar en viss motstridighet i faktorn (Hars & Ou, 2002).

Sammanfattning

Monetär kompensation är en ytterst relevant, men uppenbar faktor till varför utvecklare väljer att bidra till open source. En utvecklare som bygger en applikation kan förväntas att ta hänsyn till

dess komplexitet och beroende av existerande open source-material vid beslutet om mjukvaran skall bli open source eller inte. Utvecklare kan även förväntas att bidra till open source för att kunna sälja relaterade tjänster och produkter till det projekt de bidragit till. Utvecklarna har ett intresse av att open source-mjukvaran är av sådan kvalitet att den attraherar användare, men samtidigt har utvecklaren ett intresse av att kunna sälja kompletterande tjänster och produkter.

2.4.2 CV

En motiverande faktor bakom deltagande i open source-projekt som bekräftats av ett flertal studier är att förbättra sin resumé och därmed sin anställbarhet (Benbya & Belbaly, 2010; Baytiyeh & Pfaffman, 2010; Hars & Ou, 2002).

Premissen till detta argument är att individer besitter ett humankapital, en uppsättning av färdigheter och personliga egenskaper. Dessa färdigheter och egenskaper som individen har kan brytas ner i indikatorer och signaler (Spence, 1973). I sin signaleringsteori förklarar Spence att indikatorer är karaktärsdrag som individen inte kan manipulera, som till exempel sin ålder. Signaler däremot kan individen förbättra genom ett medvetet beslut att till exempel utbilda sig eller skaffa sig erfarenhet av något. Arbetsgivare bedömer arbetssökande på marknaden utifrån de kombinationer av indikatorer och signaler som de uppvisar.

Att förbättra sina signaler medför dock en kostnad som Spence kallar för en "signaleringskostnad". Definitionen av en signaleringskostnad är avsiktligt bred för att kunna inkludera många olika sorters kostnader. Ett exempel på signaleringskostnader skulle kunna vara den tid och de pengar som vidareutbildning kostar, men kan även inkludera kostnader som förbrukad mental energi och koncentration. Individen som vidareutbildar sig har sannolikt till gång till information om löneförhållanden för individer som får arbete efter den vidareutbildning han valt. Ett rimligt antagande är därför att individen endast vidareutbildar sig om han bedömer att avkastningen blir tillräckligt hög i förhållande till signaleringskostnaderna.

Ett likartat resonemang återfinns i vad som kallas förväntansteorin (expectance theory) (Vroom, 1964 enligt Benbya & Belbaly, 2010). Teorin innebär att en individ agerar endast om denne förväntar sig att handlingen resulterar i att individen får önskade belöningar (Vroom, 1964, enligt Benbya & Belbaly, 2010). Det är belöningarnas attraktivitet och individens förväntningar som styr till vilken grad individen anstränger sig (Bandurra, 1997, enligt Benbya & Belbaly, 2010).

Båda teorier, signaleringsteorin och förväntansteorin, kan appliceras på bidrag till open source-gemenskapen. Till att börja med kan en utvecklare som bidrar till open source ha en viss förväntan på vilka belöningar de kommer att få för sina bidrag till open source-projekt. Belöningarna i fallet med open source är ofta karriärsrelaterade möjligheter och positiva effekter

(Benbya & Belbaly, 2010). Om personens förväntningar är tillräckligt höga för att motivera ett bidrag, och denne bestämmer sig för att bidra till open source kommer, personen att spendera resurser på sitt bidrag, personen utsätts alltså för någon form av kostnad. Utvecklaren kan utsättas för kostnader i bland annat förlorade möjligheter och inkomster när de arbetar i open source-projekt. Det finns dock en långsiktig positiv biverkan av deltagandet, ett karriärsincitament. Karriärsincitamentet syftar till möjligheter till framtida jobberbjudanden och delägande i open source-baserade företag för personen som bidragit (Lerner & Tirole, 2002).

Sannolikheten att en person kommer att engagera sig i signalering kopplas åter igen till belöningsrelaterade förväntningar och signaleringskostnaderna som uppstår (Hars & Ou, 2002; Lerner & Tirole, 2002). Signaleringens effektivitet, och därmed individens motivation att signalera beror på tre faktorer: det styrs av (a) hur pass synlig utvecklarens prestation är för den omvärld utvecklaren riktar sig mot, (b) hur pass stor inverkan utvecklarens ansträngningar har på prestationen och (c) till vilken grad utvecklarens ansträngningar har möjlighet att reflektera deras kompetens (Holmström, 1999, enligt Lerner & Tirole 2002). Dessa faktorer (a, b, c) kan sammanfattas som att utvecklarens motivation ökar desto synligare hans arbete är för omvärlden och desto större grad till vilken han får erkännande för sitt arbete (Lerner & Tirole, 2002).

Om signaleringsteorin ger en korrekt bild av hur arbetssökande interagerar med arbetsgivare på arbetsmarknaden och incitamentet för utvecklare att signalera ökar med prestationernas synlighet och den graden till vilken utvecklaren får erkännande, kan open source-miljön bedömas huruvida den lämpar sig för signalering eller inte.

Open source är tillgängligt för publik inspektion, detta ger en hög grad av visibilitet för utvecklarens arbete. Utomstående kan se vilken utvecklare som bidragit med vad, till exempel vem som skrivit en specifik kod-modul. Utomstående kan även bedöma utvecklarens arbete och därmed hans programmeringsförmåga och förmåga till problemlösning (Lerner & Tirole, 2002). Utvecklarens inverkan på sin egen prestation främjas av open source-miljön efter som att utvecklaren ”är sin egen chef”. Utvecklaren har fullt ansvar för sitt eget arbete och resultat. Open source-miljön ur denna synpunkt kan sättas i kontrast till miljön på ett företag med hierarkisk organisationsstruktur där utvecklaren kan behöva ta hänsyn till organisationspolitik och överordnades åsikter. I detta scenario har utvecklaren inte full inverkan på sin prestation, inverkan den delas istället med andra roller i företaget (Lerner & Tirole, 2002).

Sammanfattning

En person har ett incitament till att förbättra sina signaler och därmed sin anställbarhet. Personen kan förväntas delta i aktiviteter beroende på hans förväntningar. Det är i personens intresse att signalera positiva karaktärsdrag till arbetsgivare, och han kan därför förväntas göra detta om den

positiva vikten av resultatet överväger kostnaderna associerade med signaleringen. Personen kan förväntas att motivera sitt val av open source-gemenskapen som en arena för sin signalering med dess öppenhet och det självbestämmande och kontroll över sitt arbete som gemenskapen tillåter.

2.4.3 Att utveckla förmåga

Utvecklare kan delta i projekt för att de vill utveckla sin förmåga inom området. Med det menar vi att genom att delta i ett open source-projekt så att kan de förbättra sina färdigheter inom programmering, samt lära sig vad det innebär att arbeta i mjukvaruprojekt. Inlärningen kan ske på flera olika sätt, och precis som Baytiyeh & Pfaffman (2010) samt Ye & Keshida (2003) anser vi att en viktig del i kunskapsinhämtandet ligger i att vara del i en gemenskap av andra utvecklare som kan dela med sig av vad de vet. En teori som beskriver hur det fungerar är *community of practice* och därför har vi valt att vidare redogöra för hur den beskriver inlärning, dels generellt och dels i en open source-kontext.

Community of Practice och Situated Learning

Community of practice som term instiftades i början av 90-talet av Lave & Wenger (1991). En community of practice är en grupp av människor som delar mål eller uppgift. De här gemenskaperna finns överallt i vår omvärld och vi är sannolikt alla medlemmar i flera stycken. Exempelvis kan det finnas en på en individs arbetsplats och en i föreningen hon är med i på sin fritid. Inom de här gemenskaperna skapas ett visst sätt att agera. Man utvecklar en viss jargong, vissa ord används och man får en viss praxis för att lösa problem och uppgifter på. Man samlar även på kunnande inom området. En sådan här gemenskap kan vara både formell och informell, den kan vara skapad för ett visst syfte, eller den kan uppstå av sig själv, runt en uppgift som ska göras.

Det finns tre karaktäristiska drag för en commuity of practice (Wenger, 1998; Wenger, 2006):

- **Domänen**. En community of practice är inte endast en gemenskap av vänner eller folk som finns på samma ställe, de måste även vara engagerade och ha ett åtagande till ett visst område. Det innebär också att man delar på en expertis om det här området. Wenger poängterar dock att expertis inte behöver vara något som utomstående anser vara expertis, det kan till exempel handla om hur ett gatugäng delar en expertis om deras identitet och hur de överlever på gatan.
- **Gemenskapen**. Medlemmar i gemenskapen deltar i diskussioner, gemensamma aktiviteter och delar information mellan varandra. De bygger relationer genom vilka de utbyter lärdomar. Det betyder att även om många människor håller på med samma aktivitet, men separat från varandra och utan att forma sociala band, så är det inte en

community of practice. Däremot behöver de inte ses dagligen eller ens särskilt ofta, det räcker att man träffas då och då för att utbyta erfarenheter och bilda sin gemenskap.

- **Praxis**. Individer inom en community of pratice utvecklar ett gemensam repertoar som bland annat innehåller erfarenheter, historier samt redskap att lösa problem med. Denna kunskap kan vara mer eller mindre medveten. Vissa gemenskaper samlar aktivt på lärdomar och dokumenterar dem, i andra sker det utan att medlemmarna lägger märke till det. Wenger (2006) ger som exempel att sjuksköterskor som på fikapausen utbyter historier från dagen kanske inte är medvetna om att det är en av de största källorna till kunskap på deras arbetsplats.

Medlemmar skapar relationer mellan varandra där de utbyter kunskaper och erfarenheter. Nyare medlemmar i gemenskapen lär av mer seniora medlemmar och det är genom att man blir en del av gemenskapen som man faktiskt lär sig. Lärandet ligger alltså i att ta del i hur gemenskapen fungerar. Genom att bli en aktiv medlem så har man lärt sig hur den fungerar. Det är det som Lave och Wenger (1991) kallar situated learning. Lärandet blir något socialt som händer mellan medlemmar, snarare än att man utför vissa aktiviteter. Man identifierar sig även som en medlem i den här gemenskapen.

Lave och Wenger illustrerade sin teori genom att beskriva några olika gemenskaper, bland annat barnmorskor i Yucatec, från en studie av Jordan (1989). Bland de barnmorskorna är lärlingsutbildning det sätt som nya flickor lär sig yrket. Lärandet händer varje dag, hela tiden. Flickorna absorberar kunskaperna från de äldre och kunniga, men inte bara kunskaperna, utan också essensen av att vara en barnmorska, vad det innebär i hur man uttrycker sig, för sig och hanterar sociala kontakter. Allt eftersom de blir äldre går de från att bara observera, till att hjälpa till med små uppgifter, som att bära vatten, till att slutligen vara en barnmorska själva. Genom att delta så blir man en del av gemenskapen och skapar sig en identitet som en del av den gemenskapen, och Lave och Wenger (1991) hävdar att det i stor utsträckning är deltagandet som utgör lärandet.

Open source-gemenskapen som en community of practice

Det går att argumentera för att gemenskaperna som bildas runt open source-projekt är communities of practice. Vi kan börja med en jämförelse med de tre kriterier som Wenger satte upp, beskrivna ovan.

- **Domänen**. Domänen i det här fallet är det gemensamma målet att skapa en viss mjukvara, exempelvis en HTTP-server i fallet Apache, eller en integrerad utvecklingsmiljö, som i fallet Eclipse. De individer som är med i gemenskapen har utvecklat en expertis i hur

man rapporterar buggar, stilmallar för hur man skriver koden som är med i projektet, och mycket annat liknande (Mozilla Developer Network, 2013).

- **Gemenskapen**. Inom de här gemenskaperna förs det diskussioner på hur problem ska lösas, utvecklare ställer frågor och ber om råd från andra i gemenskapen. Exempel på det är kommentarerna på GitHub och buggrapporteringsdiskussioner. Allt eftersom utvecklare blir allt mer involverade i gemenskapen lär de känna andra utvecklare och skapar sig ett kontaktnät. Exempel på det är de små kontaktgrupper som finns i Linuxprojektet, där utvecklarna i den "innersta cirkeln" har en grupp med personer runt sig som de litar på och får kod ifrån (Raymond, 1999). Det finns alltså grupper av individer som sysslar med liknande saker inom projektet och har bildat sociala band emellan sig. Gemenskaperna och diskussionerna visar sig även på webbsidor som Reddit, där exempelvis Linux har en helt egen undersida, där diskussioner om och kring Linux förs. Medlemmar kan ställa frågor, visa upp vad de gjort och ta upp problem de ser (Reddit, 2013).
- **Praxis**. Det finns flera tydliga exempel på hur open source-projekt har flera olika praxisar som medlemmar följer. Om man ska bidra med kod till Mozillas kodbas finns det stilguider för hur den koden ska se ut. De har definierat en mängd regler där man specificerar allt från hur många mellanrum det ska vara vid indentering till hur importerande av bibliotek ska se ut (Mozilla Developer Network, 2013). Ett exempel på att man har gemensamma verktyg är att alla som hjälper till att utveckla Linuxkärnan måste använda sig av versionshanteringssystemet Git för att få lägga till sin kod (Linux Foundation, 2012).

Vi kan med andra ord konstatera att de gemenskaper som finns runt öppen källkod kan klassificeras som communities of practice.

Situated learning sker i open source-projekt

Genom att medlemmar deltar i gemenskapen och blir en del av den, kommer de enligt situated learning att lära sig hur den fungerar. Inlärning kommer ske när de pratar och interagerar med andra, seniora medlemmar lär ut till nyare genom att visa hur de löst problem. Genom att ta del av jargong och lära sig verktygen kommer nya medlemmar, genom en slags informell lärlingsutbildning, att lära sig om ämnet och hur arbetsuppgifterna utförs.

Sammanfattning

Open source-gemenskapen är inte uppbyggd med tydliga lärare och elever, utan utvecklarna förbättrar sina kunskaper genom att delta i projekt. De får information från andra i gemenskapen,

tittar på andras lösningar och får kommentarer på sina egna lösningar. Projekten är med andra ord en bra grogrund för att lära sig nya saker och få nya insikter och för utvecklare som känner en motivation för att utvecklas är det därför en bra plats att befinna sig på.

2.4.4 Scratch an itch

Utvecklare har ett starkt incitament att utveckla en produkt för sina egna behov (von Hippel, 1988 enligt Baytiyeh & Pfaffman, 2010). Begreppet "egna behov" definieras som individens personliga arbetsrelaterade professionella behov, och bortser från psykologiska och ideologiskt inspirerade behov.

Motivationsfaktorn har även kommit upp i studier som Hertel et al. (2003) och Hars & Ou (2002). Hertel et al. (2003) tillfrågade 141 utvecklare som bidragit till open source-projektet Linuxkärnan. Studien konstaterade att en motiveringsfaktor för utvecklare att delta var att lösa Linux-relaterade problem som annars skulle tagit upp tid i deras arbete. Hars & Ou (2002) fann i sin undersökning att en del av respondenterna svarat att de programmerat för att tillfredsställa sina egna behov.

Ett försök till att förklara vad som ligger bakom denna faktor görs av Bonaccorsi & Rossi (2003). Författarna föreslår brister i existerande kommersiell och icke-kommersiell mjukvara som en motivationsfaktor för utvecklare att delta i open source-projekt. En utvecklare kan starta eller delta i ett projekt som mynnar ut i en mjukvaruapplikation som fyller ett behov som ingen annan mjukvara lyckats uppfylla på ett tillfredsställande sätt. Utvecklaren producerar därmed för eget behov (Bonaccorsi & Rossi, 2003).

Ett konkret exempel på resonemanget i Bonaccorsi & Rossi (2003) är webbservern Apache. Arbetet med apache startade på grund av missnöje med en av dåtidens webbservrar. Webbservern i fråga var UNIX-baserad och dess källkod var öppen. Personalen på National Center for Supercomputers i USA var dock inte tillräckligt snabba på att svara på förslag och frågor, enligt Brian Behlendorf och hans kollegor. Tillsammans började dessa individer på arbetet som skulle resultera i Apache Web Server (Lerner & Tirole, 2002).

Ett annat känt exempel är Practical Extraction and Reporting Language, Perl förkortat. Perl är ett programmeringsspråk som skapades år 1978 av Larry Wall. Wall skapade Perl för att assistera honom i hans dagliga arbete med de repetitiva administrationsuppgifter han utförde för sin arbetsgivare, Burroughs. Perl var ämnat för att snabbt utföra administrationsrelaterade uppgifter, och fyllde ett tomrum som Wall ansågs finnas mellan programmeringsspråken Unix Shell och C (Lerner & Tirole, 2002).

Sammanfattning

Baserat på de studier och exempel som nyss granskats kan vi anta att följande gäller för en utvecklares motivation: Om en utvecklare saknar ett verktyg och hans behov inte kan tillfredsställas av det kommersiella eller icke-kommersiella utbudet av mjukvara kan han förväntas att påbörja arbetet med att utveckla ett sådant verktyg. Om utvecklaren stöter på ett problem i hans vardagliga arbete som inte kan lösas med verktyg från det existerande utbudet av mjukvara kan han även då förväntas att lösa problemet med en egenhändigt skapad mjukvarulösning.

2.4.5 Personlig njutning

Personlig njutning som utvecklare upplever som motiverande i deras arbete kan komma i flera former och beskrivas på flera olika sätt.

När en utvecklare skapar en artefakt som han själv och andra kan använda kan skapandet i sig självt vara angenämt för utvecklaren. Speciellt att kunna observera förloppet av en produkt från idé till färdig produkt och slutförandet av detta skall ge utvecklaren personlig njutning (Baytiyeh & Pfaffman, 2010). Utvecklarens nöjdhet med resultatet av produkten är en viktig faktor att ta hänsyn till eftersom att flera open source-projekt aldrig färdigställs (Scacchi, 2002 enligt Benbya & Belbaly, 2010). Det finns en korrelation mellan utvecklares attityd mot open source och deras nöjdhet med projekten och produkterna de bidragit till: Utvecklare som har en positiv inställning till open source kan förväntas dra mer personlig njutning av slutförandet av ett projekt eller en produkt (Benbya & Belbaly 2010).

En annan motivationsfaktor som är relaterad till skapandet i sig självt är vad Baytiyeh & Pfaffman (2010) kallar "flow". Flow är ett behagligt tillstånd som en person kan befinna sig i. Tillståndet uppstår hos någon som är fullt upptagen i en arbetsuppgift där personens färdigheter och förmågor sätts på prov (Baytiyeh & Pfaffman, 2010).

Hars & Ou (2002) talar om en annan sorts personlig njutning, "peer recognition", som kan översättas till "erkännande från omgivningen", som en motiverande faktor till varför personer bidrar till open source. Detta behov av erkännande kommer från människans behov av självkänsla och berömmelse (Maslow, 1987 enligt Hars & Ou, 2002). Erkännandet som författarna talar om kommer i form av feedback på personens bidrag, från gemenskapen. Denna feedback fyller två funktioner. Dels är den en bekräftelse på att bidraget används eller är till någon sorts nytta. För övrigt fungerar det som ett incitament i sig själv för personen att förbättra sitt bidrag i hopp om att få mer positiv feedback (Hars & Ou, 2002).

Sammanfattning

Personlig njutning kan uppenbara sig i flera former. Utvecklare kan förväntas att bidra till open source för att uppnå "flow"-tillståndet eller för att se slutförandet av deras skapelser. Detta är sannolikt mer relevant för utvecklare beroende på hur positiv deras syn på open source är. Vidare kan utvecklarna förväntas att bidra för att de vill ha bekräftelse och uppskattning från andra människor, den personliga njutningen som erhålls av erkännandet av deras färdigheter kan sannolikt leda till att utvecklaren vill fortsätta att bidra även i framtiden.

2.4.6 Ideologi

Faktumet att det finns starka ideologiska kopplingar mellan medlemmarna i open source och deras gemenskap har dokumenterats i ett flertal studier (Benbya & Belbaly 2010; Bonaccorsi & Rossi, 2003; Lakhani & von Hippel 2003). Ideologi definieras av Benbya & Belbaly (2010) som tillhörighet till en viss grupp, samt någon sorts känslomässigt eller värderingsladdat förhållande till medlemskapet i gruppen.

Att identifiera sig med open source-gemenskapen kan innebära för utvecklare att de justerar och riktar om sina värderingar så att de är i enlighet med gemenskapens värderingar (Hars & Ou, 2002). Detta beteende korresponderar till individens behov och begär av acceptans och att bli omtyckt (Maslow, 1987 enligt Hars & Ou, 2002). Det kan ytterligare förklaras av begreppet inneboende motivation vilket syftar till en individs behov av att på egen hand fastställa sin relation till omgivningen (Deci 1975, enligt Hars & Ou, 2002).

En annan typ av inneboende motivation är identifiering med gemenskapen. Identifiering med gemenskapen är en form av altruism på grund av förhållandets natur (Hars & Ou, 2002). Altruism definieras som motsatsen till självviskhet, att göra något positivt för en annan individ på egen bekostnad (Ozinga 1999, enligt Hars & Ou, 2002). Definitionen given av Hars & Ou utelämnar belöningar; Baytiyeh & Pfaffman (2010) hävdar dock att även om den altruistiska individen inte får några konkreta belöningar, erhåller denne psykologiska belöningar. Exempel på psykologiska belöningar kan vara tillfredställelsen av att hjälpa en annan människa (Ross-Ackerman, 1998 enligt Baytiyeh & Pfaffman, 2010). Vidare hävdar Baytiyeh & Pfaffman (2010) att altruistiskt beteende stärker individens ego.

När en individ identifierar sig med open source-gemenskapen är det inte ovanligt att han eller hon blir villig att utföra uppgifter som gagnar andra i gemenskapen, men inte individen själv (Hars & Ou, 2002). Detta är typiskt altruistiskt beteende, men handlingarna är riktade mot en specifik grupp, open source-gemenskapen. Denna typ av altruism kallas för släktskapsselektion. Altruistiskt beteende i en gemenskap stärker individens relation till gemenskapen ytterligare, och

denna förstärkta koppling kan upplevas av individen som en motiverande faktor till att bidra (Baytiyeh & Pfaffman, 2010). Ett konkret exempel på detta är utvecklare som gått till open source-gemenskapen för att få hjälp med ett problem och fått den assistans de sökte är benägna att själva hjälpa andra i gemenskapen (Baytiyeh & Pfaffman, 2010).

Rossi & Bonaccorsi (2003) hävdar att altruism endast kan förklara programmeringsbidrag på hobby-basis. Författarna argumenterar för att altruism som motivationsfaktor inte räcker till för att förklara beteendet hos de individer som har gjort stora bidrag och uppoffringar till fördel för open source-gemenskapen. Ett försök till att förklara varför individer beter sig altruistiskt när arbetsuppgifterna inte är "roliga" och således opassande som fritidsaktiviteter görs av Lakhani & von Hippel (2003) med hjälp av teorin om släktskapsselektion. Utvecklingsarbete som anses som "roligt" kan förklaras av att programmerare arbetar gratis på sin fritid med njutningsfaktorn som motiverande faktor, men faktorer som njutningsfaktorn täcker inte "tråkigt" arbete (Lakhani & von Hippel, 2003). Motivationen bakom arbete med "tråkiga" arbetsuppgifter kommer således inte från altruism, men istället från främjandet av open source som en rörelse, samt en pliktkänsla som härstammar från tidigare erhållen hjälp, och upplevelsen att chanserna att själv få hjälp i framtiden ökar (Baytiyeh & Pfaffman, 2010; Lakhani & von Hippel, 2003). Vid sidan av det egna behovet av support kan politiska och idealistiska motivationsfaktorer urskiljas. Utvecklare i open source-gemenskaper har uttryckt att de känner att de bidrar till konstruktionen av en utopisk gemenskap med syftet att "befria" världen från patentskyddad mjukvara. Var gränserna för denna gemenskap går är oklar, flera utvecklare i Baytiyeh & Pfaffman (2010) hävdade att de arbetade för allmänhetens bästa och att de önskade att deras arbete skulle gagna människor överallt i världen.

Sammanfattning

Ideologi kan alltså delvis förklaras med hjälp av begreppet altruism. Individer kan vara altruistiska så länge arbetsuppgifterna är relativt roliga att genomföra. När vi går från fritidsengagemang till större bidrag kan släktskapsselektion, individers starka identifiering till rörelsen, förklara altruistiskt beteende. Slutligen finns det ideologiska och politiska motiv bakom bidrag till open source-gemenskapen, visioner om utopier och mjukvarans plats i världen styr individers vilja att bidra.

2.4.7 Reciprocitet

En individ som motiveras av reciprocitet agerar under förutsättningen att han eller hon kommer i framtiden att få återgäldning för sitt arbete. Han eller hon kan också agera för att återgälda

tidigare erhållna tjänster. Reciprocitet kommer att diskuteras med hjälp av den sociala utbytesteorin samt en studie av Apache Web Servers support-funktion.

Reciprocitet kan förklaras med hjälp av den sociala utbytesteorin (social exchange theory) (Vroom, 1964 enligt Benbya & Belbaly, 2010). Reciprocitet sker i förhållanden, mellan individer eller mellan organisationer och individer, som kallas sociala utbytesförhållanden. Inom ett sådant förhållande kan en av parterna förvänta sig att generösa handlingar mot den andra parten återgäldas någon gång i framtiden. Det finns ingen tidsram i teorin för när en tjänst skall återgäldas. En tjänst kan även fungera som återbetalning för tidigare erhållna tjänster (Blau, 1964 enligt Benbya & Belbaly, 2010). Detta innebär att i open source-gemenskapen skulle en utvecklare med kunskap som andra i gemenskapen skulle kunna dra nytta av vara motiverad att dela med sig av denna kunskap för att i framtiden få gentjänster av de som dragit nytta av den (Benbya & Belbaly, 2010).

Fenomenet som beskrivs av Benbya & Belbaly (2010) bekräftas i en undersökning av Apache Web Servers support system som sköts genom att användare själva besvarar andra användares frågor (Lakhani & von Hippel, 2003). I undersökningen tillfrågades användare om varför de svarade på andra användares frågor. Bland svaren återfanns anledningar som att öka chansen att själv få hjälp i framtiden, detta tyder på att respondenten har ett socialt utbytesförhållande med Apache Web Server-gemenskapen där respondenten kan förvänta sig återbetalning för sitt bidrag (Lakhani & von Hippel, 2010). Respondenter i undersökningen svarade även att de kände en plikt att återgälda tidigare erhållen hjälp, ännu ett tecken på ett socialt utbytesförhållande, i enlighet med Benbya och Belbaly (2010) samt Lakhani & von Hippel (2003).

2.5 Sammanfattning

Vi har ovan presenterat flertalet olika möjliga faktorer som kan påverka en individ till att delta i open source-projekt. Vi har gett olika perspektiv på faktorerna för att ge en djupare förståelse och grund att stå på inför vår empiriska undersökning.

De faktorer som vi identifierat är arbete mot betalning, CV, att utveckla förmåga, scratch an itch, personlig njutning, ideologi och reciprocitet.

Utifrån den här modellen kommer vi nu gå vidare till att definiera vår insamling av empiri.

2.6 Avslutande kommentarer

Vad som motiverar en individ är inte svart och vitt och därför glider många av faktorerna in i varandra och överlappar på flera ställen. Exempelvis kan det ibland vara svårt att skilja på vad

som tillhör faktorn “ideologi” eller “personlig njutning”, men vi har gjort vad vi anser vara en bra indelning, baserat på tidigare forskning.

3 Metod

Vi ämnar undersöka de motivationer som driver individerna som medverkar i open source. Detta innebär att vi måste samla information om vad som motiverar de medverkande i vad som är ett onekligen komplext fenomen. Vi vill urskilja faktorer i hopp om att identifiera nyckelkoncepten och förhoppningsvist kunna värdera dem gentemot varandra och gentemot existerande forskning.

Vår frågeställning är i grunden lutad mot kvalitativa undersökningar, då dessa är bättre lämpade för att analysera komplexa socio-ekonomiska drivkrafter (Jacobsen, 2002). Traditionellt utförs dessa i en intervjuform som utförs på plats och personligen. Fördelen med detta är främst möjligheten till följdfrågor. Nackdelen är en tendens till att man får exakt det resultatet man hoppas på, då man, om än undermedvetet, leder frågorna mot det man är intresserad av att höra, samt att den undersökta populationen av nödvändighet blir mindre då intervjuer är tidskrävande (Jacobsen, 2002).

I hopp om att balansera fördelarna med en öppen, intervjuliknande undersökning med reliabiliteten och kvaliteten som kännetecknar en omfattande kvantitativ studie tar vår undersökning formen av en tvåstegsraket. Den första delen består endast av en öppen fråga i kvalitativ stil, med hoppet att detta ska ge respondenten möjligheten att leda "intervjun". Med hjälp av informationen vi insamlar från denna första enkät ska vi utforma en andra enkät, med ett renare kvantitativt fokus, där respondenten får värdera ett antal motivationskategorier på en skala. Gemensamt för båda enkäter är att de distribueras på ett antal webbsidor som fungerar som virtuella samhällen för open source-gemenskapen.

3.1 Tillvägagångssätt

Tidigare studier inom ämnet har i stor utsträckning skett med hjälp av formulär på nätet utskickade till olika epostlistor, vilka är vanliga inom gemenskapen. I vissa fall har de varit riktade mot specifika projekt, som exempelvis Linuxprojektet, eller så har e-postadresser samlats in från olika projekt registrerade på sidor på internet. För oss var inte ett specifikt projekt intressant, utan vi ville ha en blandning av utvecklare från många olika typer av projekt för att få så stor spridning på svaren som möjligt, vilket man riskerar att inte få om de medverkande exempelvis bara är från Apache-gemenskapen. Vi valde därför att inte inrikta oss mot e-postlistor från specifika projekt. Vi tyckte däremot att det var ett bra tillvägagångssätt med formulär på internet, eftersom vi vet att många i den här gemenskapen spenderar mycket av sin tid online och är datorvana.

Vi utformade vår undersökning i två steg. Det första steget syftade att validera de faktorer vi identifierat i litteraturgenomgången samt att undersöka om vi kunde skönja fler faktorer. Det andra steget skapade en databas av de uppfattade värdena av faktorerna utefter en form av rangordning.

3.2 Enkäterna

Vår första enkät innehåller en enkel och medvetet öppen fråga

Why do you participate in open source?

Med ordet *participation* fångar vi både aktivt utvecklande individer och andra i stödgivande positioner, som felrapporterande och administrativa roller. Jacobsen (2002) påpekar att enkätundersökningens största svaghet är svårigheten att fånga djup i de tillfrågades åsikter. Vårt hopp är att vårt formulärs öppna karaktär ska balansera detta. Enkätundersökningens styrka är istället möjligheten att nå en så stor målgrupp som är möjligt, då distribution av ett formulär, speciellt i dagens internetbaserade samhälle, förenklar spridning.

Detta tog vi också vara på i den efterföljande enkäten. Till skillnad från den första var denna enkät av en stängd karaktär, respondenten fick ett antal alternativ och rangordnade dessa efter hur de motiveras av dem i deras medverkan i open source samt hur de motiveras av dem på arbetsplatsen. Syftet med denna enkät var att kunna studera hur motivationsfaktorerna stod sig internt mot varandra, samt om och hur de skiljer sig mellan arbetet i open source-gemenskaper och på arbetsplatsen.

Den andra enkäten innehöll två frågor

What motivates you in open source?

What motivates you in the work place?

Under varje fråga följde en tabell med de faktorer vi tidigare identifierat och fastställt i vår teoretiska modell och verifierat i den första enkäten. Respondenten ombads sedan värdesätta varje faktor i en skala på fem steg: *not important, slightly important, important, very important, most important*. Även i denna enkät höll vi antalet frågor till ett minimum, för att bättre matcha målgruppens temperament.

Enkät 1 i sin helhet går att hitta i bilaga 1 och de fullständiga svaren återfinns i bilaga 2. Enkät 2 finns i sin helhet i bilaga 3.

3.2.1 Målgrupp

Vi riktade oss i undersökningen mot individer som medverkar i öppenkällkodsgemenskaper, främst genom att lägga upp en länk till våra formulär tillsammans med en kort introduktion på ett antal communities. Vi hoppades nå en så stor variation av individer som möjligt inom alla aspekter av open source. Valet av webbsidor att lägga upp formulären på gjordes baserad på popularitet och specificitet. En kort undersökning ledde oss till två communities, där man kan förvänta sig en stor andel av vår tänkta målgrupp.

Hacker News

En milt modererad hemsida för användarinlämnade artiklar och länkar. Hacker News har en vinkling mot entreprenörskap och tillåter alla ämnen, men en stor del av artiklarna handlar om att skapa saker. Sidan startades av företagsinkubatorn Y Combinator som ett fokus för den växande hacker- och entreprenörsgemenskapen kring företaget. Det anses i dagsläget vara ett av de mest inflytelserika virtuella samhällena i open source. En viktig del på Hacker News är att utvecklare visar upp sina egna projekt under kategorin "Show HN". Det finns med andra ord en kultur av att dela kod och diskutera öppen källkod, vilket vi tyckte passade vår undersökning väl.

Reddit

Ett parasoll för användarskapade forum för dialog och länkdelning. Reddit är en av de största virtuella gemenskaperna och har huserat publika dialoger mellan både kändisar och presidenter. Under 2012 hade de 37 miljoner sidvisningar (Kerr, 2012). Även om Reddit innehåller några mer tveksamma delar så är diskussionen i fokus. Insatta och välinformerade kommentarer uppmuntras. De kommentarer som inte bidrar till diskussionen uppmanas röstas ner. Det finns välbesökta forum som inriktar sig mot de områden som är intressanta för oss. Här publiceras länkar till artiklar och liknande inom det aktuella ämnet, men även frågor och tips. Vi valde att publicera vår undersökning på de undersidor som heter "Linux", "opensource" och "programming".

3.2.2 Utformning

Vi använde oss av den öppna tjänsten Google Survey då den väl uppfyllde våra spartanska behov. Tjänstens välkända natur hoppades vi också skulle uppmuntra fler att svara på formuläret, då vår målgrupp är vana internetanvändare och har en tendens att undvika okända länkar. Även relevant för oss var Google Surveys förmåga att hantera de plötsliga anstormningar av trafik som är typiska för det moderna nätet.

Vårt första formulär utformades medvetet med syfte att minimera enkätens tendens att begränsa den tillfrågades svar. Att undvika att använda de kategorier vi etablerat i vårt teoretiska ramverk hoppas vi bereda väg för tydligare konflikter mellan våra definitioner och de tillfrågades svar, vilket skulle öka värdet av vår empiri. Den färdiga enkäten finns också tillgänglig i bilagorna.

Det andra formuläret byggdes utifrån svaren på det första i kombination med litteraturstudien. Den sammanslagna bilden de två kunskapsbaserna gav lade grunden för vår renare kvantitativa studie, där vi lät respondenten värdera de framtagna motivationsfaktorerna på en skala från *not important* till *most important*. För att säkerställa att respondenten förstår frågorna till så hög grad som möjligt så introducerades formuläret med en kort beskrivning av varje faktor, så som vi definierat dem. Också denna enkät finns tillgänglig för vidare översikt i bilagorna.

3.4 Analys av empiri

Vår första undersökning är av en kvalitativ typ och enligt Jacobsen (2002) så är det svårt att göra en helt objektiv analys av sådana svar. Det ligger på forskarna att själva göra en bedömning av resultatet. Vi har strävat efter att vara objektiva i vår analys, men inser att det inte är möjligt att vara fullständigt sådana.

Vår analys för enkät ett genomfördes i två steg. Det första steget innebar preparation och indelning av svaren efter de kategorier vi tidigare tagit fram i vårt teoretiska ramverk, samt gallring av ogiltiga svar. Det andra steget involverar en analys av svaret inom den eller de kategorier den överensstämmer med. Dessa analyser sammanfattas kort och ligger tillsammans med tidigare forskning som framställts i litteraturstudien som stöd för utformningen av den andra enkäten.

Ogiltiga svar definierade vi som svar som var kortare än en mening, då man inte kan läsa ut någon kontext. Vår enkät riktade sig endast mot individer som medverkar i open source. Vi ogiltigförklarade därför även de svar som varit från fel synvinkel, nämligen de som är endast användare eller företag.

I vårt teoretiska ramverk har vi identifierat ett antal olika motivationsfaktorer och det är utifrån dessa som vi kategoriserat svaren. Vi satt i grupp och läste tillsammans igenom svaren och tilldelade varje giltigt svar en eller flera faktorer. Vi var även uppmärksamma och letade efter svar som inte passade in på någon av våra fördefinierade faktorer. När vi hittade dessa antecknade vi dem. På samma sätt var vi även uppmärksamma på om någon av faktorerna inte dök upp bland svaren.

Utifrån analysen av det första formuläret och de faktorer som presenterades så reviderade vi vårt teoretiska ramverk. Detta låg sedan till grund för vårt andra formulär med en mer traditionell utformning. Respondenten ombads i detta att rangordna svaren, något som för sig väl i tabell- och diagramform. Formuläret var uppdelat i två delar för att bättre kunna besvara vår frågeställning. Den första halvan frågade efter vad som motiverar utvecklare i open source och den andra efter vad som motiverar dem på arbetsplatsen. Vi hade ett mindre tekniskt bekymmer då våra frågor i enkät två ej markerades som obligatoriska, och var därför tvungna att ogiltigförklara de ofärdiga svar som kom in.

Svaren på det andra formuläret ställde vi upp i tabellform med hjälp av Google Surveys inbyggda analysverktyg, samt medföljande medelvärde och standardavvikelse. Under varje fråga i formuläret gjorde vi en intern rangordning för att komma fram till vilka faktorer som anses vara viktigast. Sedan jämförde vi mellan frågorna efter hur den interna rangordningen ser ut för att urskilja skillnader mellan de uppfattade värdena hos de olika motivationsfaktorerna beroende på om arbetet utförs på arbetsplats eller i ett open source-projekt.

3.5 Kvalitet

Vår undersökningsdata måste uppfylla två krav, validitet och reliabilitet (Jacobsen, 2002). Det första hänvisar till giltigheten hos vår data, att vi mäter det vi faktiskt vill mäta. Det andra till tillförlitligheten, att källor till felaktigheter minimeras och att man, under idealförhållanden, kan upprepa resultaten.

Bruket av validitet och reliabilitet har kritiserats som främst relevanta för kvantitativa studier, delvis då dessa kan anses förmögna att uppnå en högre grad av objektivitet. Kvalitativa studier är medvetet subjektiva, en stor del av arbetet och det uppfattade forskningsvärdet ligger i forskarens förmåga att tolka data. Med detta i åtanke kan de kvantitativa idealen fortfarande, i någon utsträckning, appliceras på kvalitativa studier (Jacobsen, 2002).

För att balansera fördelar och nackdelar med de kvalitativa och kvantitativa skolorna har vi uppdelat vår undersökning i två enkäter. Detta anser vi därför öka kvaliteten på undersökningen.

3.5.1 Validitet

Validitetskravet uppfylls kanske enklast genom ren storlek på datamängden. Mer data förminskar risken att avvikelser ger för stora utslag (Jacobsen, 2002). Med en liten svarsgrupp är det svårt att uppnå hög giltighet på informationen då det är större risk att en majoritet av svarsgruppen är en minoritet av målgruppen. För att maximera antalet svar valde vi att göra båda enkäterna så korta och koncisa som möjligt.

Vår första enkät fick 211 svar, vilket kan ses som ett gott tecken på validitet. Dock måste detta balanseras mot det faktum att enkäten spreds fritt över flera gemenskaper och att vi inte har någon möjlighet att analysera bortfall.

Giltighet kan också delas upp i två delar, intern och extern (Jacobsen, 2002). Den interna giltigheten syftar till att vi mäter det vi ämnar mäta. Det finns en överhängande risk i alla studier att respondenter missförstår frågor eller på grund av olika faktorer svarar osanningsenligt. För att maximera den interna validiteten måste man till en viss grad leda respondenten mot den typ av svar man vill ha. Vårt ställningstagande ligger dock mot att ge målgruppen så stort utrymme till tolkning som möjligt, då detta ökar det kvalitativa värdet på studien. Detta ger studien ett potentiellt internt giltighetsproblem, främst i den första enkäten.

Den externa validiteten härrör i generaliserbarheten av studien, alltså huruvida de specifika resultaten kan appliceras på hela området.

3.5.2 Reliabilitet

Tillförlitligheten refererar till huruvida det är möjligt att upprepa resultaten. Då vår första enkät var i en mer kvalitativ stil och öppnare för tolkning led denna, likt alla kvalitativa studier gör av nödvändighet, av giltighetsproblem ur detta perspektiv. Detta ansåg vi dock lindras av den höga grad i vilken svaren överensstämde med tidigare forskning, ett klart bevis på upprepbarhet. Vår andra enkät är av en sådan stängd art att slutsatser om generaliserbarhet kan motiveras. Vi bedömer att också denna uppfyller validitetskravet.

I vår studie hade vi ingen möjlighet att kontrollera vilka som svarat på frågorna, eftersom formuläret är internetbaserat och öppet distribuerat. Dessa möjliga reliabilitetsproblem har vi accepterat i hopp om att få ett högre svarsantal och en större spridning mellan den typ av respondenter vi riktar oss mot, som vi uppfattar motvilliga mot att ge upp personlig data för insamling. Till fördel för oss förhindrar undersökningsverktyget Google Survey i hög grad risken att respondenter svarar mer än en gång.

Uppdelningen i två enkäter möjliggjorde en större och mer djupgående förståelse av respondenterna, då den andra enkäten utgjorde ett slags uppföljningsfrågor. Detta ledde till en mild sorts dialog vilket gav vår övergripande studie djup. Våra enkäter var av dels kvalitativ och kvantitativ art och bidrog alltså till reliabiliteten (Jacobsen, 2002).

Vi anser också att vår urvalsgrupp är representativ för den målgrupp som vi riktat oss in mot och beskrivit tidigare, genom att främst distribuera enkäten på de virtuella samhällen som är populära

inom målgruppen och endast målgruppen. Det bör noteras att, som ett fritt tillgängligt internetbaserat formulär var det självklart möjligt för även utomstående att svara.

3.5.3 Etik

Enligt Jacobsen (2002) finns det 3 aspekter som är viktiga att uppfylla för att undersökningen inte ska bryta mot etiska regler. Det är att den måste finnas (1) informerat samtycke, (2) rätt till privatliv och (3) krav att bli korrekt återgiven. *Informerat samtyckte* uppfyllde vi genom att vara noggranna med att ange vad studien var till för och att deras svar var en del av vår studie. Vi skrev exempelvis att vi gör en kandidatuppsats på Lunds Universitet om motivationsfaktorerna bakom open source. På det viset visste de som deltog vad det var de bidrog till. *Rätt till privatliv* uppfyllde vi genom att vi inte har samlat in någon information om vilka de är. De som självmant bidragit med kontaktuppgifter har vi tagit bort från bilagorna. I verktyget vi använde finns det inget sätt att ta reda på vilka som svarat, eller varifrån de är. Det är även ett viktigt krav att respondenterna blir *korrekt återgivna*. Vi har i vår studie valt att ibland använda citat från vissa svar för att betona något, men vi har hela tiden varit medvetna om att ett lösryckt citat kan få en annan betydelse när det inte är i sin kontext. Vi har försökt att undvika att ändra betydelse på citaten, men för att helt uppfylla kravet så har vi med alla svaren i sin helhet som en bilaga, så att andra kan läsa det i sin kontext.

3.6 Kritik av metodval

Vad som motiverar en individ är kanske inte alla gånger något som den individen är helt medveten om själv. Det kan vara så att individen vill att svaret ska vara det ena, exempelvis att man bidrar till open source för att hjälpa andra, men egentligen gör man det för mer egocentriska skäl, vilket kan leda till att respondenterna inte svarar helt sanningsenligt i vår undersökning. Vi misstänker att ideologiska skäl kan få större utslag i den första typen av enkät, då individer gärna motiverar sitt arbete på detta sätt. Detta är tyvärr inte något vi kan påverka särskilt mycket, men vi har tagit hänsyn till det när vi drar våra slutsatser. Vi har också försökt att minska risken för det genom att vi i vår andra undersökning bett respondenterna att gradera flera olika kategorier av motivationsfaktorer, vilket förhoppningsvis leder till att vi kommer närmare de egentliga anledningarna.

Vi kan inte veta om det är samma individer som svarat på enkät ett och två, varför dessa två kan skilja sig i faktiskt sammansättning av respondentgruppen. Det är möjligt att en viss subgrupp inom målgruppen attraherades mer till, och hellre svarade på, den första eller den andra enkäten. Vi kan alltså inte dra generellt dra slutsatsen att det är samma grupp individer som svarat på båda enkäter, en förlust vi accepterat.

I den här typen av undersökning har vi begränsad möjlighet till följdfrågor, vilket hade kunnat ge oss en djupare förståelse. Vi hoppas kunna få mer djup och mindre ytlighet i och med vår kombination av två enkäter, den första öppen och den andra som fungerar som ett slags uppföljningsfråga.

3.7 Avslutande kommentar

Vi hoppas alltså att ge ett nytt och fräscht perspektiv med vår tvåstegsenkät som tar vara på fördelar med både kvalitativ och kvantitativ forskning, samt förhoppningsvist minimerar nackdelarna med båda. Eftersom vi förhoppningsvis får större spridning i respondenter än tidigare studier hoppas vi kunna nå en högre grad av generaliserbarhet.

Resultatet och analys av detta, samt avslutande diskussion, ska alltså resultera i en övergripande tolkning av motivationsfaktorer i open source samt interna och externa värdesättningar dem emellan.

4 Presentation och analys av empiri

Vi kommer att inleda med att presentera och analysera vårt första formulär, då detta låg till grund för resten av den empiriska undersökningen. Denna första enkät kommer att analyseras med hjälp av citat och korta beskrivningar. Den andra enkäten analyseras med hjälp av enkla statistiska verktyg som medelvärden och standardavvikelser och presenteras i tabellform. Vi kommer också att, i sådan grad vi kan, beskriva populationen som ligger till grund för våra studier. Eftersom vår undersökning skedde i två steg gjordes också analysen stegvis. För att bättre representera detta är presentation och analys sammanvävda i en sammanslagen struktur.

4.1 Enkät ett

Syftet med denna enkät var att verifiera tidigare forskning samt, om möjligt, identifiera faktorer vi utelämnat. Denna del är uppdelad i en analys av respondenter och en uppställning av de faktorer vi skönjt. Vi kommer inte att presentera varje svar enskilt på grund av dess omfattning; för en fullständig sammanställning av svaren se bilaga 2. Ankätens utseende kan återfinnas i bilaga 1. Varje faktor presenteras med citat och en kort beskrivning av svarsanalysen. Vid varje citat finns referering till dess position i bilaga 2, för att läsaren själv ska kunna se svaret i sin helhet eftersom vi vid behov kortat ner dem. Numret på raden i referensen motsvarar numret till vänster om det fullständiga svaret i tabellen i bilaga 2.

4.1.1 Respondenter

Under loppet av två veckor fick vi 211 svar, av vilka 48 bedömdes ogiltiga i enlighet med vår tidigare definierade metod. Detta lämnade oss med 163 svar. Enkäten togs ner efter 2 veckor; vi såg väldigt lite aktivitet efter de första två dagarna men vi såg ingen anledning att ta ner den tidigare.

På grund av den anonymiserade karaktären och öppna spridningen av vårt formulär, samt bristen på frågor om information så som ålder och kön kan vi inte vidare analysera respondentgruppen. Vi är inte heller medvetna om huruvida det fanns ett bortfall.

4.1.2 Motivation

Även om det är svårt att mäta uppfattade vi svaren som välskrivna och genomtänkta. Vi blev också överraskade av längden på svaren; med en medellängd på 72 ord per svar var det väl över våra förväntningar. Också i kommentarsfälten på Hacker News och Reddit fick vi

överväldigande positiva reaktioner. Vi såg en del engagemang där, ofta ideologiskt, vilket självklart riskerar att färga vår enkätdata.

I vår analys av svaren återfann vi samtliga faktorer vi arbetat in i vårt teoretiska ramverk, men vi upptäckte också en ytterligare faktor som vi kallar *arv*. Mer om denna följer i det mer ingående stycket nedan.

Arbete mot betalning

> *There are paid positions, such as in Linux (80 percent of all contributions are paid or by individuals who are employed to do so) or the RedHat operating system. (rad 99, bilaga 2)*

Denna faktor kan ses som självklar men är fortfarande av stor vikt vid analys av området. Den bryter mot mönstret vi ser i de andra faktorerna, vilket gör möjligheterna till analys svårare, men kan fortfarande inte bortses från. Mot sunt förnuft verkar det som att väldigt få anser att betalning är en viktig motivationsfaktor, men vi misstänker att respondenter inte anger betalning som anledning då de hellre vill se sig själva som idealister eller ha mindre materiella motivationer.

Utav studierna som ingår i vårt teoretiska ramverk har endast Hars & Ou (2002) tagit upp betalning som en motivationsfaktor för att delta i open source-utveckling. Faktorn förekommer relativt sällan i respondenternas svar. Detta skulle kunna tyda på antingen att faktorn är mindre relevant än de andra faktorerna, eller att respondenterna väljer att utesluta den på grund av deras bild av sig själva som ideologer.

CV

> *Career: most of the good interviews I've had in the last few years have started with "I had a look at your github" (rad 16, bilaga 2)*

> *Showing the world what I know. I’ve been getting job offers based on my open source contributions, as well as my own open source projects (rad 59, bilaga 2)*

En stor andel av svaren angav att framtida arbetsmöjligheter var en av de huvudsakliga anledningarna till deras medverkan i open source. Det första citatet refererar till GitHub, en digital samlingsplats för öppen källkodsgemenskapen och de utvecklare som deltar, där de presenterar sina egna projekt. GitHub fungerar i denna mening som ett internetbaserat CV. Vi

lade också märke till att de som angav CV som motivationsfaktor i princip alltid även angav andra faktorer, såsom utveckling av förmåga.

Digitala samlingsplatser som internetbaserade CV är något som inte finns beskrivet i den tidigare litteratur vi tagit del av. Enligt oss speglar det väl vad Michael Spence (1973) säger i sin signaleringsteori som vi beskrivit i litteraturstudien, men denna teori i sig tar inte in faktorer utöver utbildning. Detta delvis då teorin är äldre än internet. Tjänster likt denna lämpar sig väl för signalering i och med att de är så öppna i sin natur. Denna visibilitet beskrivs också av Lerner och Tirole (2002) som en faktor som gör signalering mer effektiv.

Att ett stort antal svar indikerar CV är därför ej förvånande. Individer signalerar sin entusiasm, produktivitet och sitt engagemang genom flitigt medverkande i tjänster som GitHub. Vi kan alltså se en indikation på en bekräftelse av CV-faktorns relevans för deltagande i open source.

Att utveckla förmåga

> *Contributing to open source projects provides me with an irreplaceable education in community, communication, compromise and code (rad 93, bilaga 2)*
>
> *It gives me an opportunity to learn and grow my skills. In my job, deadlines, management and politics often limit how good the software I create can be, but within open-source projects, usually the only limit is how much time and effort I want to invest (rad 112, bilaga 2)*

En annan dominerande faktor är utvecklandet av förmåga. Många respondenter angav att de på olika sätt lär sig att skriva bättre kod, nya tekniker samt samarbete i grupp från öppen källkodsgemenskapen. Ett antal svar indikerade en uppskattning för gemenskapens stora fokus på kvalitet över deadlines, vilket indirekt kan leda till en mer värdefull delning av kunskaper inom yrket, med en högre kvalitet på kunskaperna.

Att respondenterna anger att de lär sig och använder ordet utbildning, trots att de inte är i en traditionell utbildningssituation bestående av lärare och elev, anser vi tyder på att gemenskapen exemplifierar en community of practice, som beskrivet av Lave och Wenger (1991). Respondenterna uppfattar att de lär sig genom att delta i projekten. Denna utbildning verkar även anses ha en hög kvalitet och medverkarna deltar i en slags informell lärlingsutbildning genom deras medverkan i projekt, både i utvecklingstekniker, arbetsmetodik samt samarbete i grupp.

I det andra citatet (112) säger respondenten att open source ger honom en möjlighet att växa kunskapsmässigt och att lära sig saker som han sedan sätter i kontrast till sitt arbete, som vi kan

anta är skiljt från open source-gemenskapen. Respondenten ser deadlines, styrning och politik som hinder, och ser open source som område och gemenskap som hinderlöst. Detta är något som diskuteras av Lerner & Tirole (2002) i en annan kontext, nämligen i kontext av signalering. Argumentet lyder att desto mer inverkan utvecklarens ansträngningar har på hans prestation, desto mer sannolikt är det att han väljer ett visst medium för signalering. Av respondentens svar ovan kan vi se en indikation på att en av Lerner & Tirole (2002) kriterier för en miljös lämplighet för signalering kan appliceras på även andra områden, som till exempel att utveckla förmåga. Denna åsikt ekas av andra respondenter, detta kan vara en indikation på att utvecklare ser styrning och politik som ett hinder för inlärning.

Scratch an itch

> *I like to improve the software I use, either professionally or personally. By pushing it upstream, it can ensure that the changes I made will be improved on in the future and bugs can be found (rad 131, bilaga 2)*
>
> *Usually because I want a program or feature that doesn't exist yet. Sometimes because I've written something that I don't think I'll be able to earn money from, but I think other people might find it interesting or useful (rad 147, bilaga 2)*

Individer verksamma i mjukvaruutvecklingsbranschen har en unik insikt i hur mjukvara generellt fungerar och en naturlig läggning mot att lösa problem. När dessa individer använder mjukvara i sitt dagliga liv och stöter på problem är det därför inte underligt att de vill lösa dessa. Med traditionellt utvecklad stängd mjukvara är detta inte möjligt, men open source ändrar på detta. Här är det istället uppmuntrat att lösa problem och förbättra mjukvara man använder. Detta indikeras tydligt också i respondenternas svar där många av dem anger att de också förbättrar de öppna programvaror de utnyttjar.

Exempel på “scratch an itch”-faktorn finns i Hertel et al. (2003) och Hars & Ou (2002). När existerande kommersiell eller icke-kommersiell mjukvara inte räcker till för att tillfredsställa en utvecklares behov är det inte ovanligt att de löser problemet med hjälp av sina tekniska kunskaper (Bonaccorsi & Rossi, 2003).

I det första citatet uttrycks inte något missnöje med existerande mjukvara, i detta fall kan den bakomliggande anledningen ligga närmare lösandet av tekniska problem i open source för att göra arbetet effektivare. I det andra citatet talar respondenten om en brist i det existerande mjukvarusortimentet, till skillnad från förbättringsbehov som uttrycks i det första citatet. Vi kan därmed utgöra två aspekter av “scratch an itch”-faktorn: Dels en vilja arbeta för att förbättra

nuvarande verktyg, men också motivation som härstammar från ett missnöje med existerande kommersiell och icke-kommersiell mjukvara, i enlighet med Bonaccorsi & Rossi (2003).

Personlig njutning

> *You get recognition, not just for the beauty of the building, but also for the way you have laid the bricks (rad 23, bilaga 2)*
>
> *I release code under open source license because programming is fun and again I have no other use for it (rad 152, bilaga 2)*

Eftersom respondenterna till stor utsträckning bidrar till projekt på sin fritid anser vi att det inte är underligt att många anger personlig njutning som en motivationsfaktor. De har sökt sig till denna bransch för att de är attraherade till utveckling och problemlösning.

Öppenkällkodsprojekt ger medverkande större möjlighet till självbestämmanderätt och kreativt utlopp. Då många av utvecklarna rör sig i denna gemenskap har de ett intresse av att få erkännande från de andra medverkande. Det indikeras också i studien att nöjet i att lära ut är en viktig faktor.

Baytiyeh & Pfaffman (2010) skriver att utvecklare kan uppleva tillfredsställelse av att se en av sina idéer förverkligas och produktifieras. I citatet det första ovan (23) uttrycks en åsikt mer närliggande "peer recognition" eller erkännande från omgivningen (Hars & Ou, 2002). Feedback av denna typ skall vara motiverande för utvecklare och motivera dem till att vidareutveckla och förbättra sina bidrag (Hars & Ou, 2002). Vi noterade att flera respondenter uttryckte en glädje i att få uppskattning och feedback från användare och andra medverkande. Det andra citatet (152) uttrycker en njutning av själva programmeringen och inte, som i första citatet, erkännandet som kommer från omgivningen. Detta är närmare vad som beskrivs av Baytiyeh & Pfaffman (2010). Utav svaren kan vi alltså urskilja att båda formerna av personlig njutning, det vill säga erkännande och arbetsglädje, finns representerade bland respondenterna och bör ses som relevanta delar av faktorn.

Ideologi

> *I believe that everybody should have the right to analyze and modify the software that she uses. Ultimately, this leads to better software and thus is helpful for humanity as a whole (rad 25, bilaga 2)*

> *It's freedom from dependance on proprietary developers who widely only care about how much money that they can separate from their users, among many other reasons (rad 85, bilaga 2)*

> *Ownership of an idea is something that needs to be outgrown (rad 132, bilaga 2)*

Svaren under denna kategori var överlägset mest omfattande och emfatiska. Svaren är väl genomtänkta och polerade, antagligen ett resultat av många år av diskussion inom gemenskaper. Faktorn medföljdes också av en stark motvilja mot det proprietära och det stängda. Utvecklarna ser stora problem i mjukvarubranschens tendens mot slutenhet och upphovsrättslag generellt.

Dessa svar kännetecknas av att de lägger en stor vikt vid problemen som de anser existerar i branschen idag och de identifierar sig med en motståndsrörelse mot kommersialismen och det proprietära. Altruism är en annan vanlig faktor bland svaren, respondenterna beskriver en glädje i att dela med sig av sina kunskaper till andra, även utan ett direkt löfte om reciprocitet.

Det är svårt att inte fundera på huruvida ideologernas stora engagemang och drivkraft kan tämjas och utnyttjas även på arbetsplatsen. Om detta skulle vara möjligt kan man tänka sig att det skulle leda till både innovation och produktivitet.

I enlighet med vad som konstaterats i andra kapitlet av denna uppsats betecknas ideologi av en identifiering med, samt ett känsloladdat förhållande till en grupp (Benbya & Belbaly, 2010). Ett tydligt exempel på detta är motstridigheten mot kommersialisering av mjukvara till fördel för mjukvara som utvecklats av den egna gemenskapen. Denna fientlighet mot den "stängda" och "proprietära" mjukvaran återfinns i flera av respondenternas svar, till exempel i det andra och tredje citatet ovan kan vi urskilja en tydlig motvilja mot kommersiella utvecklare och ägande. Denna fientlighet mot det proprietära beskrivs av Baytiyeh & Pfaffman (2010) samt önskan att, och känslan av att utvecklarens bidrag gör gott för mänskligheten i stort.

Reciprocitet

> *Just about everything I learned about coding, I learned by hacking on code that someone else made available. I release free software to give back to the community that made me what I am (rad 4, bilaga 2)*

> *A secondary motivation is that I use a whole lot of free software daily, it would be rude not to contribute back (rad 2, bilaga 2)*

Respondenterna antyder att de känner en viss skuld till gemenskapen eftersom att de tidigare använt mjukvara som utvecklats i projekt ledda av den. De anger att de med sina kunskaper har möjlighet att ge tillbaka till människor som de fått mycket från.

Även om mjukvaran distribuerats utan krav och utan förväntningar på användarna verkar det som att det skapas en känsla av vilja ge tillbaka. Detta fenomen av givande och tagande står i stark konstrast mot den generellt kapitalistiska branschen som de befinner sig i.

Citaten ovan, och liknande svar är typiska exempel på den pliktkänsla utvecklare känner gentemot gemenskapen som dokumenterats av Lakhani & von Hippel (2003). Från svaren kan vi urskilja ett så kallat socialt utbytesförhållande mellan respondenten och open source-gemenskapen.

Arv

> *I like to help other people. I feel useful doing open source. When I'm gone my work may live after me! (rad 84, bilaga 2)*

Under analysen av enkätsvaren upptäckte vi en faktor vi hade svårt att dela in i vårt ramverk. Arv är bland de ovanligare nämnda faktorerna bland respondenterna men vi misstänker att den kan vara av stor vikt för förståelse av utvecklarna och dess motivationsfaktorer. Det finns ett starkt grundläggande behov hos människor av att lämna något efter sig till eftervärlden, och detta återfinns också i mjukvaruutvecklingsbranschen. Som motivationsfaktor kan glädjen att skapa och vetandet det man skapat lever vidare vara värdefull. Den öppna källkodsvärlden gör skaparen synlig i en mycket högre grad än i kommersiella projekt där man ofta inte vet vem som skrivit vad.

Arv-faktorn är ny för författarna på så vis att den inte uppkommit i någon av källorna som litteraturstudien är baserad på. Faktorns låga frekvens i studien, i kombination med att den inte nämnts i tidigare forskning som vi studerat, kan antyda att den är ovanligare som motivationsfaktor till engagemang eller att den har en lägre valens, vilket betyder att den är mindre sannolik att nämnas. Bristen på denna faktor kan bero på att själva belöningen ligger så pass långt i framtiden, efter individens bortgång, att utvecklare finner större motivation i belöningar som ligger närmare nutid.

4.1.2.9 Sammanfattning

Vi återfann alla de faktorer tidigare forskning identifierat, men vi reviderade också vårt ramverk med en ny faktor som vi kallar *arv*. Nedan följer vårt uppdaterade ramverk i tabellform, med citat från enkät ett för varje faktor. Alla svar från enkäten återfinns numrerade i bilaga 2.

Tabell 4. 1 Sammanställning av enkät 1, citat

Arbete mot betalning	There are paid positions, such as in Linux (80percent of all contributions are paid or by individuals who are employed to do so) or the RedHat operating system (*rad 99, bilaga 2)*
CV	Career: most of the good interviews I've had in the last few years have started with "I had a look at your github" (rad 16, bilaga 2) Showing the world what I know. I've been getting job offers based on my open source contributions, as well as my own open source projects (rad 59, bilaga 2)
Att utveckla förmåga	Contributing to open source projects provides me with an irreplaceable education in community, communication, compromise and code (rad 93, bilaga 2) It gives me an opportunity to learn and grow my skills. In my job, deadlines, management and politics often limit how good the software I create can be, but within open-source projects, usually the only limit is how much time and effort I want to invest (rad 112, bilaga 2)
Scratch an itch	I like to improve the software I use, either professionally or personally. By pushing it upstream, it can ensure that the changes I made will be improved on in the future and bugs can be found (rad 131, bilaga 2) Usually because I want a program or feature that doesn't exist yet. Sometimes because I've written something that I don't think I'll be able to earn money from, but I think other people might find it interesting or useful (rad 147, bilaga 2)
Personlig njutning	You get recognition, not just for the beauty of the building, but also for the way you have laid the bricks (rad 23, bilaga 2) I release code under open source license because programming is fun and again I have no other use for it (rad 152, bilaga)
Ideologi	I believe that everybody should have the right to analyze and modify the software that she uses. Ultimately, this leads to better software and thus is helpful for humanity as a whole (rad 25, bilaga 2) It’s freedom from dependance on proprietary developers who widely only care about how much money that they can separate from their users, among many other reasons (rad 85, bilaga 2) Ownership of an idea is something that needs to be outgrown (rad 132, bilaga 2)

Reciprocitet	Just about everything I learned about coding, I learned by hacking on code that someone else made available. I release free software to give back to the community that made me what I am (rad 4, bilaga 2) A secondary motivation is that I use a whole lot of free software daily, it would be rude not to contribute back (rad 2, bilaga 2)
Arv	I like to help other people. I feel useful doing open source. When I'm gone my work may live after me! (rad 84, bilaga 2)

4.2 Enkät två

Med grund i den tidigare forskning vi funnit i litteraturstudien samt våra egna empiriska resultat från den första enkäten går vi vidare med en nästa enkät. Vi har identifierat och skapat ett ramverk av motivationsfaktorer som vi nu vill kunna ställa upp och jämföra, samt värdera. Analysen kommer här ta form i tre delar; vi analyserar och jämför svaren på varje fråga för sig, sedan avslutningsvist ställer vi de två mot varandra. Varje faktor redovisas med diagram och tabell över resultatet, samt standardavvikelse och medelvärde. Medelvärdet tas fram genom att konvertera våra svarsnivåer, *not important* till *most important*, till siffror, 1-5. Den presenteras sedan med korta anteckningar från oss. Varje delfråga på enkäten sammanfattas sedan mer djupgående. Slutligen görs en analys av de två frågorna i kontrast mot varandra.

Definitionslista

I tabeller och analyser kommer vi att använda omväxlingsvist engelska och svenska, då enkäterna distribuerades på engelska. För att förenkla och förtydliga följer här en uppteckning av de ord vi kommer använda och hur vi översätter dem samt deras kodade motsvarighet i tabellerna.

Tabell 4. 2 Översättning av benämningar, enkät 2

Not important	Oviktigt	1
Slightly important	Någorlunda viktigt	2
Important	Viktigt	3
Very important	Väldigt viktigt	4
Most important	Viktigast	5

4.2.1 Respondenter

Enkäten låg uppe i tre dagar och ansamlade 206 svar, varav 20 ofullständiga. Detta gav oss 186 giltiga svar. Enkäten låg endast uppe 3 dagar, vilket gav oss en tillräcklig svarsmängd. Vi såg likt vår tidigare erfarenhet väldigt lite trafik efter de första två dagarna.

Likt det föregående distribuerades detta formulär på en öppen samlingsplats för open source-gemenskapen, vi kan därför inte analysera något bortfall. Som vi beskrivit i metod-delen har vi undvikt att samla information om respondenterna utöver våra faktiska frågor, och har därför ingen grund att göra en analys av respondenterna på.

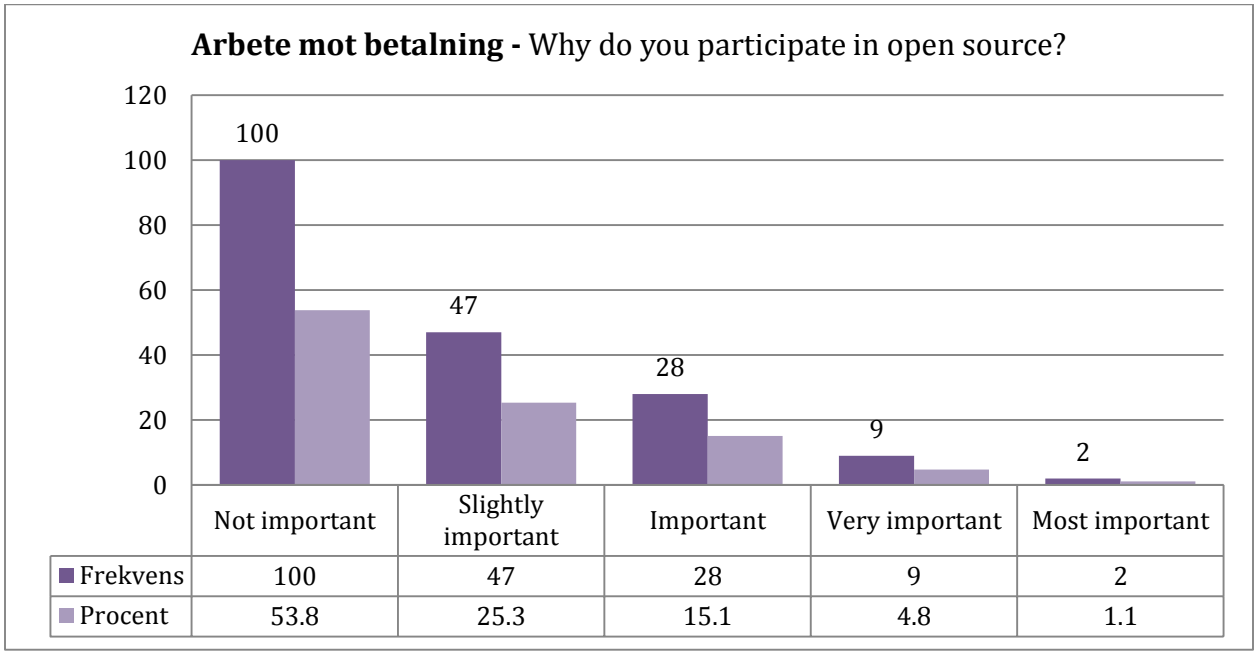
Som tidigare nämnt har vi publicerat den här enkäten på forum som är helt öppna för alla, men de som besöker dem har antagligen ett stort intresse för mjukvaruutveckling, eftersom länkar och liknande på sidorna behandlar till största delen det ämnet. Däremot tror vi att olika typer av individer på de här sidorna är olika benägna att klicka på och svara på vårt formulär. De som brinner mycket för ämnet open source och har en stark känsla av ideologi är antagligen troligare att vilja svara på våra frågor och dela med sig av sina åsikter. De som däremot har mjukvaruutveckling endast som arbete är troligtvis inte lika motiverade att klicka på vår undersökning och därför får vi eventuellt en underrepresentation av dem.

4.2.2 Motivation i open source

Den första frågan ber respondenten rangordna våra angivna faktorer i till vilken grad de motiveras av dem i sitt engagemang i open source. Vi ställer upp detta i tabellform samt presenterar medelvärde och standardavvikelse. Detta följs sedan för varje faktor med en analys av vår data. Frågan är avgränsad till motivation inom open source specifikt.

Arbete mot betalning

Figur 4. 1 Svar enkät 2, arbete mot betalning i open source



Medelvärde: 1,74

Standardavvikelse: 0,958

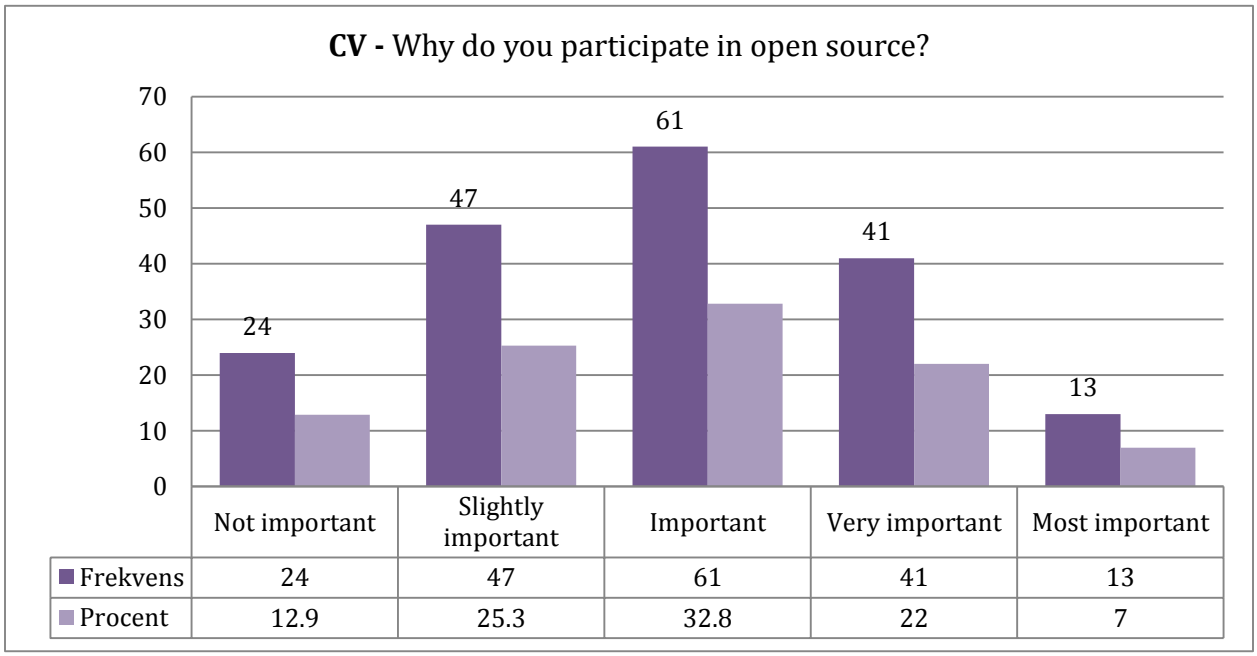
Vi kan se en indikation på att betalning för att medverka inte uppfattas som en viktig faktor. Mer än hälften (100) av respondenterna ansåg det vara oviktigt. Detta antal är betydelsefullt eftersom att det är den största mängden som något av svarsalternativen i undersökningen fått. En tolkning av detta skulle kunna vara att respondenterna är överens om att betalning inte är viktigt. Detta stämmer väl överens med att vi tidigare identifierat att de flesta bidrar med tid på sin fritid. Som vi nämnde tidigare under rubrik 4.2.1 kan det vara så att de som faktiskt får betalt för att utveckla öppen källkod på arbetstid har varit mindre benägna att svara på enkäten och det kan påverka resultatet av den här frågan. Det kan även vara så att de som svarar väljer, omedvetet eller medvetet, att tona ner den här faktorn eftersom det i gemenskapen och i individens interna motivationsmodell kan anses vara mindre ärofullt att vara engagerad för pengarnas skull. Det skulle kunna leda till att färre anger den som viktig.

Arbete mot betalning som motivationsfaktor återfinns endast i en av tre artiklar i vårt teoretiska ramverk. Representationen av denna faktor i den första enkätundersökningen var också relativt låg. Vår data visar klart och tydligt att faktorn inte ses som särskilt viktig, detta blir ännu tydligare när den sätts i kontrast till resultatet för många av de andra motivationsfaktorerna. Den

relativt låga standardavvikelsen på 0,958 tyder på enighet bland respondenterna. I kombination med medelvärdet på 1,74 kan vi utläsa att motivationsfaktor inte sågs som särskilt viktig. Hars & Ou (2002) tar upp försäljningen av relaterade tjänster och produkter till den open source-produkt som utvecklats som ett finansiellt incitament till utveckling. Vår data tyder på att detta incitament inte ses som viktigt av respondenterna som svarat.

CV

Figur 4. 2 Svar enkät 2, CV i open source



Medelvärde: 2,85

Standardavvikelse: 1,119

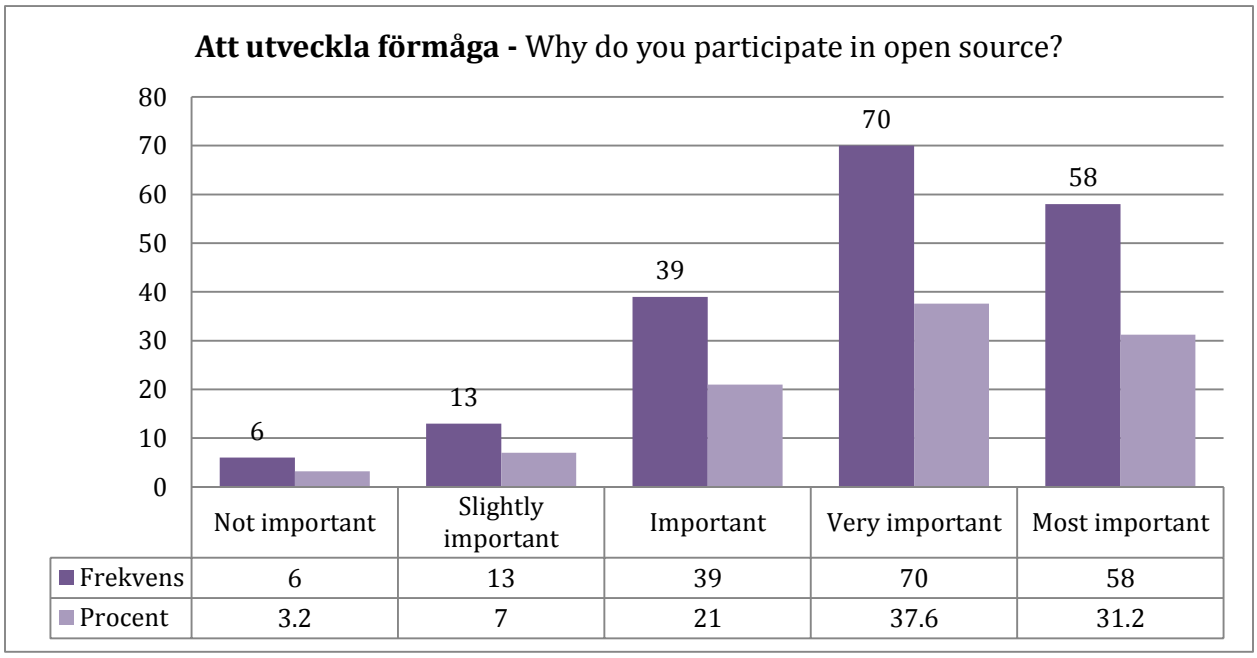
Datan visar att de den största andelen respondenter satte CV-faktorn, deltagandet i open source för att förbättra sin resumé och anställbarhet, som viktigt (32,8 %). Det näst mest populära alternativet var *någorlunda viktigt* (25,3 %) följt av *väldigt viktigt* (22,0 %). Fler respondenter ansåg att faktorn var *oviktig* (12,9 %) än som ansåg att den var *viktigast* (7,0 %).

CV-faktorn återfinns i samtliga studier i det teoretiska ramverket. Det var även en faktor som uppkom frekvent i den första enkätundersökningen. Enligt vad Lerner & Tirole (2002) skriver är open source-gemenskapen en miljö som lämpar sig för signalering. Trots detta kan vi se i datan att en fjärdedel ansåg det endast som *någorlunda viktigt* och 12,9 % såg det som *oviktigt*. Detta

skulle kunna tala emot Lerner & Tirole (2003) teori. Medelvärdet ligger under 3 (2,85), vilket tyder på att fler respondenter såg faktorn som mindre viktig. Alternativet är att de respondenter som inte tyckte att CV-faktorn var särskilt viktig helt enkelt inte hade ett intresse eller behov av signalering. Standardavvikelsen ligger på 1,119 vilket antyder en viss oenighet. Ett betydande antal respondenter svarade trots allt åt hållet att CV-faktorn var viktig, faktorn är alltså viktig, men inte viktigast för respondenterna.

Att utveckla förmåga

Figur 4. 3 Svar enkät 2, att utveckla förmåga i open source



Medelvärde: 3,87

Standardavvikelse: 1,039

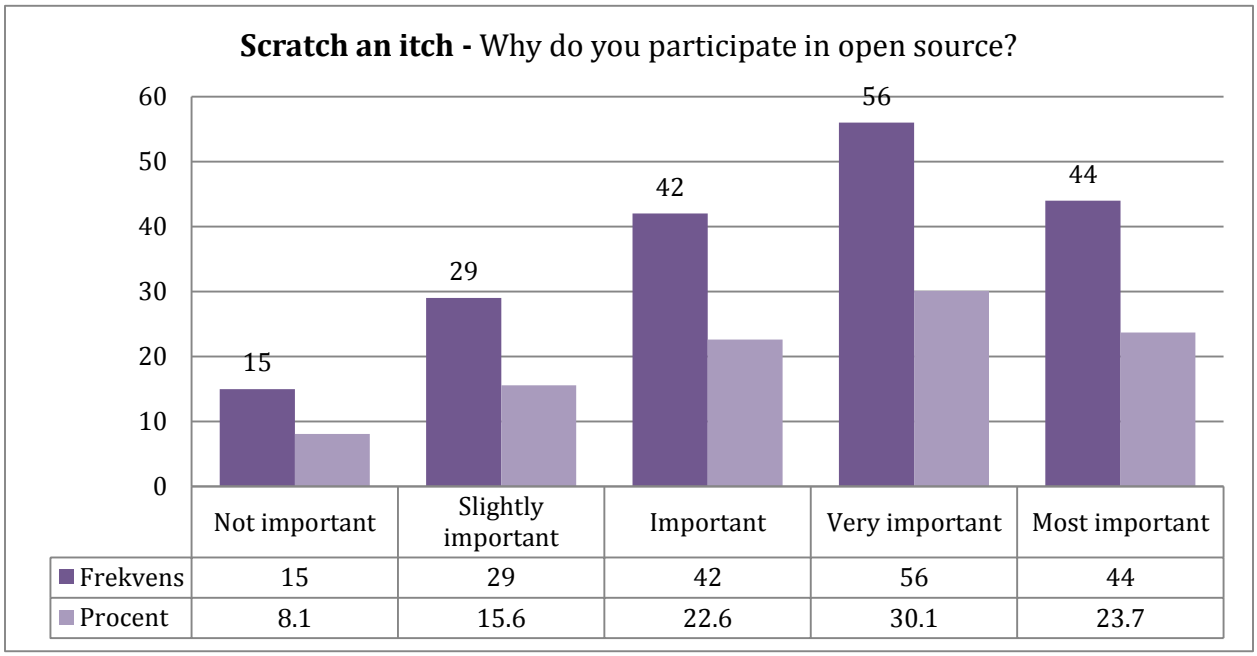
Att utveckla förmåga ansågs vara en viktig faktor av respondenterna. Endast 3,2 % ansåg att faktorn var *oviktig*, 7,0 % ansåg att den var *någorlunda viktig* medan 21,0 % ansåg att faktorn var *viktig*. Det mest populära svarsalternativet var *väldigt viktigt* och fick 37,6 %, följt av *viktigast* med 31,2 %.

Motivationsfaktorn "att utveckla förmåga" har dykt upp i samtliga studier som vi inkluderat i vårt teoretiska ramverk. Av datan kan vi få en indikation på att utvecklandet av den egna förmågan är viktigt. Detta bekräftas av medelvärdet 3,87 som ligger väl över värdet för *viktigt*, 3.

Standardavvikelsen på 1,039 tyder på en viss enighet bland respondenterna. Detta resultat reflekteras av responsen till vår första enkätundersökning där flera respondenter ansåg att inlärningen var väldigt viktig. Detta kan ha att göra med att open source-gemenskapen som en *community of practice* är en effektiv miljö för inlärning och utveckling av den personliga förmågan.

Scratch an itch

Figur 4. 4 Svar enkät 2, scratch an itch i open source



Medelvärde: 3,46

Standardavvikelse: 1,235

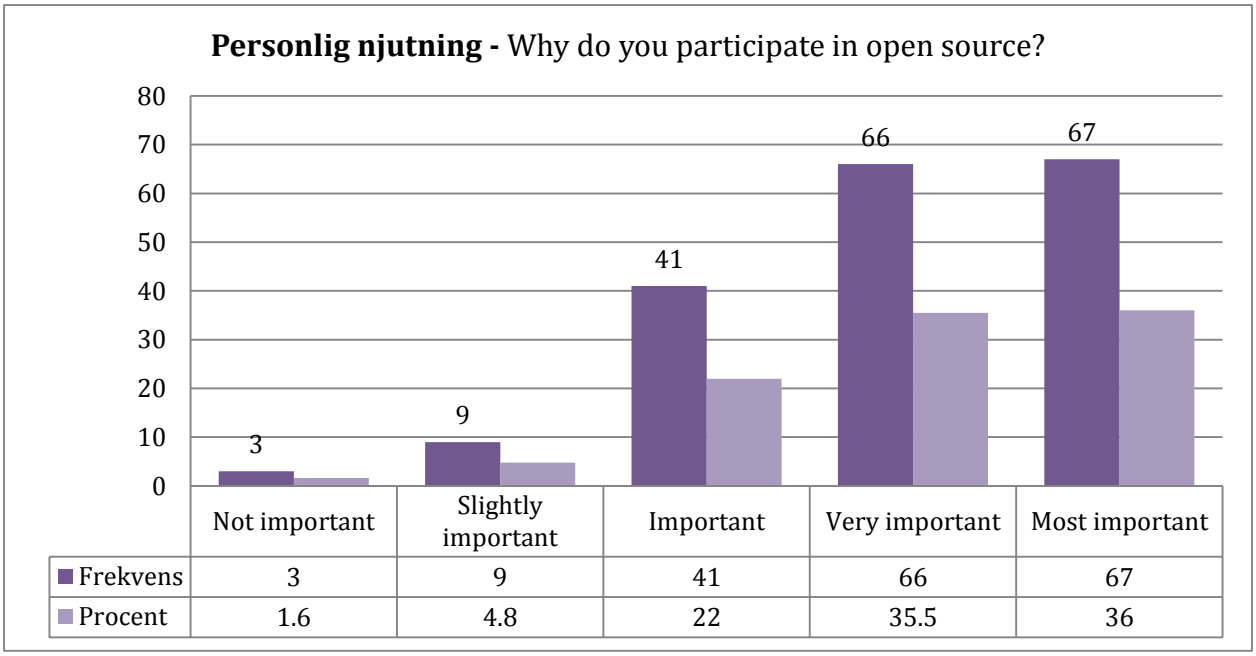
En fjärdedel av respondenterna ansåg att faktorn "Scratch an itch" var viktigast som motiverande faktor bakom bidrag till open source. 30,1 % markerade faktorn som *väldigt viktig* medan 22,6 % markerade den som endast *viktig*. *Någorlunda viktig* och *oviktig* fick 15,6 respektive 8,1 procentenheter.

Scratch an itch är en faktor som nämns av Baytiyeh & Pfaffman (2010). Faktorn kom även upp i vår första enkätundersökning. Av datan från vår andra enkätundersökning kan vi avläsa att scratch an itch är en faktor med en viss viktighetsgrad för respondenterna. Medelvärdet av 3,46 tyder på att faktorn uppfattades som viktig. Faktorns existens förklaras av Bonaccorsi & Rossi

(2003) som en reaktion till brister i existerande kommersiell och icke kommersiell mjukvara. Anledningen till att faktorn är av hög viktighet kan alltså tolkas som ett symptom på ett annat problem, ett mjukvarurelaterat problem som utvecklare väljer att lösa på egen hand istället för att förlita sig på återförsäljarna eller andra utvecklare. Standardavvikelsen på 1,235 tyder på en viss oenighet bland respondenterna, men oenigheten visar sig till största del mellan de tre "viktigare" svarsalternativen, i enlighet med faktorns viktighet.

Personlig njutning

Figur 4. 5 Svar enkät 2, personlig njutning i open source



	Not important	Slightly important	Important	Very important	Most important
Frekvens	3	9	41	66	67
Procent	1.6	4.8	22	35.5	36

Medelvärde: 3,99

Standardavvikelse: 0,961

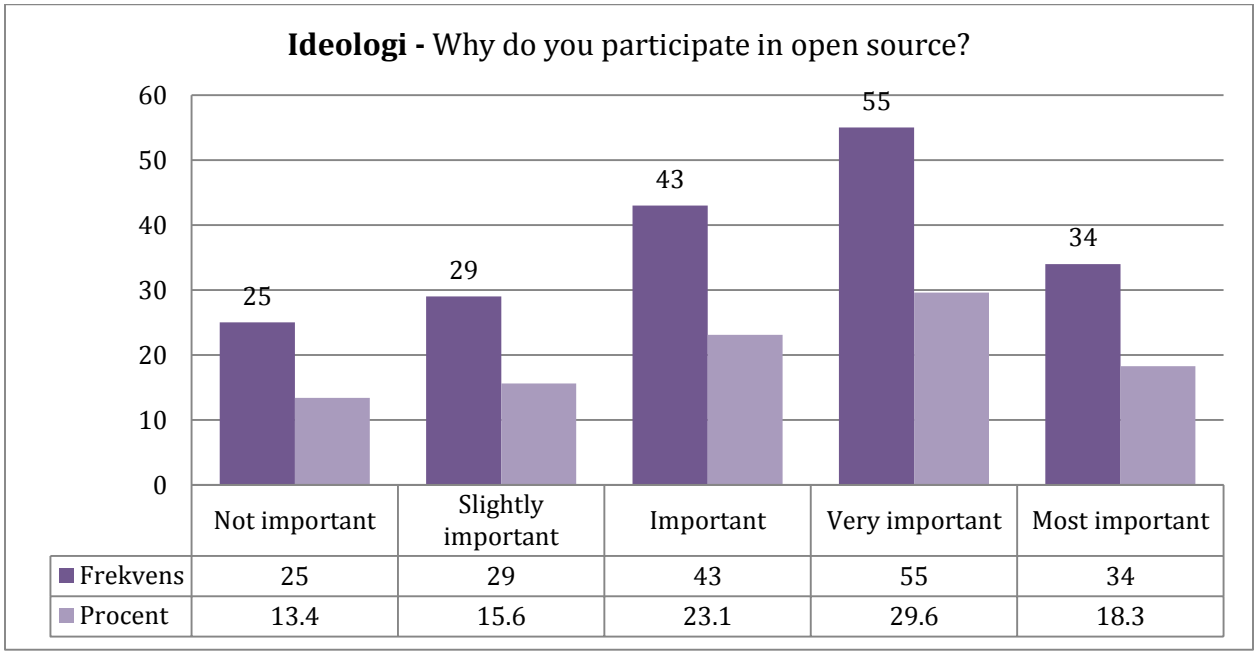
Personlig njutning som motiverande faktor bakom bidragande till open source var en mycket viktig faktor för respondenterna i vår undersökning. 35 % av respondenterna svarade att de såg njutning som den viktigaste faktorn och ytterligare 35 % ansåg att faktorn var *väldigt viktig*. Endast 1 % av respondenterna såg njutning som *oviktigt*, 6 % svarade att njutning var *någorlunda viktigt* och 23 % besvarade frågan med att njutning var *viktigt*. Njutningsfaktorn gav en av de mer ojämna svarsfördelningarna i undersökningen, men nästan uteslutande i det positiva spektrumet. Njutning uppfattas som viktigt, men i olika grader. Personlig njutning, eller *enjoyment*, kan ses som en sekundär faktor av många, då den möjligtvis uppfattas som att den

förminskar deras arbete. En hobby är något man gör för skojs skull, men öppenkällkodsprojekten dessa individer medverkar i är mer än en hobby.

Njutning som motiverande faktor återfinns i samtliga studier som ingår i vårt teoretiska ramverk. Som vi kommit fram till i vår litteraturgenomgång kommer njutningsfaktorn i flera former. Vi kan baserat på datan göra antagandet att en stor majoritet av respondenterna eftersträvar någon slags personlig njutning. Det relativt höga medelvärdet av 3,99 hjälper till med att stödja detta antagande. Vi ser även en mindre standardavvikelse på 0,961 vilket tyder på en viss enighet. Det skulle kunna vara ett sorts flow-tillstånd eller förverkligandet av en produktidé som beskrivs av Baytiyeh & Pfaffman (2010) som fungerar som motivation. Alternativt skulle det kunna vara självkänslan och berömmelsen som Hars & Ou (2002) beskriver under benämningen "erkännande från omgivningen". Vi kan konstatera att oavsett form, är njutningsfaktorn en dominerande faktor när det gäller bidrag till open source-gemenskapen.

Ideologi

Figur 4. 6 Svar enkät 2, ideologi i open source



Medelvärde: 3.24

Standardavvikelse: 1,294

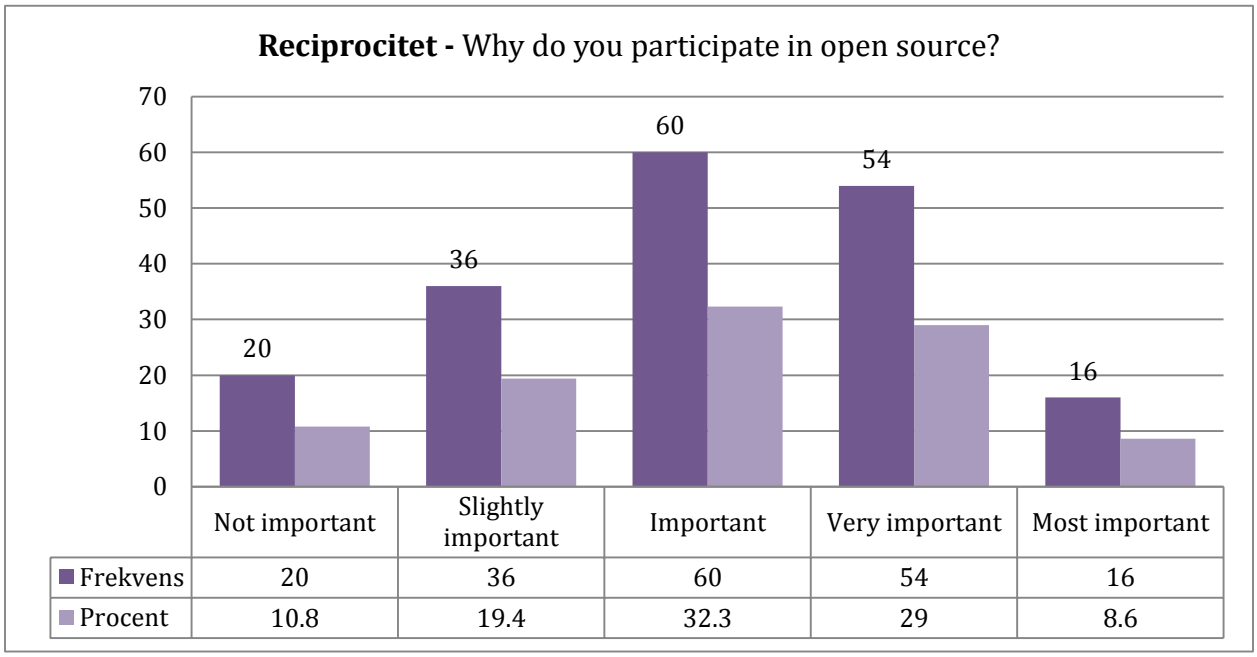
Det största antalet respondenter svarade att ideologi som motiverande faktor för bidrag till open source var *väldigt viktig* (30 %). Det näst populäraste svarsalternativet var *viktigt* (22 %), följt av *viktigast* (19 %). Ideologi är den kategori som har jämnast fördelning av respondenternas svar. Endast fem procentenheter skiljer antalet som tyckte att ideologi var *oviktig* (14 %) och de som tyckte att ideologi var *viktigast* (19 %).

Hars & Ou (2002) hävdar att identifikation med open source-gemenskapen kan innebära att utvecklare riktar om sina värderingar i enighet med open source-gemenskapens värderingar. Detta påstående i kontrast till vår data indikerar att ungefär en tredjedel av de som bidrar till open source inte identifierar sig med gemenskapen. Detta är ett tecken på att de tankar som uttrycks av respondenter i bland annat Baytiyeh & Pfaffman (2010), om att befria världen från kommersiell mjukvara, inte resonerar effektivt med lite över en fjärdedel av individerna som bidrar. Detta förstärks av medelvärdet på 3,24 som endast ligger 0,24 över gränsen för *viktigt*. Standardavvikelsen är relativt hög jämfört med de andra faktorerna och tyder på en viss oenighet. Det ska dock påpekas att en stor del av respondenterna markerade ideologi som *viktigt* till *viktigast*, vilket fortfarande indikerar att open source-gemenskapen drivs till stor del av ideologi.

Vi kan alltså se en indikation på att medan open source-gemenskapen har starka ideologiska drivkrafter, är det inte alla som identifierar sig med, och håller med om politiken bakom dessa.

Reciprocitet

Figur 4. 7 Svar enkät 2, reciprocitet i open source



Medelvärde: 3,05

Standardavvikelse: 1,123

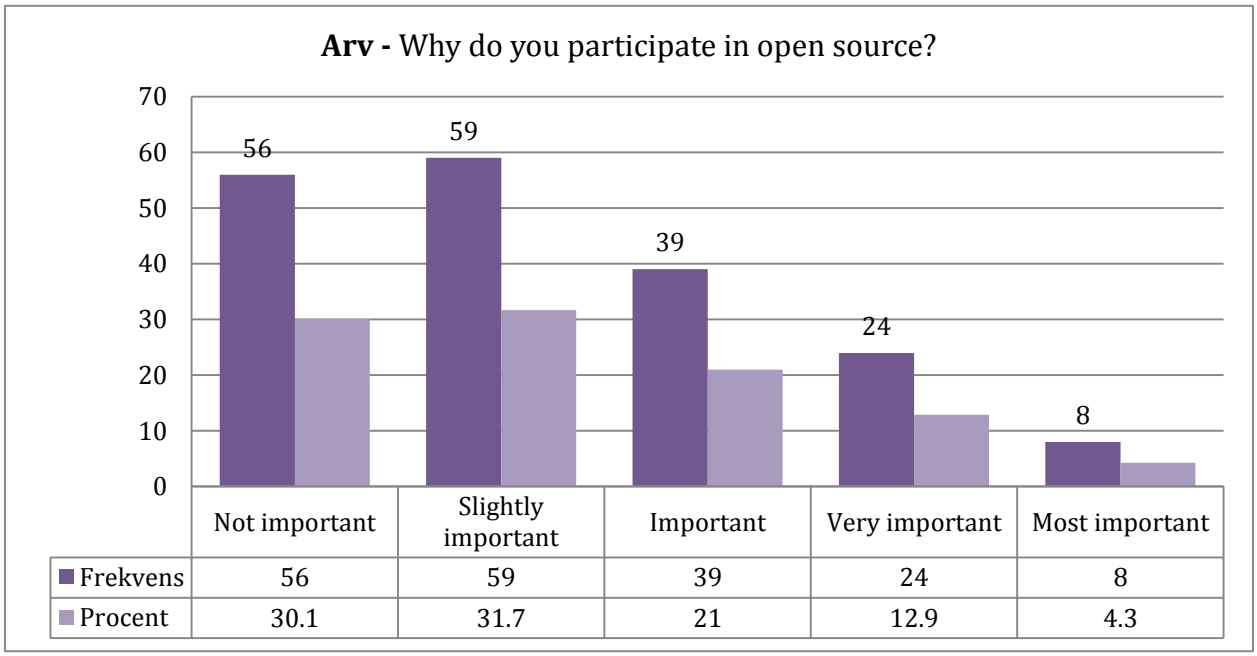
I datan för faktorn reciprocitet ser vi relativt små svarsmängder i de två yttre svarsalternativen. 11 % av respondenterna markerade reciprocitet som *oviktig* och 9 % av respondenterna markerade kategorin som *viktigast*. Högst antal röster fick *viktig* med 32 %, följt av *väldigt viktig* med 28 % av rösterna. Reliabiliteten av datan när det gäller reciprocitet, om den stämmer överens med verkligheten eller ej, beror på respondenternas förmåga att uppskatta effekten av deras egna reciprocitetsrelaterade impulser. Resultatet skall granskas, med faktumet att respondenterna kan vara olika bra på att bedöma hur mycket de påverkas av plikt och skuldkänslor i efterhand, i åtanke.

Reciprocitetsfaktorn är en som har uppkommit både i den tidigare enkätundersökningen, samt i Benbya & Belbaly (2010). Pliktkänslan som beskrivs av Lakhani & von Hippel (2003) reflekteras i respondenternas svar ovan, och vi kan av detta få en uppfattning av dess inflytande

som motiverande faktor. Av statistiken kan vi avläsa att pliktkänslan är definitivt relevant i de flesta av fallen, och i många fall väldigt relevant, men det är få fall då faktorn är den mest relevanta, eller helt irrelevant. Faktorn kan alltså tänkas ofta agera sekundärt i kombination med andra starkare faktorer. Detta styrks av att medelvärdet, 3,05, indikerar att faktorn är viktig, men inte alltid den viktigaste faktorn som spelar in.

Arv

Figur 4. 8 Svar enkät 2, arv i open source



Medelvärde: 2,30

Standardavvikelse: 1,155

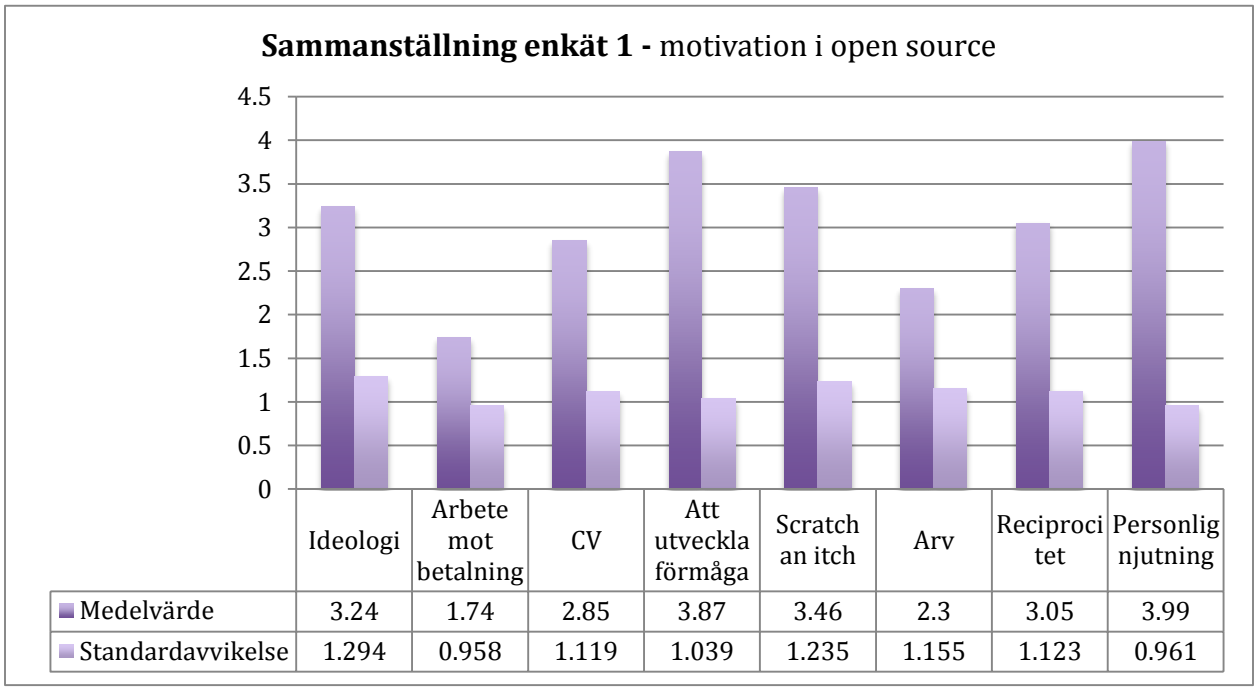
Arv som kategori sågs inte som särskilt viktigt av majoriteten av respondenterna. 31 % ansåg att arv var *oviktigt*, ytterligare 31 % ansåg att arv var *någorlunda viktigt*. Faktorn *viktigt* fick 40 röster motsvarande 20 % vilket lämnade *mycket viktigt* och *viktigast* på 14, respektive 5 procent.

Som vi tidigare nämnt återfanns inte arv-faktorn i någon av den litteratur vi granskat. Inte heller var faktorn frekvent i vår första undersökning. Tolkningen som gjordes i analysen av resultatet av den första undersökningen, att kategorins sällsynthet berodde på en relativt liten grad av viktighet, bekräftas av datan från undersökning nummer två. Medelvärdet på 2,30 understiger 3.

Över 30 % ansåg att faktorn var någorlunda *viktig* eller *oviktig*. Av detta kan vi utläsa att faktorn är av viss relevans, men inte lika viktig för respondenterna som de andra faktorerna.

Sammanfattning

Figur 4. 9 Sammanställning av medelvärden och standardavvikelser enkät 2



	Ideologi	Arbete mot betalning	CV	Att utveckla förmåga	Scratch an itch	Arv	Reciprocitet	Personlig njutning
Medelvärde	3.24	1.74	2.85	3.87	3.46	2.3	3.05	3.99
Standardavvikelse	1.294	0.958	1.119	1.039	1.235	1.155	1.123	0.961

Utifrån motivationsfaktorernas medelvärden kan vi urskilja en hierarki som antyder faktorernas grad av viktighet. Det visar sig att respondenterna ansåg att personlig njutning var den viktigaste faktorn med ett medelvärde på 3,99 vilket landar 0,01 under värdet för kategorin *väldigt viktigt*. Faktorn hade även en väldigt stor andel markeringar som *viktigast.* Efter personlig njutning kommer att utveckla förmåga, med ett medelvärde på 3,87. Scratch an itch följer med 3,46 i medelvärde. Reciprocitet är den slutliga kategorin med över 3 i medelvärde, den hade 3,05 i medelvärde. Under 3 ligger CV med 2,85 i medelvärde, arv med 2,30 och sist arbete mot betalning vars medelvärde ligger på 1,74. Standardavvikelserna för kategorierna skiljer sig inte avsevärt. Den lägsta standardavvikelsen återfinns hos den kategori som även hade lägst medelvärde, arbete mot betalning (0,958). Det skall även noteras att standardavvikelsen för kategorin med högst medelvärde var näst lägst, personlig njutning hade en standardavvikelse på 0,961. Högst standardavvikelse återfanns hos ideologi-faktorn, med en standardavvikelse på 1,294.

Med hjälp av den här datan kan vi göra ett antal konstateranden. Personlig njutning är den faktor som respondenterna värderar som viktigast. Det finns i och med den relativt låga standardavvikelsen en viss enighet bland respondenterna. Med detta kan vi konstatera att resultaten indikerar att personlig njutning är den viktigaste faktorn för respondenterna i undersökningen. Detta kan vara en indikation på att saker som "flow" och att skapandet av bidraget i sig själv medför njutning som tas upp av Baytiyeh & Pfaffman (2010), samt Hars & Ou (2002) påståenden om erkännande av omgivningen är viktiga motivationsfaktorer för bidrag till open source. Vår tolkning av resultatet kan även innebära mer vikt för faktorerna från dessa studier. Det skall dock noteras att erkännande från omgivningen inte var explicit inkluderad i enkätens definition av personlig njutning.

Likt personlig njutning var respondenterna relativt eniga i sitt markerande av arbete mot betalning som en *oviktig* faktor. Detta reflekteras åter igen i svaren från den första enkätundersökningen. Teorin som läggs fram av Hars & Ou (2002) om att försäljning av relaterade produkter och tjänster är en viktig motivationsfaktor kan vi ifrågasätta med resultatet av enkätundersökningen som grund. Det skall dock nämnas, som nämnts tidigare att målgruppen som enkäterna riktades mot lutar generellt sett åt det ideologiska hållet, iväg från det monetära.

Den faktor som var mest polariserande var ideologi-faktorn. Vi kan se en indikation på att open source-gemenskapen är starkt ideologiskt driven, men att en betydande fraktion av gemenskapen inte håller de värderingar som befordras av majoriteten. En annan tolkning av det är att den fraktionen helt enkelt inte identifierar sig med gemenskapen. Sambandet mellan identifikation och värderingar förklaras av Hars & Ou (2002). Det går inte att utläsa från datan om polariseringen beror på att fraktionen inte identifierar sig med gemenskapen eller endast inte håller med om värderingarna. Om de som svarat att de inte såg ideologi som viktigt tillfrågats om de identifierade sig med gemenskapen eller inte skulle Hars & Ou (2002) påstående kunna valideras.

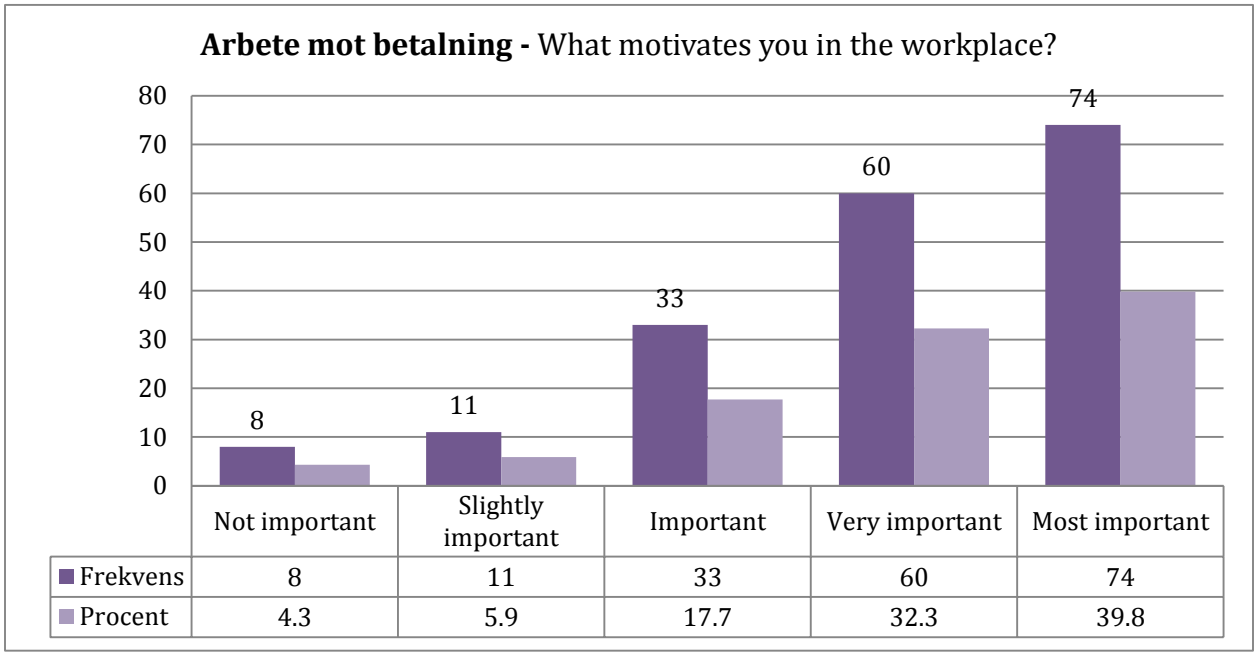
Vi kan baserat på datan från den andra enkätundersökningen konstatera att den "nya" kategorin, arv, som inte återfinns i litteraturgenomgången är en någorlunda viktig faktor. I den första undersökningsansatsen fick vi dels kunskapen om att faktorn existerade, och dels en indikation på att den skulle kunna vara viktig för en del utvecklare. Resultatet av den andra enkätundersökningen bekräftar att arv är en viktig faktor. Med ett medelvärde på 2,30 ligger den väl över gränsen för *någorlunda viktig*. Detta kan vi tolka som att faktorn är av en viss relevans, men kan ses som sekundär till de viktiga motivationsfaktorerna för respondenterna.

4.2.3 Motivation på arbetsplatsen

Likt den föregående frågan är denna avgränsad, men i detta fall till motivation på arbetsplatsen. Vi hoppas delvis kunna undersöka hur motivationsfaktorer ser ut på arbetsplatsen, men huvudsakligen ska undersökningen utnyttjas för att skapa en kontrast mot hur motivation uppfattas i öppen källkods-gemenskapen. Därför kommer analysen av de här svaren inte vara lika omfattande.

Arbete mot betalning

Figur 4. 10 Svar enkät 2, arbete mot betalning på arbetsplatsen



Medelvärde: 3,97

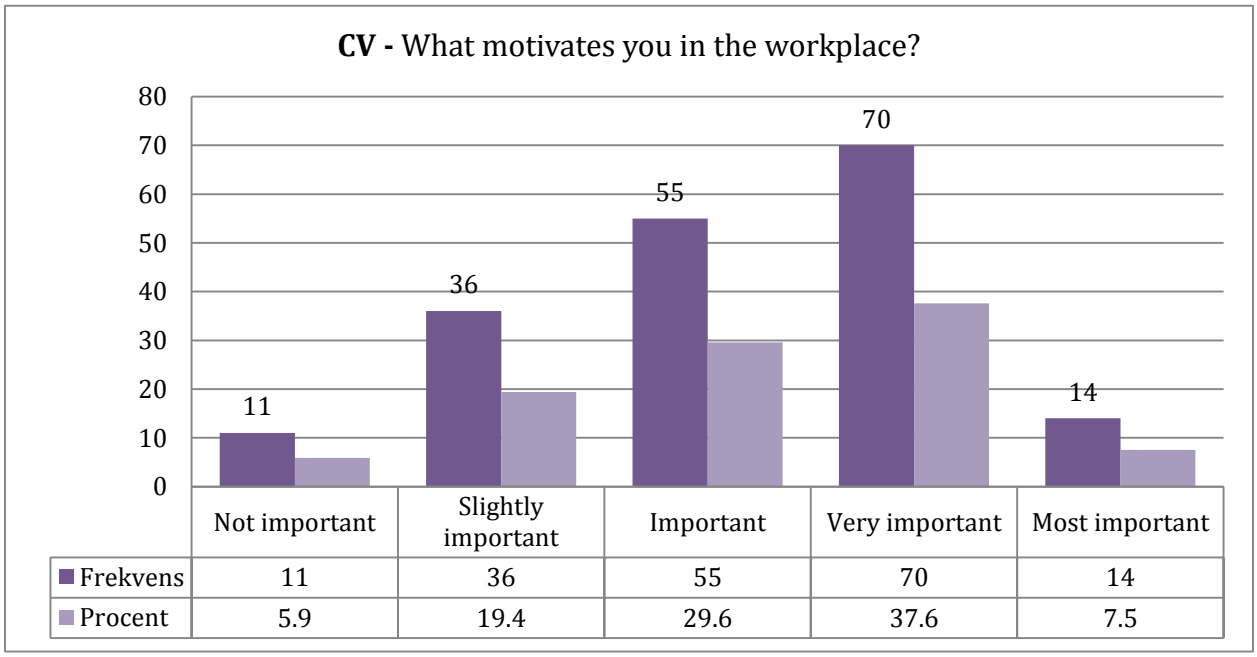
Standardavvikelse: 1,098

Vi kan här se att betalning verkar vara en starkt motiverande faktor för individer när de är anställda, något som kanske inte är så överraskande. 39,8 % av respondenterna satte den som den som *mest viktig*, följt av 32,2 % som hade den som *mycket viktig*. Endast 8 individer satte den som *oviktig* och medelvärdet hamnade på 3,97 av 5, vilket vi kan anse vara relativt högt för den här studien.

Det här resultatet var för oss föga förvånade, att individer arbetar för att få pengar är inte något nytt. Frågan ställs i vår undersökning mest som en referensfråga.

CV

Figur 4. 11 Svar enkät 2, CV på arbetsplatsen



Medelvärde: 3,22

Standardavvikelse: 1,033

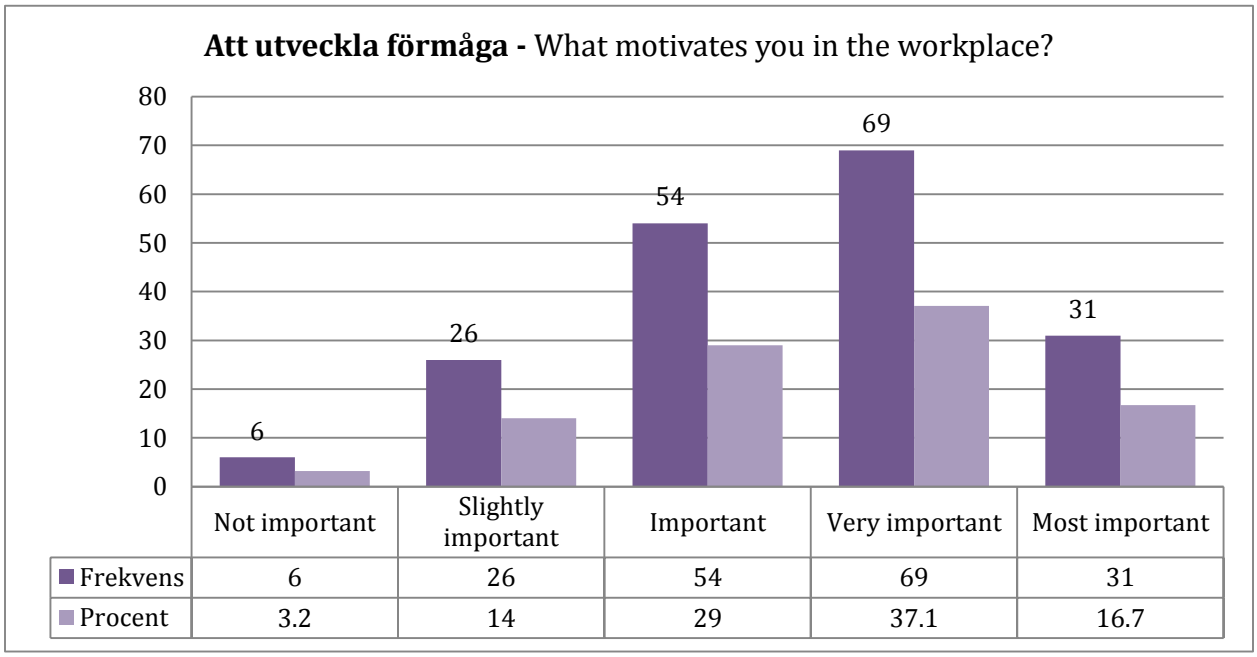
Under den här faktorn svarade majoriteten av respondenterna, 37,6 %, att de anser att förbättra sin CV är *mycket viktigt*, även på arbetsplatsen. Däremot var det få som hade faktorn uppsatt som *mest viktig*, 7,5 %. Många svarade även att den var *viktig* vilket i slutändan gav ett medelvärde på 3,22.

Vad vi utläser från det här är att individer inte slutar vilja bättra på sin resumé även efter att de faktiskt har ett jobb, antagligen för att det finns chans att de i framtiden vill eller blir tvungna att byta arbetsplats. Antagligen kan de genom att ha ett visst jobb få möjligheter till avancemang i framtiden. Rätt jobb och arbetsuppgifter kan signalera att man exempelvis är ambitiös eller kunnig, något vi beskrivit i vår litteraturgenomgång. Förbättring av CV ansågs inte vara det allra viktigaste men inte heller *oviktigt*, sammanlagt 86,6 % valde de tre mittersta graderingarna. Vad

vi utläser från det här är att CV:t är viktigt för många individer, men inte engagerar särskilt mycket, det är inte en polariserande fråga.

Att utveckla förmåga

Figur 4. 12 Svar enkät 2, att utveckla förmåga på arbetsplatsen



Medelvärde: 3,50

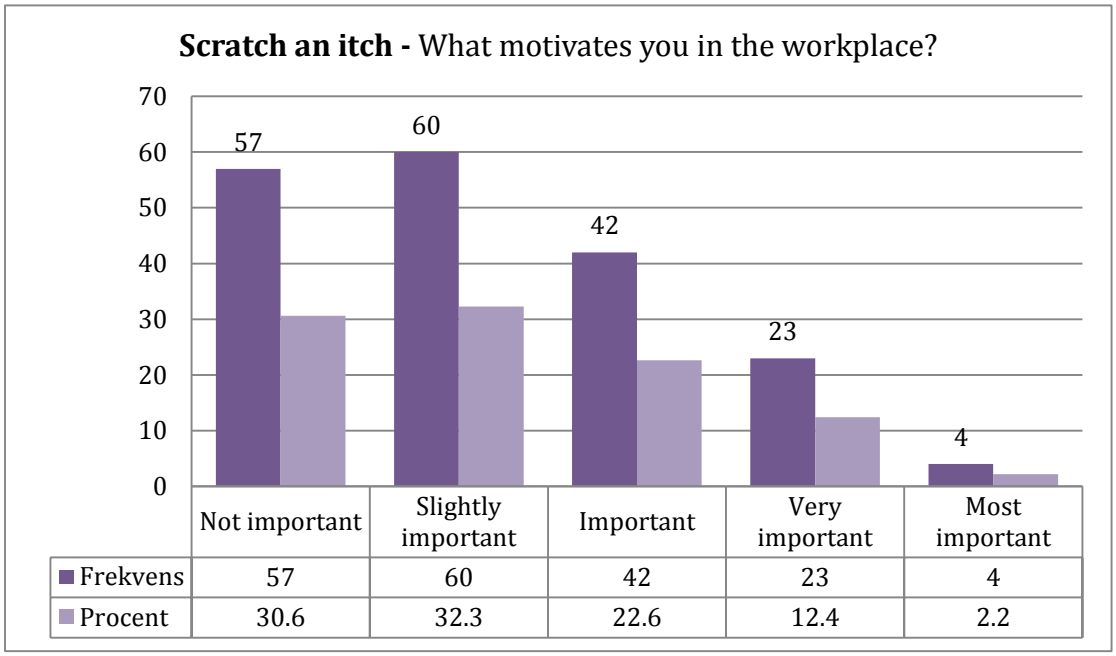
Standardavvikelse: 1,031

En stor andel av respondenterna, sammanlagt 66,1 %, anger att den här faktorn är *viktig* eller *väldigt viktig* för dem på arbetsplatsen. Medelvärdet är även här beläget i det övre skiktet, vilket tyder på att utveckling av förmåga är något som motiverar. Det är även 16,7 % som satt den som *viktigast*, men däremot få som satt den som *oviktig*. Att utveckla förmåga tolkar vi därför som en viktig faktor men inte en som provocerar till starka reaktioner.

Utveckling av förmåga verkar vara en relevant motivationsfaktor även på arbetsplatsen och detta återspelar sig i medelvärdet.

Scratch an itch

Figur 4. 13 Svar enkät 2, scratch an itch på arbetsplatsen



Medelvärde: 2,23

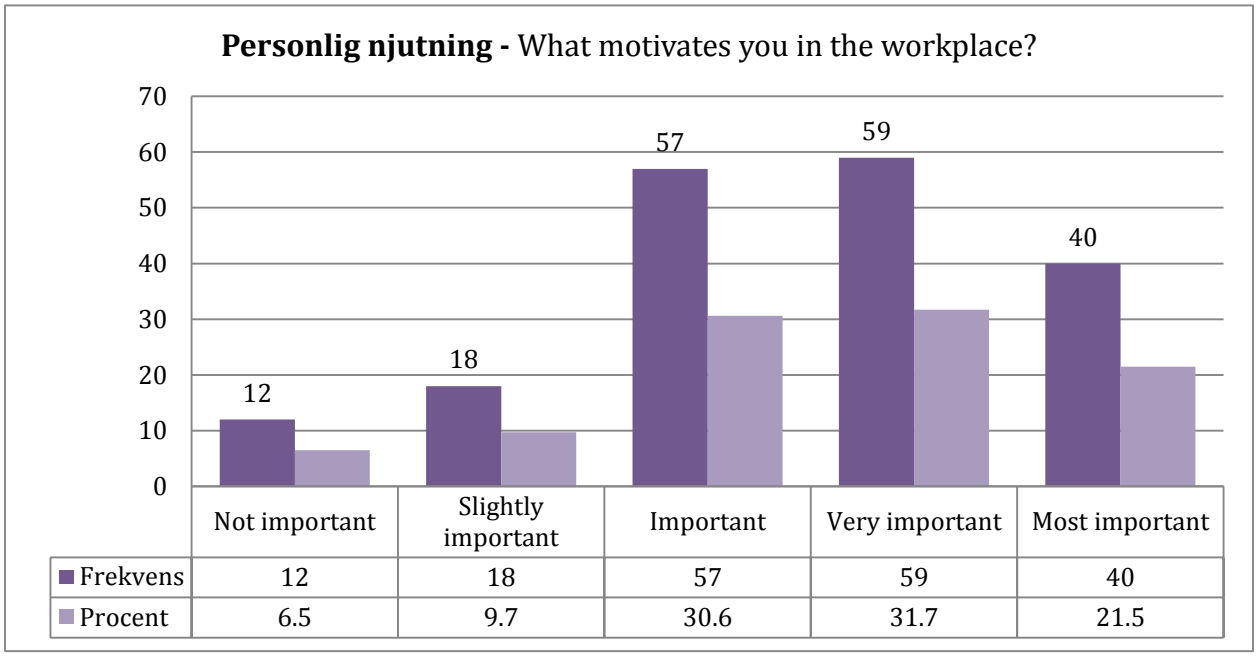
Standardavvikelse: 1,083

Scratch an itch sågs inte som en viktig faktor av respondenterna vilket är tydligt vid närmare inspektion av datan. 30,6 % tyckte inte att faktorn var viktig och 32,3 % såg den som *någorlunda viktig*. Av svaren indikerade 22,6 % att kategorin var *viktig*, 12,4 % tyckte att faktorn var *väldigt viktig* och 2,2 % markerade *viktigast* som svar. Faktorn har ett medelvärde på 2,23 och en standardavvikelse på 1,083.

Medelvärdet är en klar indikation på att faktorn är relevant som motivationsfaktor. Det finns dock en betydande andel, 30,6 %, som anser att faktorn är irrelevant. Detta kan bero på att respondenterna inte är i behov av förbättring av sina existerande mjukvaruverktyg, eller att det inte existerar i deras företagskultur att utveckla sina arbetsverktyg. Av resultatet att döma är faktorn relevant, inte som den viktigaste faktorn, men som en sekundärfaktor i och med bland annat att en större andel (32,3 %) ansåg att den var *någorlunda viktig*.

Personlig njutning

Figur 4. 14 Svar enkät 2, personlig njutning på arbetsplatsen



Medelvärde: 3,52

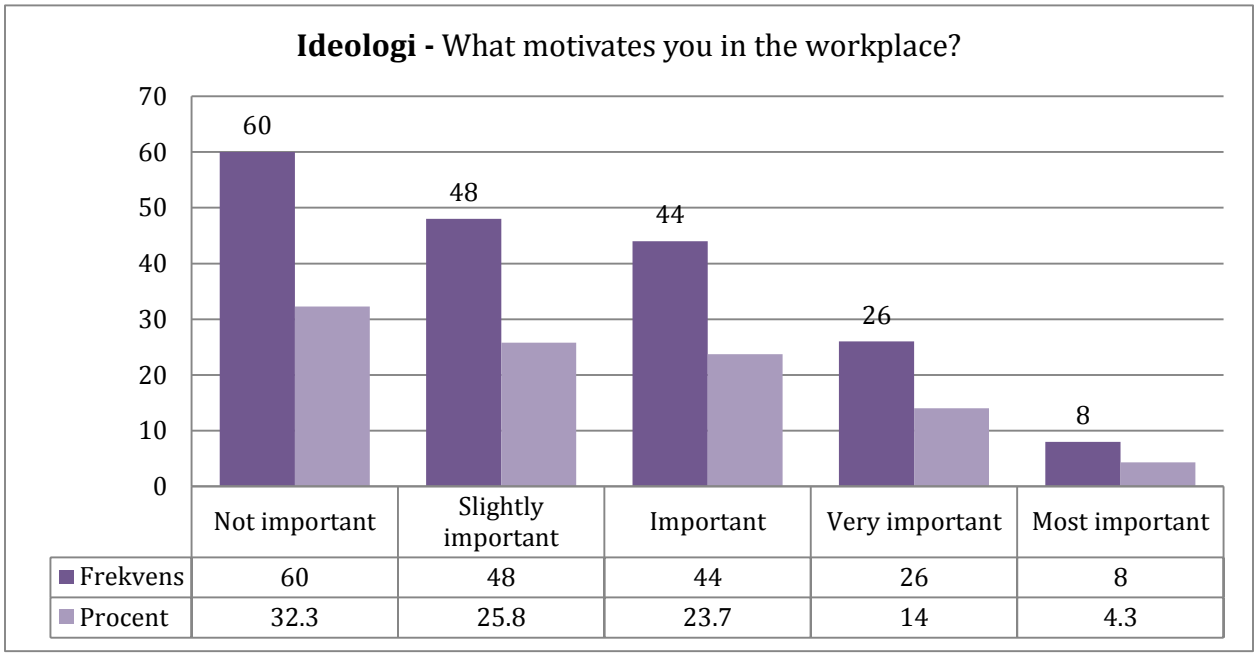
Standardavvikelse: 1,126

Personlig njutning ansågs av respondenterna att vara en viktig motivationsfaktor. 21,5 % av respondenterna ansåg att faktorn var *viktigast* och 31,7 % ansåg att faktorn var *väldigt viktig*. Vidare fick svarsalternativet *viktigt* 30,6 %. Svarsandelarna minskar sedan drastiskt, *någorlunda viktigt* fick 9,7 % och *oviktigt* fick 6,5 %. Medelvärdet för faktorn låg på 3,52.

Av medelvärdet att döma var personlig njutning en faktor som var av stor vikt för respondenternas motivation. Eftersom att svaren är i arbetsplatsens kontext kan detta dock färgas av respondenternas åsikter om sina arbeten. Datans pålitlighet förlitar sig på respondenternas förmåga att särskilja deras tycke för open source från deras tycke för arbetsplatsen. Respondenterna värderade för övrigt faktorn högt, vilket tyder på att personlig njutning är en viktig faktor.

Ideologi

Figur 4. 15 Svar enkät 2, ideologi på arbetsplatsen



Medelvärde: 2,32

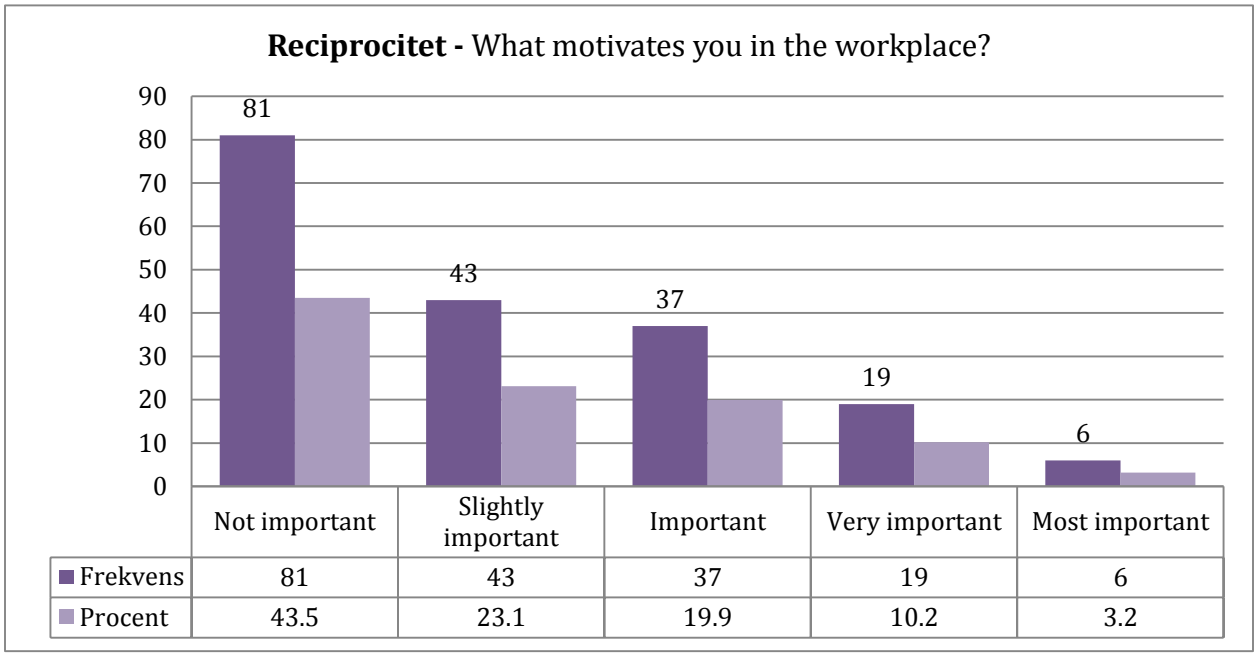
Standardavvikelse: 1,187

Ideologi är till skillnad från i open source här indikerad som en relativt ointressant faktor. Hela 32,3 % angav den som *oviktigt* medan endast 4,3 % *viktigast*. *Väldigt viktigt* ligger också lågt i relation till tidigare resultat, men *viktigt* och *någorlunda viktigt* tillsammans utgör en icke-ointressant 29,5 %.

Satt i relation till de tidigare svaren är en så stor andel *oviktigt* en väldigt stark indikator på en brist på denna faktor som motivation på arbetsplatsen. Det tes uppenbart att företag generellt inte utnyttjar ideologiska motivationer i sin företagskultur, alternativt att anställda sällan känner sig motiverade av ideologi i sitt dagliga arbete.

Reciprocitet

Figur 4. 16 Svar enkät 2, reciprocitet på arbetsplatsen



Medelvärde: 2,06

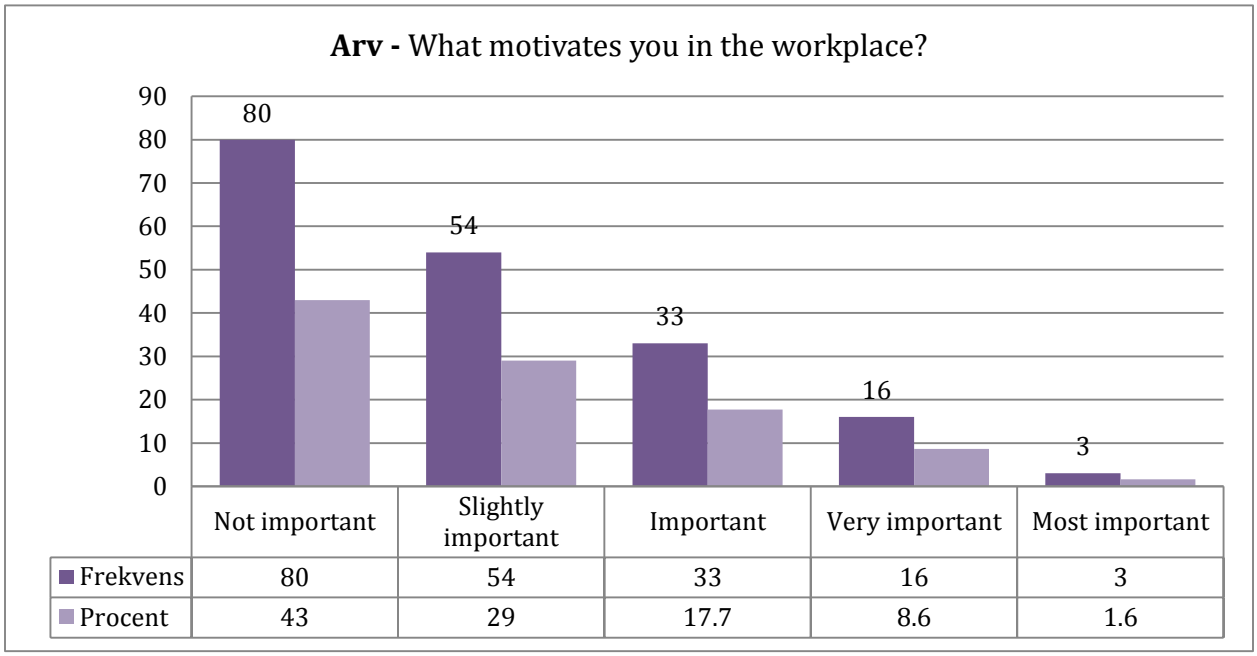
Standardavvikelse: 1,156

Reciprocitet anser respondenterna vara av viss vikt och betydelse. 3,2 % hävdade att faktorn var *viktigast* och 10,2 % markerade reciprocitet som *väldigt viktig.* Svarsalternativet *viktig* fick 19,9 % av markeringarna och *någorlunda viktigt* fick 23,1 %. För utom detta ser vi att 43,5 % av svaren markerade reciprocitets-faktorn som *oviktig.* Medelvärdet för reciprocitetskategorin var 2,06 vilket är 0,06 över värdet för svarsalternativet *någorlunda viktigt.*

En stor del, 43,5 %, uppfattade inte reciprocitet som viktigt över huvud taget. Detta kan bero på hur deras arbete är upplagt, om organisationen är strikt hierarkisk, och arbetsflödet generellt sett endast går i en riktning finns inget utrymme för de sociala utbytesförhållanden som diskuteras av Benbya & Belbaly (2010). Trots den stora andelen markeringar som *oviktigt* pekar medelvärdet på att faktorn ändå är av vikt för en del utvecklare, dock av varierande grad av betydelse, som vi kan observera från spridningen av andelen svar över alternativen *viktigt, väldigt viktigt* och *viktigast.*

Arv

Figur 4. 17 Svar enkät 2, arv på arbetsplatsen



Medelvärde: 1,97

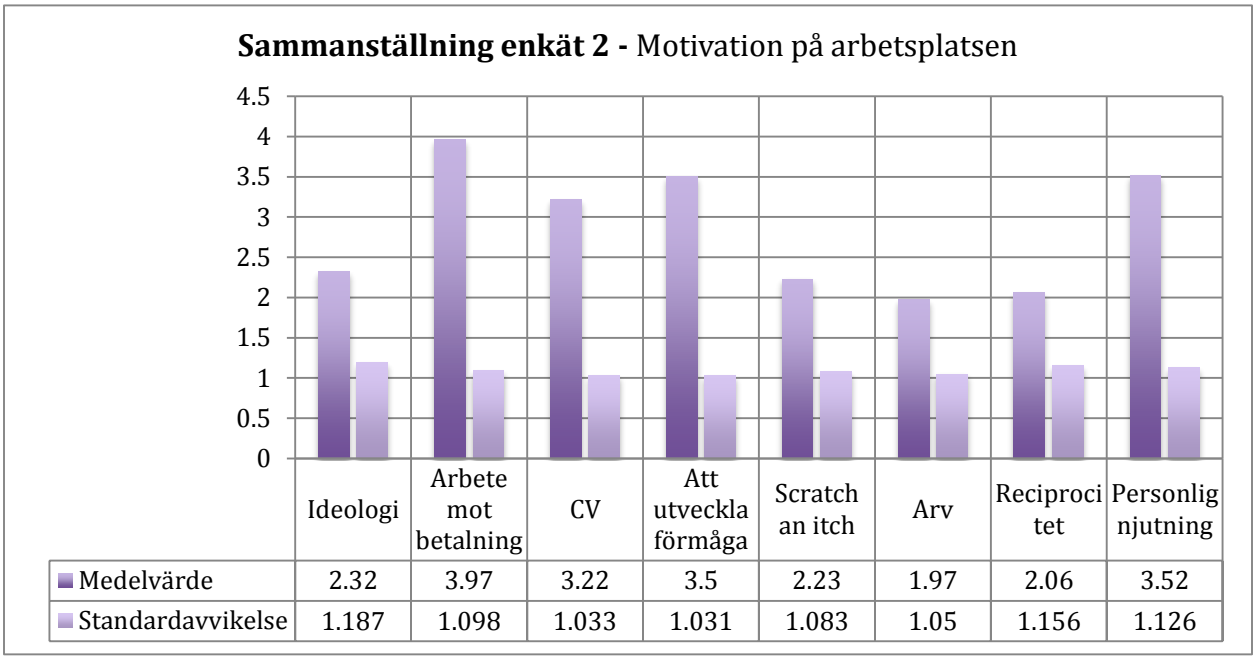
Standardavvikelse: 1,050

Arv som motiverande faktor uppfattades som *oviktig* av 43,0 % av respondenterna och endast 1,6 % uppfattade den som *viktigast*. 29,0 % av respondenterna ansåg att faktorn var *någorlunda viktig*, andelen markeringar minskar sedan successivt med 17,7 % markeringar för *viktigt* och 8,6 % av markeringarna för *väldigt viktigt.*

Att en stor andel, 43 %, svarar att arv-faktorn är irrelevant på arbetsplatsen visar på en klar polarisering vad gäller om faktorn ska ses som relevant eller inte. Trots den stora andelen av negativa svar indikerar medelvärdet på 1.97 att faktorn är någorlunda relevant för en del utvecklare, men att den inte ses som en av de primära motivationsfaktorerna. Baserat på detta kan vi anta att faktorns viktighet kan bero mycket på individen, en del individer kan kasta den helt åt sidan medan andra ger den en viss vikt.

Sammanfattning

Figur 4. 18 Sammanställning av medelvärde och standardavvikelse enkät 2, på arbetsplatsen



	Ideologi	Arbete mot betalning	CV	Att utveckla förmåga	Scratch an itch	Arv	Reciprocitet	Personlig njutning
Medelvärde	2.32	3.97	3.22	3.5	2.23	1.97	2.06	3.52
Standardavvikelse	1.187	1.098	1.033	1.031	1.083	1.05	1.156	1.126

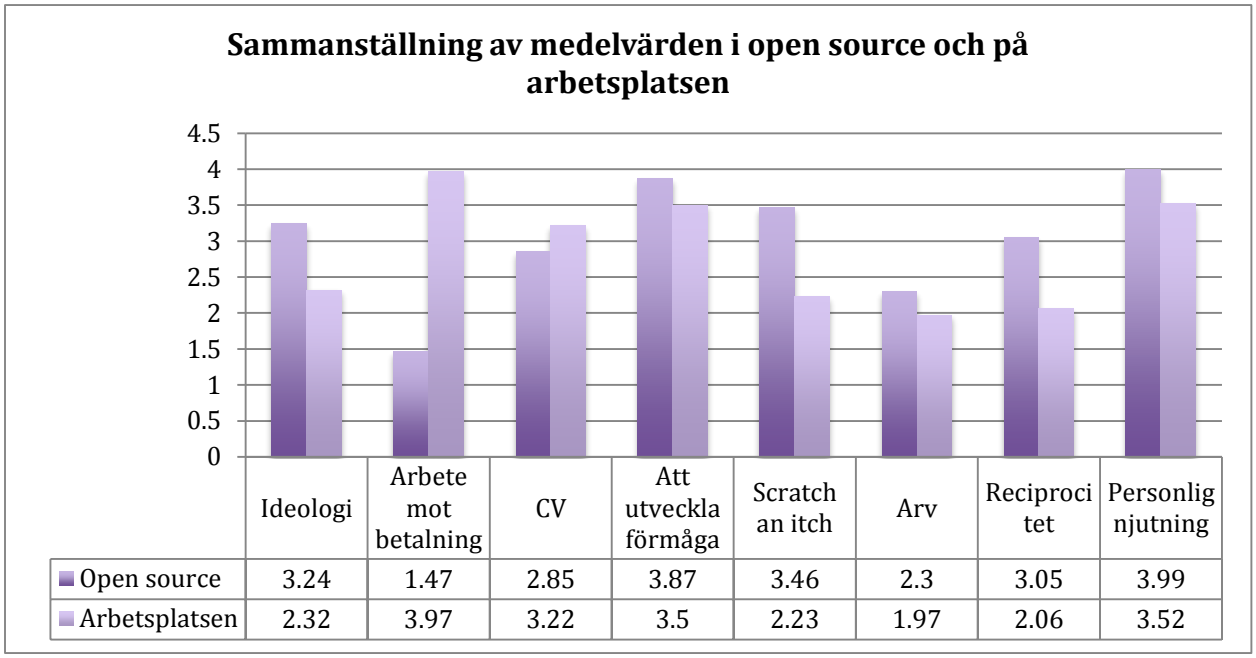
Också här kan vi urskilja en hierarki av svar utifrån medelvärdena vi tagit fram, vilka agerar som indikatorer på uppfattad vikt hos svarsgruppen (se figur 4.18). Arbete mot betalning ligger högst på denna skala, vilket var förväntat. Även om en analys av betalning som incitament ligger utanför vår studies omfång är den fortfarande intressant som en referenspunkt. Vi ser att de tre faktorer som kommer närmast betalning i viktighetsgrad är CV, utveckling av förmåga och personlig njutning. Vi kan alltså konstatera att dessa tre, tillsammans med betalning, är de primära motivationsfaktorerna på arbetsplatsen.

I en något lägre, men inte negligerbar, utsträckning anses ideologi, arv och reciprocitet motivera på arbetsplatsen. I vår skala och i jämförelse med de andra utslagen är dessa synnerligen lågt värderade.

Värt att notera är att standardavvikelsen är lägre på denna fråga än den föregående. Åsikterna på vad som motiverar på arbetsplatsen skiljer sig inte i samma grad mot varandra som i open source, vilket kan tolkas som en brist på engagemang. Det krävs en högre grad av ansträngning för att ta sig in i och medverka i öppenkällkodsgemenskapen, då det ofta görs utöver en traditionell arbetsplats.

4.2.4 Jämförelse av motivation på arbetsplatsen och i open source

Figur 4. 19 Jämförelse av medelvärden i open source och på arbetsplatsen, enkät 2



	Ideologi	Arbete mot betalning	CV	Att utveckla förmåga	Scratch an itch	Arv	Reciprocitet	Personlig njutning
Open source	3.24	1.47	2.85	3.87	3.46	2.3	3.05	3.99
Arbetsplatsen	2.32	3.97	3.22	3.5	2.23	1.97	2.06	3.52

När faktorernas medelvärden i respektive miljöer, open source-gemenskapen och arbetsplatsen, jämförs kan vi se en hel del distinkta skillnader.

Ideologi graderades som viktigare av respondenterna i kontexten av open source-gemenskapen och mindre viktigt i kontexten av arbetsplatsen. Detta kan tolkas som att open source-gemenskapen har en starkare kultur än företagen. Om Hars & Ou (2002) påstående om att individer som identifierar sig med open source-gemenskapen riktar sina värderingar därefter kan appliceras på en företagskultur, skulle detta kunna tyda på att open source-kulturen är starkare än de flesta företagskulturerna. Alternativt skulle det kunna innebära att individer har lättare för att identifiera sig med open source-kulturen än med en företagskultur.

Sammanställningen av datan från den andra enkätundersökningen pekar på en distinkt skillnad mellan betalning som motivationsfaktor på arbetsplatsen och i open source-gemenskapen. Denna skillnad kommer förmodligen inte som en förvåning, det är förmodligen ett relativt säkert antagande att de flesta går till sin arbetsplats för att tjäna sitt uppehälle i monetär kompensation. Open source-gemenskapen kan även inkorporera detta, vilket vi har delvis sett i svar från den första enkätundersökningen, men sannolikheten kan tänkas vara större att utvecklaren skulle vara motiverad av pengar på arbetsplatsen än i open source-gemenskapen.

Under faktorn scratch an itch kan vi utläsa att respondenterna såg faktorn som viktigare i open source-gemenskapens kontext än arbetsplatsens. En tolkning av detta skulle kunna vara att arbetsplatserna förser till större grad utvecklarna med de verktyg som de behöver, och därför finns det mindre utrymme och behov för improvisation och egen-utveckling. Detta skulle stämma överens med Bonaccorsi & Rossi (2010) poäng om missnöjet med brister i existerande kommersiell och icke-kommersiell mjukvara, om vi kan göra antagandet att företag har generellt sett större tillgång till, alternativt resurser att spendera på, mjukvara. Detta skulle kunna vara en förklaring till att scratch an itch fått lägre andel markeringar i företagsmiljön.

Skillnaderna i andelarna relaterade till arv faktorn i open source och arbetsplats-kontext var relativt små. Datan visar att faktorn är mer relevant i open source-gemenskapen än i företagsmiljön. Detta skulle kunna bero på att arvet har större överlevnadschans i open source-gemenskapen jämfört med företaget, om man tar i beräkning att företaget kan till exempel byta mjukvaruplattformar eller gå i konkurs.

Reciprocitet visade sig enligt datan från den andra enkätundersökningen vara viktigare i kontexten av open source-gemenskapen än arbetsplatsen. Medelvärdet för reciprocitet i open source-gemenskapen låg över 3 medan medelvärdet för arbetsplatsen låg strax över två. Skillnaden kan, som tidigare alluderats, bero på att strukturerna i de olika miljöerna kan vara mer eller mindre fördelaktiga för de sociala utbytesförhållandena som beskrivs av Benbya & Belbaly (2010) och som står som utgångspunkt för reciprocitet.

Personlig njutning är en faktor som fått höga svarsandelar i båda sammanhang. När vi jämför dem ser vi att personlig njutning som motiveringsfaktor är något lägre för arbetsplatsen än för open source-gemenskapen. Skillnaden skulle kunna tolkas i relation till motiveringsfaktorn *arbete mot betalning.* Respondenterna värderar betalning högre än njutning i arbetsplatsen, men de värderar njutning högre än betalning i open source-gemenskapen. Detta kan åter igen bero på skillnader i miljön och under vilka premisser som individerna interagerar med miljöerna. Open source-miljön kan vara mer attraktiv för intresse-programmerare. Vi kan även tolka resultatet som att det ändå är viktigt med gillandet för det egna arbete, men att denna faktor blir ännu viktigare när betalningsincitamentet inte finns vid sidan. Vi kan alltså baserat på datan se att fenomen som “flow” och “peer recognition” som diskuteras av Benbya & Belbaly (2010) och Hars & Ou (2002) är relevanta i både open source-gemenskapen och arbetsplatsen.

5 Slutsatser

Vi har kartlagt vad som motiverar individer i sin medverkan i open source och skapat en rangordning av dessa faktorer som också visar dem i relation till motivation på arbetsplatsen. Vi har visat att motivation skiljer sig på arbetsplats och i open source. Medan faktorer som personlig njutning ses som lika motiverande i de två skilda världarna syns en övertygande skillnad i uppfattningen av vikten av betalning och ideologi som drivkraft. Open source-gemenskapen är djupt engagerad i sin sak och detta återspeglas i motivationsfaktorerna, med ett starkt fokus på ideologi. Den här ideologiska motivationen känner få av svarspersonerna på sin arbetsplats. Över lag lägger respondenterna större kraft på sina svar när det gäller deras medverkan i open source, jämfört med deras motivation i det dagliga arbetslivet. Resultaten från enkät två indikerar på att det finns en övergripande brist på motivation på arbetsplatsen när den ställs i jämförelse med det livfulla och innovativa öppenkällkodsfenomenet. För dessa individer är gemenskaperna en plats av större motivation och engagemang.

I analysen av vår första enkät identifierade vi en faktor som fattats i den tidigare forskning vi tagit del av i kapitel 2. Behovet och viljan att lämna något efter sig är en grundläggande mänsklig drift, och återfinns alltså också i mjukvaruindustrin. Medan den inte hade en så polariserande effekt som frågor om betalning och ideologi ansåg en stor andel av respondenterna att den existerade som motivationsfaktor i open source. Detta såg vi i mycket mindre grad på arbetsplatsen, något vi tolkar som en brist på igenkännande. På en arbetsplats är det svårare att vinna beröm och uppskattning och att få sätta sitt namn på något, vilket öppenkällkodsgemenskapen generellt sätter stor vikt på. Alla i gemenskapen vet vem Linus Torvalds är, inte för att han tjänade pengar eller var en duktig PR-person, utan för att han skapade och delade med sig av något som alla haft nytta av. Vi misstänker att en högre grad av erkännande för individers arbete kan leda till en högre motivationsnivå för dessa.

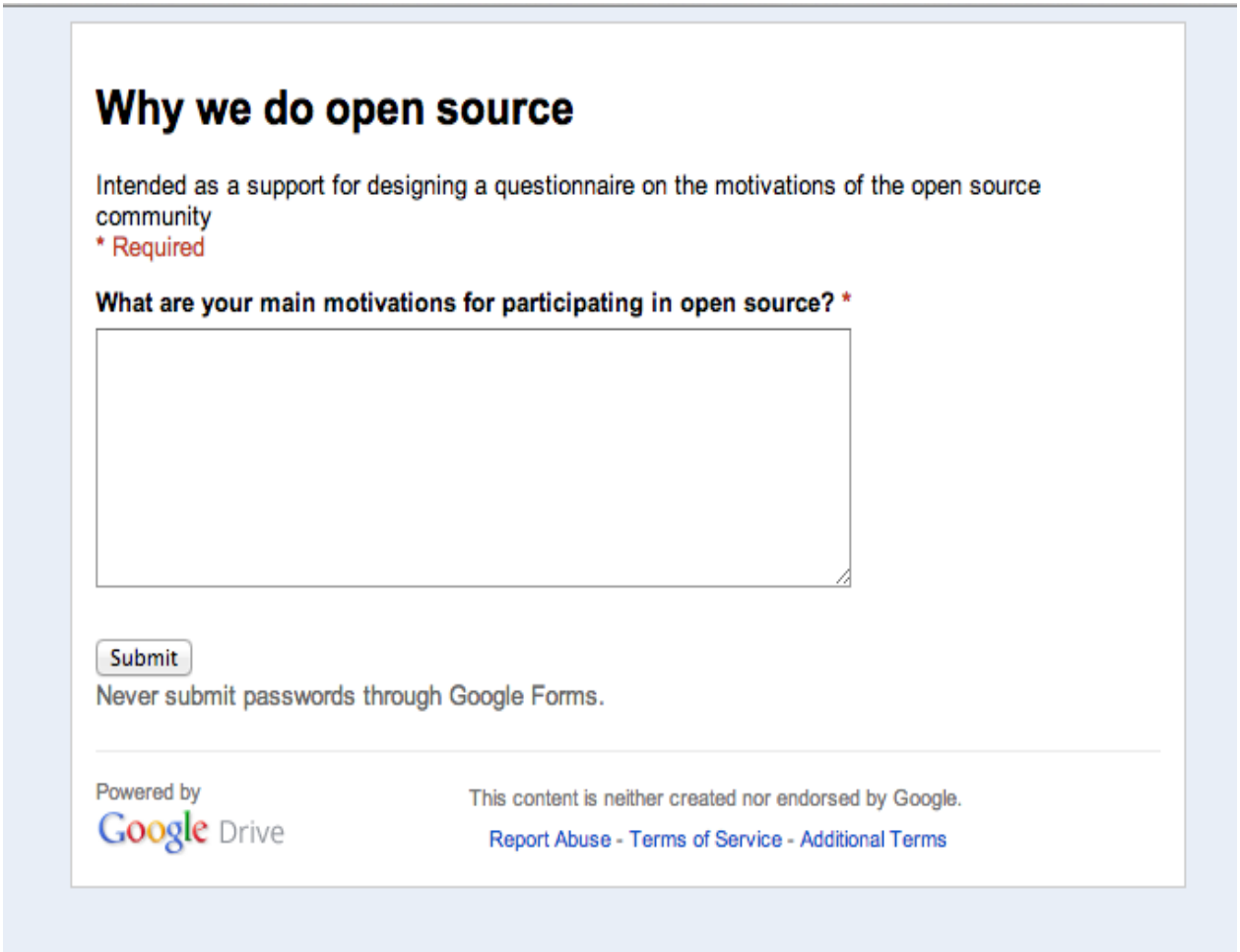
Den överlägset mest kraftfulla motivationsfaktorn verkar vara ideologi och en stor andel av svaren rangordnade denna faktor i de högsta nivåerna, *väldigt viktigt* och *viktigast*. Detta stämmer väl överens med vår analys av svaren från enkät ett, där ideologi var bland de vanligaste motivationsfaktorerna som nämndes. Individerna visade också ett enormt engagemang och politisk medvetenhet i sina argument och förklaringar. Ideologi som drivkraft är otroligt stark men samtidigt följs den av en naturlig intolerans mot många vanliga aspekter av företagskultur, som slutenhet och fokus på vinst, som också indikeras av svaren från enkät ett. I svarsdatan från enkät två ser vi hur svarsgruppen motiveras lika lite av ideologi på arbetsplatsen som de gör mycket i gemenskapen. Detta kan vara för att arbetsplatserna inte tar vara på de anställdas ideologiska engagemang eller för att individerna själva lägger sina ideologiska åsikter åt sidan.

Vi har svårt att dra generaliserbara slutsatser för öppenkällkodsgemenskapen i helhet då vi har begränsad insikt i vår respondentgrupp, utöver att de besöker sidor som *Hacker News* och *Reddit*. Vi ser dock goda möjligheter att reproducera resultaten, då vår metod är öppen och enkel. Vi upptäckte också ett problem i vår svårighet i att bedöma huruvida individernas svar påverkats av om de identifierat sig med open source-gemenskapen eller inte. Synnerligen tydligt är detta i respondenternas svar på till vilken grad ideologi motiverar dem i öppenkällkodsarbetet. Även om en stor andel svarat att det är en av de viktigaste motivationsfaktorerna finns en tydlig subgrupp med rakt motsatt åsikt (se figur 4.6). Hars & Ou (2002) fann att individer som är delaktiga i en gemenskap har en tendens att ta på sig dennas åsikter. Denna konflikt i teori och empiri misstänker vi kan vara orsakad av att en del av svarsgruppen medverkar i, men ej känner sig som en del av, öppenkällkodsgemenskapen.

Motivationsfaktorer är endast en av de många aspekter som skiljer den moderna arbetsplatsen från öppenkällkodsgemenskapen; för att få en helhetsbild och en fullständig förståelse av orsakerna till fenomenets framgång krävs fortfarande mycket forskning. Från utvecklingsmetodik till gruppdynamik, organisationsteori och human-computer interaction, ett stort antal fält täcks in i problemområdet. Vår studie visar att det finns begränsningar i den tidigare forskning som gjorts, främst kanske i hur målgrupper valts. För att ge tydligt generaliserbara slutsatser krävs en mycket större och öppnare respondentgrupp än de som kännetecknat studier i området. Fenomenet vi studerat är en konstant växande kraft i branschen och vidrör alla, forskare och yrkesverksamma. Vår studie är endast på en begränsad aspekt av ett ofantligt komplext fenomen, där alla delar måste beskrivas innan tydligare slutsatser går att dra.

6 Bilagor

6.1 Bilaga 1 - Enkät 1



Why we do open source

Intended as a support for designing a questionnaire on the motivations of the open source community

* Required

What are your main motivations for participating in open source? *

Submit

Never submit passwords through Google Forms.

Powered by Google Drive

This content is neither created nor endorsed by Google.

Report Abuse - Terms of Service - Additional Terms

8.2 Bilaga 2 - Fullständiga svar enkät 1

Rad	What are your main motivations for participating in open source?
1	Professional * Publish fixes about bugs discovered in open source products we use in our company (enhance the products we use, and expect that our fixes would be maintain through version) * As a tribute to the community without we cannot do our job - we use a lot of open source products for infrastructure, code development, etc. and we (as developers) think that we are in debt to that community Personnal * To show my personnal projects to my friends * To allow others to take experience from mine * Because I believe in free (as freedom) idiom
2	I don't care for open source at all. Free software on the other hand... The foremost reason is idealism. I like the concept of being part of creating something that everyone can use. My way of trying to better society. A secondary motivation is that I use a whole lot of free software daily, it would be rude not to contribute back.
3	My users care whether the application is open or closed source Managing repository and all that goes along with it looks great on resume Open source + free gets more users (resume, again) Only reason to go closed source would be to monetise with ads/in app purchases, which would only make a negligible amount of money
4	Just about everything I learned about coding, I learned by hacking on code that someone else made available. I release free software to give back to the community that made me what I am.
5	Because having open source software available enriches all our lives. I also consider "computing technology" to be something too important to be constrained entirely by the profit motive and/or purely corporate interests. Technology is our common human heritage, not merely a vehicle for shareholders to get rich. Having technology available which is decoupled from the profit motive is a huge boon to humanity and enables progress to be made, the kind of progress that just can't happen if you need to worry about per-core licensing costs for your scientific compute-cluster.
6	Because it is the perfect way for a code to make the world a better place by increasing human knowledge and tools.
7	Because i get to do what i want to. In a world where everything is controlled for me, i get to be in total control. I can run windowmaker and only log in as root if i want or i can run openSUSE KDE and be able to instantly configure everything ever with a semi handicapped user account. I get to change everything and if i don't like something i can change it, in so far as i can change the code that makes it happen. In addition, i can directly connect/help/support the real people that work on the software, not a large no-face corporation. The best part of open source though is that it is up to me as a user to support it's growth. I am given responsibility. I know that i must donate to the GIMP project if i use it, rely on it, and need it. I am not "forced" to support a project i don't want because i am pre-charged for it with the purchase of a computer. I am treated as a responsible adult and people tend to live up the way they are treated.
8	All operating systems suck, why not use a free one? Seriously though, open source means that I can fix bugs that annoy me in projects that I use and control

	how my system is built from the ground up. It means that I don't need to be strapped down to one vendor for software and I can become the vendor for software that I use regularly. It all boils down to control. I can set the rules by which my computer works and that makes it less sucky.
9	To "scratch an itch" that is, to fix bugs or add features I desire. To be able to hack scripts together to work in a highly automated, optimised workflow that I feel helps me get my job done. To read and learn more about software development of large projects.
10	Why I use it: Because I enjoy understanding how every aspect of my computer works. Why I contribute code: Because things were broken and realized I could fix them.
11	I see no reason to not release the code I write. I can see how releasing billion dollar secret sauce algorithms would be stupid for commercial companies, but the code I write? I want to learn and teach, to help the world move forward with software. [borttagen kontaktuppgift]
12	Because resources are easily available on the net. Community support is strong. Because it somehow has a cool factor to it.
13	* help other people * get industry recognition pretty much the same reason why we blog
14	Learning technologies I don't get to learn in School, while teaching others at the same time. To get to work on some piece of technology or field I'm interested in by showing I'm qualified. (This has already happened once.)
15	I participate in open source, cause I want to present all the people with soft- and hardware which gives them the four essential freedoms[0]: * The freedom to run the program, for any purpose (freedom 0). * The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this. * The freedom to redistribute copies so you can help your neighbor (freedom 2). * The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this. I believe this is a fundamental part in giving personal, social and political freedom to all people. [0] http://www.gnu.org/philosophy/free-sw.html
16	Personal gratification: it's cool to see people use code I wrote Education: I learn a lot from hacking on open source projects, especially when my code is reviewed, discussed, or improved upon Career: most of the good interviews I've had in the last few years have started with "I had a look at your

	github". Reputation: the best way to establish yourself as a leader in some field is to contribute to a major open-source project
17	Moving the (computer) world to the next level. Freedom. Knowledge.
18	- I like coding - I like showing others what I can do - My resume - Some basic computer tools are just better developed as a community effort (programming languages, system tools, internet daemons, etc.)
19	So I'm employable.
20	I open source my code to give something back, because I also use a lot of open source code. I open source my code so I can learn from what other people are doing with it. I also open source to show my code as part of my portfolio, so potential employers can check it out.
21	altruism, symbiosis, ego and profit. -- altruism: it feels good to help. plus, there are times you solve a problem that others are likely to encounter and want to save them the grief. -- symbiosis: open source technologies make it easier to create our projects. to ensure these resources dont go away, we contribute back to them. -- ego: it is quite an accomplishment to have a large number of other devs use your work. you also want to show your knowledge on a given subject. -- profit: there is money in providing support to open source technologies. it also helps strengthen trust in a service.
22	You get out what you put in. While much of the ecosystem is funded by large companies, and having a take-only attitude is fine, the community still requires people to give up time and effort to code, support and document. Despite dramas playing out on mailing lists, IRC and such, most projects are run by decent, fun people. Open source contributions also give you an unparalleled opportunity for learning in the real world, whether you want to learn software engineering, technical writing, management skills, graphic design or any number of things.
23	Because it promotes sharing of knowledge. It gives you a sense of positive feeling about yourself, that you are doing a good deed. As students, we would have struggled to find good sources to study well written programs, or have benefited from reading sources from open source projects. A part of us will always want to do something in return. You get recognition, not just for the beauty of the building, but also for the way you have laid the bricks
24	I'm part of open-source community, so, I must contribute how ever I can, almost all the software that I use is open-source, I want better and better versions of it, and if nobody contributes, this will never happen..
25	I believe that everybody should have the right to analyze and modify the software that she uses.

	Ultimately, this leads to better software and thus is helpful for humanity as a whole.
26	I open source my own components/projects if I think there's some generic utility to them. I like the idea that people around the world could potentially be using my ideas and code. In some ways I also treat it as a way to learn. If a project gets some serious attention, pull requests start coming in with good practices that I didn't know about. It takes some balls to do it though, and makes me nervous every time. As for contributing to others' projects — I mainly do it if I'm using a component and find some bugs/improvements to make. Mainly for moral reasons I guess.
27	- Interesting projects - collaboration with different people - much more interesting than daily work ;D - great community
28	Because knowledge must be free (as in speach) and accessible to everyone. I our world, code is law, not controling the code is not controling your world. I cannot tolerate that my code could be used to reduce the freedom on someone else.
29	freedom. open source is just a natural consequence of free (as in speech) software.
30	I publish open source code: - To be able to use the modules and applications I created for other purposes: at other employers, for myself, etc.. - To get respect from other developers. - To get better jobs due to my repositories at GitHub. I submit bug reports and send patches to fix and advance software I am using or intend to use. I promote and encourage others to use open source and particularly free software, because I think it is the only model that respects the rights and freedoms of software users (including myself) while being fair to its creators.
31	Because marketing is hard. I enjoy writing software- but I have no desire to jump through the hoops required to sell it. It's way easier for me to be like "here is this thing. Use it. Don't. Whatever" and then go back to doing what I love this way.
32	The barrier for entry into many open source projects is low. This means that you can work as part of a team, working on cool projects, without having to pass a job interview. So the feeling of belonging to a team is quite powerful. When I was freelancing i felt the need to 'give back' to project I was making a good income from. This helped me also understand the technologies better. These days, it is possibly closer to the fact that the area I am working is quite innovative - part of the inner circle of emerging technologies - with proprietary software in the back seat. Being in the inner circle, we have people watching from the outside and interested in the directions we go in.

	Open source has allowed me to travel the world, work from home and work with some of the smartest people in the industry - with low barriers between employers.... and importantly, receive a good salary. I don't think I could work in closed source now, I feel quite lucky.
33	To be quite honest, I *started* working open source because Red Hat offered the opportunity to work on a project I'd wanted to do for years, and have it get greater exposure than I could get on my own, for good pay. Nothing idealistic about it. That said, I will probably *continue* doing open source because I've come to believe that collaborating with others through an open-source community is an eminently practical way to make something better faster. Secondarily, I believe that it's nearly impossible to do meaningful innovative work in computing without drawing heavily on the "intellectual commons" of open source, and once one does that there is a moral imperative to give back. None of this precludes my working on closed source again some day, but it makes that a much less appealing alternative.
34	Close Source development means that everyone (every company) is re-inventing the wheel again and again. Much development cost and time is wasted. Also, if some company dies, all the work might be lost. I don't like the thought that one just totally waste the time and work of humans. Open Source is working together with each one to advance the whole human race. Nothing is wasted. The whole human community benefits. This is much more effective and makes more sense because you don't waste so much work resources for no reason. It's also much more social.
35	Transparency. I was amazed with how much it gave me. And by giving back, I am giving others what I received and more.
36	Usually if I have found some use from Open Source software then I try to contribute something back
37	Because I can "use my powers for good" with open source: something I do can potentially benefit anyone on the world. Additionally, it used to be the way I could have the most impact on the world by participating in open source projects. Now that I'm no longer a kid, I'm not sure whether that's still true, but it often feels that way.
38	To solve a problem that humanity has once, and well, so nobody else need ever solve it again, and the code is freely available for all to use and modify.
39	Feeling like I'm part of something bigger than me and my own personal code. I won't try to rationalise it, because it's pretty much an emotional response, but I enjoy the feeling of other people using my code, because it means I've had a positive influence on the world.
40	Learning from awesome people, getting my code reviewed, helping others and doing something significant.
41	* It lets me improve my personal programing skill seeing others way of doing thing * It lets me have the control of the software I use, and thus fixing them if they're broken * Normally community are made of very interesting people
42	If you buy a car, and it breaks, you are free to take it to any car mechanic to have it fixed -- you are not forced to go back the one mechanic who sold you the car.

	If you hire an architect and construction company to build a house, and years later you want to add a garage to your house you can hire a different architect and construction company -- you do not have to go back to the group of people who originally built your house. I believe software should work the same way -- if you purchase software from me, you should have the freedom to hire any interested software developer to make further changes to that software. You should not be forced to hire me to do the changes, just because I built the first version of that software. Proprietary software is an easy way for software companies to create a vendor lock-in situation, which I think is abusive.
43	(for self-created projects) - Opening the source of things I think may be useful for other people, either for usage or learning. (for existing projects) - Help the project fix bugs or add other features (giving back to something that has been useful to me)
44	Writing open source software is one of the easiest ways (among blood and sperm donation) to change many other's lifes for the better.
45	I was able to learn development thanks to open source and the first full time development job I had was at a company that used an almost all open source stack. Imagine some poor kid wandering into a university library to try and get a library card, being given a computer login and having their entire direction in life changed just because someone was altruistic enough at some point to make their source code available and free to use.
46	I publish all the software that I develop as open source where I think it might be useful to someone else. I usually contribute to other open source projects when I find bugs or miss a feature, and it's reasonably easy to fix/implement.
47	Because I believe that computer science, much like other science, progresses more quickly when we share our data and can use the results of others in our own projects. We learn more. We grow moe. Moreover, data interchange and open specifications should be at the center of every government's IT strategy.
48	Fortune & Fame
49	* Fixing things - This is also my motivation for USING open-source-software. When something goes wrong, the ability to find and fix the problem without dependence on a third-party is incredibly important. * Security / Wisdom of Crowds - (popular) public code is reviewed by thousands of eyes. Problems are found fast. Solutions are discussed in public. * Self-Improvement - Nothing makes you write better code than knowing someone better than you will likely be reviewing it at some point. * Self-Validation - It feels good to solve problems. It feels even better to have other people acknowledge and extend your solutions.
50	Pragmatism and generosity. When I have a problem to solve, one of my first thoughts is usually "Someone has already solved this." So I go looking. Much of the time, there's an open source package that either does exactly what I want, or can be easily

	tweaked to do what I want. If that's the case and the license is compatible with my current project, I use it. Then, when I have to fix a bug or add a feature, it only makes sense to push it back upstream, so that upgrades don't mean painful merges. If I can't find a module that does what I want, I'm left with writing it. If my new module remains closed, then only me (and any teammates) can possibly fix any problems or add any features. If I open source it, though, other people with the same problem can find my module, use it, and fix bugs/add features. So, if others like my code, I may just get fixes and features for free. It's a win/win, and it helps others. What's not to like? (I do think some code should remain closed - if you're writing something for a business, code that directly deals with trade secrets or that directly implements high-value features is one of your competitive advantages. As you'll note, I didn't mention ideology as one of the reasons I contribute to open source.)
51	1. To produce better quality software via studying, modifying or remixing earlier works. i.e. applying something resembling the scientific method to engineering. 2. To avoid the possibility of being alienated from my own work, as can occur with "work for hire" 3. To produce software which anyone can use, regardless of their financial or geographical situation.
52	So I can write code I'm proud of. Everything I've released has been carefully crafted so it's a work of beauty, relatively speaking. I don't get to do that much for my usual contract work (yes, I'm one of those developers - sorry).
53	I contribute to projects I use myself for two reasons. 1) To give something back to a project which has helped me. 2) To add features and fix bugs which I have a need of. By open sourcing the fixes I do not have to maintain an own branch of the code.
54	I'm not a much of a developer, but I use OSS in my work extensively and I've shared the scripts and simple tools when I think they'll be useful to someone else. I use because: A) I feel much more confident supporting something which I know I can dig into and observe the function of. This can give me and understand of what I'm doing wrong, allow me to work around it or possibly patch it entirely. Contrast this with the helplessness of troubleshooting the black box of closed software. B) It's cost effective. Money not spent on expensive licenses and implied support contracts can be applied elsewhere.

	Since much of my work deals with public funds, I feel particularly obligated to maximize value. C) I feel what I learn using OSS is more transferable, and applicable to other systems and problems than closed software. I contribute because: A) I think sharing discovery and tools and building on the work of others is key to our success as a species. It feels completely natural. B) It opens the door for symbiotic improvements. Personally, I seem to be pretty good at find a novel perspective on a problem, but have little knowledge of how to code it efficiently. Sharing my code with someone who knows how to do those things benefits us both. C) To learn by practice and example.
55	I enjoy programming and like to program in my free time. Because it is my hobby I like to share my work with other people who have similar interests. I also feel that my skills improve when I read the code written by developers who are more talented that I am. I hope my code can also contribute to the knowledge of others. Finally, programming at work involves compromises due to time or the beliefs of others. Writing open source code in a small project is a creative outlet that involves no such compromises.
56	To fix bugs in the software I use daily.
57	The biggest reason I love open source is being able to look at how something works. My main motivations for releasing my own software as open source is to further enable that for others. There's also some fringe benefits, like drive-by-contributions. I've made some good friends with strangers who contributed to one of my projects. I contribute to others' software mainly to make it more suitable to myself and others. I still get a little rush when someone pulls in my changes.
58	love developing and when I do it on free time want it to be public to possibly get recognition or credit for it
59	* Show that I enjoy what I'm doing. * Showing the world what I know. I've been getting job offers based on my open source contributions, as well as my own open source projects. * Improve tools that I am using in production/at work. * If I'm simply making something to test out, why not make it open source? * Bigger Open Source projects tend to higher code quality, many times better documentation, a thriving community that is usually way better than a general support if I have questions. The community can also usually be reached through a channel _I choose_ (stack overflow, mailinglist, IRC...). Not to mention that open source tend to be very much cheaper.
60	Making the world a better place.
61	Openness! If everything were open source there would be a lot less trouble in learning/doing programming, everything you could think of is accessible and reusable, with the appropriate license.
62	Synergy, maintenance, returning back to the community, transparency and credibility in front of the clients. It's really interesting to see that you're writing about OS - I will be writing my Master Thesis on the topic of CSR and Open Source (business major).

	You can drop me a line at redacted so we can potentially share resources and help each other; there are very few academic resources useful for writing the thesis on this topic. Cheers and good luck
63	I learned a lot in my youth from using Linux systems and being able to really delve into all the internals - no hidden magic, everything available to take a look if you wanted to. As a teenager, I used to have print-outs of large parts of the core of the Linux kernel, of X Windows, and so on. This availability of knowledge, but also the freedom that comes from free software systems, are important values for me. I want to contribute to such systems being viable, modern environments. A secondary motivation is that for purely hobby projects, there is a chance of finding collaborators that make the project more fun to work on and advance more quickly.
64	I write open source code, because that means that others can benefit in more ways than just having the result. It can also mean, that more dedicated developers can take over maintaining the piece of code I wrote without me having to worry about it. I write bug reports and feature requests mostly for personal gain, that the software I use performs better. I suggest using open source software to others for it to gain more support in general. This will ensure that the project gets kept alive. I prefer using open source software to closed source ones, because to me that means that I can use it for a long time. If development halts, someone else can take over. Even if that does not happen, perhaps a bunch of small patches here and there keep it compiling and running on newer systems. Back-tracking a route of out-dated libraries can be tiresome, especially if some architectural reason prevents one dependency to run on the new system. Further more I believe that the source code will be kept cleaner, because the developer knows that literally everyone can read the code (although most never will it is still a strong incentive). Additionally peer review from outsiders can help spotting bugs. Of course, there is less pressure than having to support the code for a signed contract, which means that proprietary software might make more use of CI suits, but it stands to question whether most proprietary software does indeed use proper procedures, while with open source software you can usually review the process. And if many people of the community see the project going the wrong way, they can fork it and attempt to do a better job. Something that is very hard with proprietary software.
65	I try to open source because I don't know of a better way. Oinksoft is a small shop right now and I'm more likely to get help with what I'm working on by sharing the components with others. Even the largest, most capable companies cannot compete with the talent of the open source world at large. Also, it also forces me to author reusable, developer-friendly software components. I have a lot of stuff in the pipeline that I haven't gotten to writing all tests for and releasing yet, designing for open source distribution has made the packages far more cohesive and usable.
66	To help create a common infrastructure that helps to advance our civilization's tech level in a manner that is available to everyone everywhere. Also, because it's fun.
67	- I'm ideological in the sense that I believe the greater human calling is to collaborate on producing works

	of excellence, driven by passion and the purpose of advancing our collective state, as opposed to always competing at pumping out crap and pointless products and manipulative advertising to claim the day's market share and siphon profits. - I don't only want to believe in the ideals behind open source but practise them, and not in private because the essential point of it all is that we're working in the same space, visible to each other, offering criticism and encouragement, and gathering around to help when someone comes up with a big idea. - there is something I actually find more humble and honest about people who write in open source, as if they're saying yep here's some of the work I do, maybe it's not of the highest quality, and it's not all intended or even suited for making money, but I'm practising and having fun anyway, --- as opposed to keeping their code private as if it is all too genius to share with anyone. reticence and the reluctance to share are often only safeguards for a pretence, unless you are waiting to perfect your work in the shadows before you make the grand debut. all of that is contrived: --- if you intend to be part of a community, then just get your work into the open, flaws and all, introduce yourself, and go from there. -- redacted
68	To learn from others, get in contact with interesting smart people and to reuse existing things instead of wasting my time on rebuilding something already done a hundred times.
69	To build my reputation and to support a community from which I benefit on a daily basis. Probably a bit of altruism to it too.
70	In my opinion, *the* economic problem of the 21st-century is *convexity*. Go here for further reading: http://michaelochurch.wordpress.com/2013/04/03/gervais-macleod-21-why-does-work-suck/ . I have a whole slew of posts on economic topics that I've put together over the past 2 months. The effect of convexity is a movement from low-risk commodity work to risk-intrinsic work (i.e. its value is purely in its natural uncertainty, and not discomfort as with commodity labor) that is more fun and creative, but harder to make a living from. The overarching theme of economics from 1975 to now has been the offloading of commodity work to machines, who can do it more reliably. The hard and somewhat creative stuff, like programming these machines, is what's left for humans. (Actually, there's a lot of concave grunt work in coding; the goal is to have it done by programs called *compilers* that, again, can do that concave/boring stuff way better than we.) So the only thing left for humans is the risk-intrinsic convex stuff. I've been studying Convexity for years and it is a big f_king social problem. Convex work has high expectancy but *lots* of risk and it can't put forward the mediocre, regular income that average people are used to (and could not stand to be without). Software is the leading edge of convex economics. It's the first battlefield between the old industrial regime (everyone gets a mediocre wage for mediocre work) and Convexity. With convexity, you have a long learning period before your earning period (in which, ideally, you keep learning). During that (poorly paid or unpaid, often) learning period, you build a lot of cool stuff. Eventually, you get to a level of expertise and product-quality that people will pay for it; but it's impossible to know (until you engage directly with the market) how close you are to that point. So you end up building lots of still-quite-cool (if not professional-grade) stuff in which there's no harm in giving it away for free. As programmers, we don't really fear people "stealing our ideas"; ideas are cheap and easy; code is hard. That's part of why we pretty much unanimously hate software patents. Convexity creates risk for workers (because it makes full employment uncommon if not untenable) but also for institutions that need to hire people. Talent discovery is a massive problem. (I'm writing on that right

	now.) Companies have to pay about $10,000 to hire a 1.2-level (scale here: http://michaelochurch.wordpress.com/2012/01/26/the-trajectory-of-a-software-engineer-and-where-it-all-goes-wrong/) programmer (interview time/opportunity cost) and about $40,000 to hire a 1.5 (add recruiting fees, hiring bonuses, perks and more interviewing). How much does it cost to hire a 2.2+? (Sometimes you need that level of talent.) Almost *a million* (mostly in R&D budgets to build an autonomy culture.) The only places where 2.2+ programmers want to work are companies with extremely high levels of autonomy where they can do 2.*3*+ level R&D work; no managerial meddling, full autonomy over time, plenty of resources. (The 2.2+ "repay" that $500-700k by doing excellent work, of course.) The reason Valve has an open allocation culture and Google used to have one (before ~2009) is that, even if it's "expensive" to hire engineers and have them doing R&D with full autonomy, that's what you have to do if you want to hire the (rare and picky) 2.2+. So talent discovery is a hard problem, and there's a bilateral matching problem wherein companies find it really hard to hire the people they want, and even good engineers have significant job latency. The bilateral matching problem only gets *worse* as you move up the skill curve. (I'm a 1.8 and have had 3-month job searches; then again, I'm really picky.) The old way for undiscovered (usu. young) talent to gate-crash this discovery problem was to pay 50 thousand goddamn dollars per year to some institution already sitting on billions, for 4 years, and then get discovered by another billionaire corporation, climb its organizational ladder doing low-paid boring work, and eventually show (around age 35) that you're ready to do real work (after 10+ mind-numbing years of order-taking that killed your creativity). Well, that's just too inefficient to hold water (get it? convexity, holding water? bad joke. anyway...) anymore. Traditionally, convex labor with extreme talent-discovery problems (such as Hollywood acting and scientific research, where the natural talent us rare) fell to gatekeepers (literary agents, Ivy League admissions officers, graduate departments, athletic recruiters) who were imperfect and sometimes corrupt, but did the job. The problem with software is that no one is qualified to serve this "middleman" role because only an equal or superior engineer can judge another software engineer's work. So, it's not just that it's desirable to get around these middling agents; it's that the only people who can really do the job are software engineers, who'd rather write code. So the new way to fix the talent-discovery problem seems to be to create this giant, amazing gift economy that we call "open source" (and blogging, print-and-play games, also are part of this). You participate because (a) it's more fun to build cool stuff and give it away than to spend 75+ percent of your time selling and <25 creating, and (b) it's a great way to overcome the talent-discovery problem. So that's why we give cool stuff away for free.
71	Three reasons. 1. To keep a good thing going. As a developer, early all my tools, from my editor to my language to the libraries I use and the web server my code runs on, are open source and free. I love that! I love that there's no cost to start a project and that I can tweak the open source code as I like. I want to keep that going, so I contribute back. 2. Reputation. Open source projects show off my abilities and help my career. There are few things as satisfying in work as getting the respect of those I respect. Open source provides a wider field for doing that than code that's only seen by others in my company.

	3. Keeping my work. If I solve a problem well at one job, it would be irritating to have to solve it all over again at the next. If I made my solution open-source, I can reuse it.
72	My software is another medium for my art. I thought apps/software would be the best way to get my art seen by the most people, and used to help them out or perform whatever function they desire. Then I realized that a better way of proliferating my art was to do so through open source, that way other developers use my art to reach exponentially more people than my model could.
73	Professional reputation.
74	Give back to the developers who helped me get to where I am today!
75	Participating in open source projects as a developer looks good on the CV.
76	To learn, to contribute back to open-source tools that I use, and for self-motivation
77	In part, the free software paradigm: everybody knows what they're getting, nobody has political motivations to impede development, software can evolve in a robust way which benefits from the entire community's experience. There is also a political motivation within the corporate software paradigm, in that accepted contributions to popular open source initiatives increase the perceived value of a programmer to employers. Lastly there is a kind of natural clarity to developing out in the open, even if no one is watching your work. It just feels good. Subtle, but powerful to the psychology of many developers.
78	It has probably been said in this very thread before but here goes nothing. Here is why I made some of my own code open. Some of it was college stuff. I wrote assignment programs and made them open so others can see it. (I am almost the only one in entire class who can write code that does not suck) I made it open so others can understand. One other project I made open was because it was a practice code and I wanted to show it off and maybe (again) help others who want to do something similar. I also made it open because I didn't intend to work on it anymore. One time job, so if someone else finds it of any use, they can commit to the main repo and improve the source repo instead of keeping the changes to themselves. And in the likely case that no one commits anything (no one has, yet) I wanted it to be somewhat of a demo code in case I wanted to show my skills to someone quick or impress someone.
79	I've always had a desire to help my fellow human beings. In this case, I tend to build mostly developer tools, and love to help other developers learn the craft and perfect their trade. Gives me a sense of accomplishment that is more difficult to find in companies within the industry outside of a few evangelical posts.
80	Open source is less buggy. If something is open source, it has more people working on it all the time, which usually results in less bugs and a better overall project.
81	I learned how to do software looking at big FOSS projects, now i have my own projects as a way to give back to the community.
82	I use a lot of open source software on a daily basis, my whole company is based on the work of others. (web development) This is my way of saying "thank you" and giving something back, hoping someone else finds it useful and can do something amazing with it. As a result it is a good reputation gain and I lose nothing. I can only encourage everyone to try it once or twice.
83	Since Eric S. Raymond wrote Cathedral and the Bazaar in the 90s, it's been clear that (often) contributing

	code with an Open Source license results in lower development costs in the long run because you don't have the overhead of maintaining your own private fork. For example... Lets say you pick up a piece of Open Source software, implement it and make some bug fixes and new features along the way. By contributing those bug fixes you avoid the overhead of having to merge your own bug fixes and features into the code base again and again as you pull in other bug fixes and features from other developers. By releasing your bug fixes and features into the open source code base you have a competitive advantage over your competitors who maintain a private fork of the same software. Open Source is not charity. It's profit driven. For those that say "it's the right thing to do," either they're doing it for the competitive advantage mentioned above without realizing it or they might just be wasting their time releasing useless open source code. That's not to say I don't think releasing your code is often the right thing to do, I'm CTO for a nonprofit bringing Open Source to Open Educational Resources (http://ole.org) and collaborative development in farming (http://farmhack.net), I'm clearly no crazy capitalist but we use open source for the competitive advantage because as a nonprofit, why not do the best you can do? Open source rocks. Thank you to the giants who pioneered Open Source.
84	I like to help other people. I feel useful doing open source. When I'm gone my work may live after me!
85	It's freedom from dependance on proprietary developers who widely only care about how much money they can separate from their users, among many other reasons.
86	Because I want to be in control of the software I use, not the developer. I want software that respects my freedom to share it and use it.
87	Because it allows the potential for some great software to be created.
88	Open sourcing my software allows me to take part in a worldwide collaborative effort to improve how we interact with technology.
89	Contributing back to a community which helped me develop my own skills as I was starting out.
90	code for fun
91	Because otherwise it would have been sitting on my hard drive collecting dust. Also, programs are like poems. You want to show people your writing sometimes. Some code is better to contribute to the commons. This isn't a zero-sum game. The fact that we have libraries of free and open code to handle all sorts of routine functions benefits everyone. This is basically a reason why any business open sources code. When Facebook figured out better ways to handle MapReduce they open sourced it. The community now maintains it instead of them alone. It saves costs and establishes their reputation, and they're not in the business of selling MapReduce software anyway.
92	1) I make source to share. Even when I make it for myself I want to share. 2) I want to share because I'm proud of my realization 3) I want to be famous 4) It is part of my portfolio. Very significant for job search.
93	Contributing to open source projects provides me with an irreplaceable education in community, communication, compromise and code
94	I love coding and my own projects I get involved with are more interesting than the what I work on at work. I also like learning new skills and programming languages.
95	Unless I believe there's money to be made, I don't see why I would do all the work myself.
96	Philosophical: I'm helping a community produce good software instead of a company create more money. I think it's the only sensible way to develop software.

	Balance: My computer runs almost exclusively Open Source Software and this is my way of giving back.
97	Because it's fun and everyone profits from it. It also means lower dependence on companies, which don't necessarily do things for the greater good.
98	To improve the world and make high quality software for everyone to use. Also get to know people in this industry, and maybe make a new for myself.
99	There are 2 different approaches of 'open source': There are paid positions, such as in Linux (80percent of all contributions are paid or by individuals who are employed to do so) or the RedHat operating system. I am however not employed at such a company (yet!) I do work on open source because - it's fun to train your brain - you solve your very own problem/itch you have. - it's better for a greater good. I could decide to not open source my stuff, but then I will be the only one to maintain and extend it. If open-sourcing however, people will pick up and improve.
100	It depends on what I'm doing. If I'm doing translations it's because I feel like I'm doing good for humanity, and my grandmother appreciates that her desktop is translated. If I'm writing my own project and releasing it as Open Source it is part to build my portfolio and part that if anyone actually found anything I've done useful I will be the happiest of all. If I'm submitting code to an existing project it's mostly to solve an issue I'm having with the project.
101	Easy, I work with a piece of software that has been developed for the past 10 years plus, and I'm still finding little quirks with it. To give you an example, the other day I had a very busy system that would randomly throw a single error, that may or may not have been related to the problem. Because the program was open source, in order to find out more info all I had to do was open up the source file, '/' for the error that was being thrown, and I could see the conditions that caused the error. Fixed the problem, didn't even need to go outside for a smoke.
102	Because I truly believe in the FLOSS values and model. For me, the ideology behind FLOSS is far bigger than the software industry, FLOSS has created an economy where quality and the community's best comes before profit for the individual in the whole process. Back to the software industry: as a system developer I try to use as much FLOSS libraries as possible, not only because I believe in FLOSS, but because I those are best. Using small proprietary products is especially risky: How long will it be supported? What does the code really do, does it anything fishy? Will the price increase? For large proprietary products the risks above are less prominent, but here rises new problems since a large project probably will be an important part of my project: What if I encounter a bug that the owner doesn't fix? What if they limit any of the features I need? What if I need functionality they refuse to implement? I the end find the quality of FLOSS to be vastly better, I can often use the software gratis, and if anything happens, I can myself develop new features, fix bugs or such, or I can pay someone else to do it (maybe

	even the original developer!).
103	it's because I feel like I'm doing good for humanity.
104	Software freedom. Don't support a lot of the restrictions on propriety software, in terms of how I can use the software and anything that tries to restrict the products I can use in conjunction with that software (eg, apple store).
105	Open source and free (as in freedom) software is about giving users control over the software they use, as opposed to proprietary software, which controls its users.
106	Sharing and being seen (exposure). To some extent learning the trade and how open source works (administration, technology, etctera). Experience and developing skill within the relevant project (programming language, security, architecture, design...), and learning from others.
107	* I like programming for fun. * I can fix a problem in a program that I use and share it with people easily. * It's a great way to get some real-world experience for your resume. * I find open-source software in general is just better; because it's usually made by people that use it for the people that use it. * No exorbitant licensing fees or legal hoola-hoops to jump through.
108	The community. I feel, even though the various licenses invoked these days really drive away open source development, there's still a huge community of people who respect the work that goes into making software, no matter the importance. Do I participate in it? Every so often. If I see something to patch, or provide an additional feature on, I'll post to the GitHub. But, I also know my limits and know what I want to work on, so I'm not going to dive right into the Linux kernel and start providing patches that do nothing but inflate my ego. Open source is a very big market that many businesses are not focusing on. A prime example is Red Hat. The distro is released to the public, but the support is what they pay for. Businesses would rather a company support their own product than have to figure things out for themselves.
109	If I need or want something myself, and make it, what's the harm in sharing?
110	I believe that the computer should be controlled by its user. The open source and libre communities works towards that goal of creating software that does what's users want. Either by proxy (i.e. allowing third party programmings to fork the project and add very specific features) or by the user themselves, e.g. the extensibility of Emacs.
111	Initially, my limited budget as a student made me buy a half-off laptop running a fringe flavor of Linux. Initially all the free stuff was fun but eventually, the value of freedom became more apparent. My contribution is mostly in the way of testing and translation (primarily on launchpad).
112	It gives me an opportunity to learn and grow my skills. In my job, deadlines, management and politics often limit how good the software I create can be, but within open-source projects, usually the only limit is how much time and effort I want to invest.
113	I hope that open source will have a big enough impact on the world to make sure that in the future, information, knowledge (and software) will be free and development will not be monopolized. I would like to have a sustainable future, a world where people get credit for the things they accomplish and *not* just for how much power and money they (unrightfully) dare claim for themselves. When we live in a world where it is dogmatic that information is released for *free* - released to everyone equally - just for the credit one gets for inventing something innovative, then (and possibly *only* then) we

	can be sure that our future is bright and fair. Shortly, I just want to live a good life in a good world, and to achieve this I find myself morally obligated to support the righteous cause of open-sourcing all the things.
114	I develop open source software for many reasons: * Programming is fun * I get to interact with smart interesting people * There is sometimes software that I want that no one else will make or buy but there may be others who will use it if it's available so why not * I love being able to fix an issue with the program I'm using rather than have to go through tech support and wait a release to get the issue fixed * Lots of other reasons PS. What sort of college lets you do this as your thesis ? It seems overly simple and more importantly non-technical. Unless your doing a philosophy major or something like that in which case I apologize :)
115	I have some projects I'm passionate about, and find that contributing my time and effort in the form of code is more effective and gratifying than contributing money. If the scale of the change is right, I'm able to directly _bring into being_ the changes I'd like to see in the project.
116	I see installing a software in your computer just as hosting a stranger in your home. If he's going to be so close to me and with access to all my personal stuff, I want to know exactly who he is and what are his intentions. Everyone who is not willing to disclose such information is not eligible to share my space. This is where open source really shines for me, and I do everything within my reach to support it.
117	We chose an open source solution for our latest IT project as the area in which we are developing (automated lecture capture) is fairly new. After running a pilot programme we had a good idea of what needs our new system had to meet. We ran a vendor exorcise to see if any off-the-shelf product could meet these needs but found they fell short in areas that were critical to the project. We did find an open source solution that could meet the essential requirements of the project with some development effort on our part. This is the long and short of the decision, only the open source solution could be customised (quickly) to meet our users' needs. All commercial options would have required us to sacrifice the user experience in ways we were unwilling to do. The educational market moves very rapidly and open source solutions tend to lend themselves to agile software development practices that support what is often user-driven change. A secondary element of the decision to become involved in an open source project was cost. Our 5-year total cost of ownership (including all staffing and hardware) put the open source solution cost at 50% of the commercial offerings.
118	Sharing is caring :)
119	I get satisfaction from providing for my computing needs without spending more than I have to. Initially my adoption of GNU tools professionally was about enabling development without having to purchase expensive development suites from the vendors. Now it is a combination of frugality and laziness.
120	Because it is fun and give the the feeling of doing the right thing.
121	In rank order: 1. Ego. Or, the glory. 2. It gives you influence in a project you would not have otherwise, and all software I contribute to is software I use myself. 3. A way of socialising. Met lots of cool people that way, though this became apparent later. 4. Improve my programming skill. Most of my contributions are in the way of translation and testing

	actually, though I have some rudimentary programming skill. However, I don't enjoy programming much so this is not a huge priority, but I'm always learning, even as I'm writing this, and this is one step closer to greater knowledge.
122	Because it is fun, gives publicity, and you get to interract with a lot of skilled programmers.
123	I believe we are collectively smarter than we are individually. By allowing the wider community to work together we create better programs. I relish the opportunity to control my system. Closed source products like Windows and Mac don't let me do that. They aren't fun anymore. I believe transparency = security. If something is implemented poorly, the community is able to tell. While I as an individual may not be able to detect poorly designed code, I trust someone else will.
124	Because I hate the idea of being locked into using closed-source software to get things done faster. On a closed source software stack, they will never make certain improvements to their software that would actually make your job easier. Sometimes they purposely make your job harder so that you then have to pay for training. And, they often want to charge you money based on the number of users or servers. This is not conducive to scaling up and growing according to your business' needs. On an open-source software stack, if something comes up that makes life easier, then it can be adopted and supported quickly. No cost per user / server means it can be spread throughout the whole organization with nothing to worry about, and can be used as part of a scaling solution for massive needs. Best of all, you often get multiple large organizations contributing to the codebase, since they face problems that make your own issues look small in comparison, and this causes open-source software to have *more* features and *more* code quality than any proprietary solution. Another less-measurable but still excellent benefit about the open-source world is interoperability. Closed-source software is oftentimes explicitly designed to be vertically-integrated and inoperable with anything else. You want full validated support for C#, then you have to use IIS, buy a Windows Server license, etc. etc. But because open-source pieces of software have to be strong on their own to be popular, you will see all kinds of different crazy combinations that are possible. Also, what everyone else said about freedom in the GNU sense. That is very important for the future of humankind, since everything is being computerized these days.
125	I believe in a world where one shouldn't have to pay for necessary software.
126	The idea that information belongs to the world, and the free sharing of information.
127	Idealism. I'm an idealistic person, I'd like everybody to be equal. This might sound out of context, but Open Source software is the place to start: We are all equal and we contribute in anyway we can, without anybody being more important than anyone. Some people programm, some make art, some write. Me? I translate and spread the word. I always feel happy for contributing to the programs I use every-single-day in my computers.
128	It's fun and a lot of times helpful in my everyday job. It's also nice for establishing a good reputation.
129	it makes everything easier. Learning, working and having fun with computers it's easier with open source, you get to meet an awesome community and feel as a culture hero every time you share your creations.
130	I'll often open source m projects if I know it's not revolutionary or if I simply get bored with it. It's also nice to get advise from people more professional than me, as I'm currently just a student.
131	I like to improve the software I use, either professionally or personally. By pushing it upstream, it can

	ensure that the changes I made will be improved on in the future and bugs can be found.
132	Ownership of an idea is something that needs to be outgrown.
133	I like a deeper understanding of computers, and with that understanding, more control. I like the workflow I've built. The only thing I compiled myself and use on a regular basis is dwb, but I really love it. It's cheaper than proprietary. Like you guys (I expect) I'm a poor college student. Note that I am not yet good enough to make significant contributions back to the open source community.
134	Originally, I thought open source was a socialist plot to take over the world or something, and the "not reinventing the wheel" bit was a scam. Then I actually started programming, and I found that the ability to study preexisting programs to either deduplicate effort, or to better duplicate for your own ends, is such a aid, that had there not been the open source movement, I'd likely never had stayed into programming.
135	I am a Star Trek fan. I've been a Star Trek fan since my daughter came into my life. I had been raised in household of boys and I felt it was important to be able to bond with my daughter. So in that, I needed to have positive female role models for my daughter that had something to do with my geek life and Star Trek Voyager helped with that. Anyway, one day I was thinking about Star Trek and how they travel so far and how they use this operating system on their ship. And what ships in the future will use for an operating system. I wondered to myself, is some Bill Gates figure being paid in the future for the Enterprise's computer system? Ultimately I came to the conclusion that licensed software is not a sustainable business model for technology to evolve.Waiting for some business to release a new feature is not a way that we will get to the stars. We will never become something to Star Trek or Firefly or what have if we are stuck trying to convince an operating system business that we need access to this function to make this new technology. Ultimately, open source must win. Open Source must concur business for us to get to the where I want us to be. So, I support open source software. I use it as often as possible.
136	Learning more and since open source developers care much more about quality of the code rather than trying to hit deadlines the code is generally done more correct.
137	Because I create relatively small game mods that are rarely perfect. Open source allows other modders to see what I'm doing and improve/fix/collaborate to fix my mods or create larger mods that may build on the idea of my mods.
138	Most of my knowledge of various computer science concepts or applications has stemmed from open source in some way. Contributing to a project that I find interesting not only makes me want to expand that knowledge but also share what I know with anyone and everyone.
139	Using OSS: 1. Provided the output is acceptable, the choice between free and paid software is a no-brainer. 2. Knowing the source is open assures me that anyone can review it for correctness. At the very least, if I don't trust the distributor of a binary, I can compile it on my own. 3. Access to source means that I may easily make any changes I want. 4. Open source ensures longevity. Anyone is free to publish their own updates, especially if a software publisher discontinues support. Developing OSS: 1. Publishing source under an open license removes the legal stigma associated with copyright and proprietary licenses, making it more appealing to developers concerned about license terms.

	2. If the community at large is permitted to use and modify a program, then improvements are more likely to be found and may even be included in the main branch.
140	Unlike proprietary software, FOSS gives me a feeling of control over the application allowing me to hack the code for my own purposes. Proprietary software companies also make me feel that they not trust the users. They also control the users by adding malicious features to said application.
141	To become a better programmer. Because I love programming. To use it as a portfolio once I start seeking out interviews.
142	Because an open culture fosters a culture of innovation.
143	See how other people code Get used to read and understand code written by other people Build a better portfolio Learn about new technologies, languages, etc. Because coding is fun
144	it's my job. as in I get paid for it
145	I grew up using an open source browser and mailclient. I grew into the ecosystem and started programming on my own. I didn't release the source of early projects. Later I learned about git(hub). From this point on, nearly every personal project was open sourced. It was just clear for me, that I wanted everyone to benefit from my spare time programming. For one point paying on the internet sucks for some people and it was just imperfect spare time code. I slowly got dragged further in the ecosystem (participating in bigger projects) and now I can't imagine to write any proprietary code.
146	I want to give back. Most of what I do for work is based on open source work - least I can do is contribute.
147	Usually because I want a program or feature that doesn't exist yet. Sometimes because I've written something that I don't think I'll be able to earn money from, but I think other people might find it interesting or useful.
148	A give and take principle. Open-source things has helped me to learn almost half of what I know about programming. Now that I have mastered general programming and C++ I find it thrilling and thoroughly enjoyable to give back to the community. Also it is really cool to point at something you see someone else use and just go "I made that", or "I implemented the code that makes this program able to stream 1080p on a 486". One big motivation comes from GNU & Linux, where basically my whole OS is open-source, so that if I miss any function in the OS or any accessory/application, I can just implement it. Same thing goes for bugs.
149	Learning from others and helping cool new things come to be
150	Disclaimer - I don't support use of the term "open source," but I am a supporter of free software. That's free as in freedom, not price. I develop free software because it's important for anyone who uses a computer to be able to study and modify the programs that they use. When I was a teenager I became interested in software engineering. The proprietary software I was using made it very difficult or impossible for me to learn the things I wanted to learn. The authors of that proprietary software were intentionally keeping their source code secret from someone who wanted to study and improve it. I was forbidden from learning more about how my own computer was operating. This was very discouraging. However, eventually I found that I was able to download the GNU/Linux operating system, study and

	modify the code, and share my modifications with the rest of the community, because it was free software. The whole reason I was motivated to enter the software engineering field that I work in now was because of free software. I want everyone else to be able to follow this path, especially our children. I don't believe we have anything to gain from keeping secrets from a new generation of people who are willing to learn, especially as computers become more and more important in their lives.
151	Because I see it as a necessary part of the "third revolution". Because I believe in collaboration being a better social model than competition. Because it gives a purpose to my life.
152	I contribute to open source because I use it. If I find an issue which I am knowledgeable and capable enough to address for my self I contribute it back as I have no other use for it. I release code under open source license because programming is fun and again I have no other use for it. While I don't run a business I would expect if I tried I would have more success with open source than I would with closed. Focus would be on different forms to sustain the business than just selling software.
153	My first open source project started as a weekend project that I thought could be useful to others and in the linux world it was "the natural thing" to make it open source. In hindsight I can say that making my project open source made it a better project, because of the contributions of others. It was mostly motivation by the positive feedback and new ideas that I haven't thought of. It is also a lot easier (at least in the linux/unix ecosystem) to get exposure because it's more likely that your project will be packaged for distribution when it's open source. Then there's the point that it's a great way of learning new stuff. Be it by looking at someone else's code, or social skills by interacting with other poeple, or bug reports or patches that make you see your own mistakes and learn from them. And not to forget it's (most of the time) fun to interact with other people.
154	I like the philosophy of open source. Freedom to use the software on my computer the way I want is very important to me. I do not want a mega courp telling me how often I can install my own software and who I can share it with and if I can share it, or if I can change something. I've used various Linux distros for years and Firefox..I like these way better than what is provided by companies who have made absurd amounts of money price gouging people.
155	1. So that I know for certain that large companies aren't spying on me through the software I use. 2. Because I simply don't have the money to purchase the software large companies offer. 3. To learn something, whether that be a new programming language, a new piece of software or anything that interests me, without having to fork out large sums of money to do so. For example, I'm currently learning about DJing by playing around with the program Mixxx (among other things), which is open source. 4. So I can interact with like-minded people.
156	In no particular order: - Learning to code means coding a lot. An open source project is as good as any for getting experience. - As the GPL license text states, "This program is distributed in the hope that it will be useful [...]" and that's another motivation for FOSS involvements -- the hope that it will be useful to someone! - Directly or indirectly, it can enhance you chances of being hired. - And more!
157	Because it free, and if you know how to change the software, you can.
158	Learning experience, but also free high-quality software that I can be assured "Just Works" and is trustworthy. Also reliable and faster than Microsoft.

159	Selfishness: if there's a problem in an open-source project that affects me, I will do whatever I can to make it be fixed, depending on my aptitude with the project's programming language, from reporting it to fixing it myself.
160	I received help in critical situations from open source communities, for which I was very thankful, and by which I was impressed. I wanted to give back, and it was something I wanted to be part of.
161	I'm participate in open source because it's one of the things necessary for _free_ software. (guess I don't have to explain what free software is)
162	http://hexayurt.com is my project - a highly successful and widely used simple housing system. Simply, there's no other way to get an idea to scale in an industry which has extraordinary capital requirements, regulation and volatility of demand. Disaster relief shelter companies either manufacture tents in Pakistan with factories that are on constant standby (dropping other work to pick up an emergency call for tents) or are heavily funded for a few years by government orders which then dry up under the next administration. That's why the field has essentially been stagnant since the 1960s. Open source (well, Free Hardware) provides a way to break out of that cycle and deploy innovation. I hope that's helpful, please feel free to follow up if you have questions: [borttagen kontaktuppgift]
163	It is another level up the abstraction ladder from creating software through coding. I started out coding, then becoming a tech lead, the pushing for open source, then contributing to open source. The latest step for me has been to help create open source communities and to be an administrator in order to facilitate growth on a "higher" level than just coding for the project would be. I apologize for tip-toeing around the question, but I think that my answer needed some context. I contribute to open source because creating software is more fun when collaborating with others.

8.3 Bilaga 3 - Enkät 2

Page 1 of 1

Motivation in Open Source

We're doing a bachelor's thesis at Lund University on the Open Source culture and what motivates individuals to share their work and efforts with the community. Please take the time to answer our simple survey, thank you!

This is a follow up to our previous survey where we asked what motivates you to participate in open source. Based on the results of that survey we designed a follow up intended to rank the factors internally. We are very grateful to everyone who responded.

Definitions of the factors
Ideology: Altruism, political motivations, sharing is caring
Getting paid: Being employed to work on open source
Resume: Improving your resume for future employment
Improves skill: Learning how to work in a project, improving coding skills, learning new techniques
Scratching an itch: Improving software that you use, adding features, fixing bugs
Legacy: My code lives on, creating public artifacts
Reciprocity: Giving back to the community
Enjoyment: Having fun, feeling the flow, the joy of creating, artistic expression

Why do you participate in Open Source?
What factors motivate you to partícipate in open source? Please rate each on a scale from "Not important" to "Very important".

	Not important	Slightly important	Important	Very important	Most important
Ideology	○	○	○	○	○
Getting paid	○	○	○	○	○
Resume	○	○	○	○	○
Improves skill	○	○	○	○	○
Scratching an itch	○	○	○	○	○
Legacy	○	○	○	○	○
Reciprocity	○	○	○	○	○
Enjoyment	○	○	○	○	○

What motivates you in the workplace?
Which of the above mentioned factors motivate you to perform in a work place? Please rate each on a scale from "Not important" to "Most important".

	Not important	Slightly important	Important	Very important	Most important
Ideology	○	○	○	○	○
Getting paid	○	○	○	○	○
Resume	○	○	○	○	○
Improves skill	○	○	○	○	○
Scratching an itch	○	○	○	○	○
Legacy	○	○	○	○	○
Reciprocity	○	○	○	○	○
Enjoyment	○	○	○	○	○

Litteraturförteckning

Barney, J. B. (1986). Organizational Culture: Can It Be a Source of Sustained Competitive Advantage? *The Academy of Management Review, 11*(3), 656-665.

Baytiyeh, H., & Pfaffman, J. (2010). Open source software: A community of altruists. *Computers in Human Behavior, 26*, 1345-1354.

Benbya, H., & Belbaly, N. (2010). Understanding Developers' Motives in Open Source Projects: A Multi-Theoretical Framework. *Communications of the Association for Information Systems, 27*, 589-610.

Black Duck Software & North Bridge Venture Partners. (April 2013). *Seventh Annual Future of Open Source Survey Results Show Culture, Quality and Growth Driving an Open Revolution*. Hämtat från Black Duck: http://www.blackducksoftware.com/news/releases/seventh-annual-future-open-source-survey-results-show-culture-quality-and-growth den 28 April 2013

Bonaccorsi, A., & Rossi, C. (2003). Why Open Source software can succeed. *Research Policy, 32*, 1243-1258.

Corbet , J., Kroah-Hartman, G., & McPherson, A. (den March 2012). *Linux Kernel Development. How Fast it is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It* . Hämtat från The Linux Foundation: http://go.linuxfoundation.org/who-writes-linux-2012 den 17 April 2013

Coverity. (2011). *Coverity*. Hämtat från Coverity Scan 2011 Open Source Integrity Report: http://www.coverity.com/library/pdf/coverity-scan-2011-open-source-integrity-report.pdf den 25 April 2013

Coverity. (den February 2012). *Coverity Scan: 2011 Open Source Integrity Report.* Hämtat från Coverity: http://www.coverity.com/library/pdf/coverity-scan-2011-open-source-integrity-report.pdf den 20 April 2013

Coverity. (u.d.). *Coverity*. Hämtat från Coverity Scan: 2011.

Cutler, K. (den 24 Oktober 2009). *Mark Zuckerberg on how to build hacker culture inside a company Read more at http://venturebeat.com/2009/10/24/live-blogging-mark-zuckerbergs-talk-at-startup-school/#08evYu7LGH4KYbAA.99* . Hämtat från Venture Beat Social: http://venturebeat.com/2009/10/24/live-blogging-mark-zuckerbergs-talk-at-startup-school/ den 20 April 2013

GitHub. (den 10 April 2013). *Five years*. Hämtat från GitHub: https://github.com/blog/1470-five-years den 30 April 2013

Goldstein, H. (September 2005). *Who Killed the Virtual Case File?* Hämtat från ieee Spectrum: http://spectrum.ieee.org/computing/software/who-killed-the-virtual-case-file den 30 April 2013

Hars, A., & Ou, S. (2002). Working for Free? Motivations for Participating in Open-Source Projects. *International Journal of Electronic Commerce, 6*, 25-39.

Hertel, G., Nieder, S., & Herrman, S. (2003). Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. *Research Policy, 32*, 1159-1177.

Jacobsen, D. (2002). *Vad, hur och varför? Om metodval i företagsekonomi och andra samhällsvetenskapliga ämnen.* Lund: Studentlitteratur.

Jordan, B. (1989). Cosmopolitical obstetrics: Some insights from the training of traditional midwives. *Social Science and Medicine, 28*, 925-944.

Jordan, B. (1989). Cosmopolitical obstetries; Some insights from the training of traditional midwives. *Social Sience and Medicine, 28*(9), 925-944.

Kerr, D. (den 31 December 2012). *Reddit's visitors skyrocket in 2012 with 37 billion page views*. Hämtat från CNET: http://news.cnet.com/8301-1023_3-57561466-93/reddits-visitors-skyrocket-in-2012-with-37-billion-page-views/ den 28 April 2013

King, R. (den 6 September 2012). *Big Companies Open Up to Open-Source Software*. Hämtat från The Wall Street Journal: http://online.wsj.com/article/SB10000872396390443589304577633733725243996.html den 28 April 2013

Kishida, K., & Ye, Y. (2003). Toward an Understanding of the Motivation of Open Source Software Developers. *Software Engineering*, 419-429.

Lakhani, K., & von Hippel, E. (2003). How open source software works: "free" user-to-user assistance. *Research Policy, 32*, 923-943.

Lave, J., & Wenger, E. (1991). *Situated Learning - Legitimate Peripheral Participation.* Cambridge: Cambridge University Press.

Lerner, J., & Tirole, J. (2002). Some simple economics of Open Source. *The Journal of Industrial Economics, Volume L*, 197-234.

Linux Foundation. (2012). *Using Git*. Hämtat från Linux Foundation: http://ltsi.linuxfoundation.org/developers/using-git den 25 April 2013

Macale, S. (den 09 November 2011). *Sheryl Sandberg on how Facebook's culture differs from Google's. She's worked at both.* Hämtat från The Next Web: http://thenextweb.com/facebook/2011/11/09/sheryl-sandberg-on-how-facebooks-culture-differs-from-googles-shes-worked-at-both/ den 20 April 2013

McPherson, A. P.-E. (2008). *The Linux Foundation*. Hämtat från Estimating the Total Development Cost of a Linux Distribution: http://www.linuxfoundation.org/sites/main/files/publications/estimatinglinux.html den 16 April 2013

Mockus, A., Fielding, R., & Herbsleb, J. (2002). Two case studies of open source software development: Apache and mozilla. *ACM Transactions on Software Engineering and Methodology, 11*(3), 309–346.

Mozilla Developer Network. (den 3 April 2013). *Coding Style*. Hämtat från Mozilla Developer Network: https://developer.mozilla.org/en-US/docs/Developer_Guide/Coding_Style#General_C.2FC.2B.2B_Practices den 25 April 2013

Netcraft. (2012). *September 2012 Web Server Survey*. Hämtat från Netcraft: http://news.netcraft.com/archives/2012/09/10/september-2012-web-server-survey.html den 16 April 2013

Open Source Initiative. (u.d.). *The Open Source Definition*. Hämtat från Open Source Initiative: http://opensource.org/osd den 10 Maj 2013

PLC Magazine. (den 1 April 1993). *Taurus: Learning lessons from failure*. Hämtat från PLC Magazine: http://plc.practicallaw.com/1-100-3790 den 30 April 2013

Raymond, E. (1999). The Cathedral and the Bazaar. *Knowledge, Technology, & Policy*, 23-49.

Reddit. (2013). *Reddit Linux*. Hämtat från Reddit Linux: http://www.reddit.com/r/linux den 25 April 2013

Smith, K. (den 18 April 2013). *Here's what it's like to work at Facebook right now*. Hämtat från Business Insider: http://business.financialpost.com/2013/04/20/here-what-its-like-to-work-at-facebook-right-now/ den 20 April 2013

Spence, M. (1973). Job Market Signaling. *Quarterly Journal of Economics, 87*, 355-374.

W3Counter. (2013). *Web Browser Market Share Trends*. Hämtat från W3Counter: http://www.w3counter.com/trends den 16 April 2013

Waterman, R. H., & Peters, T. J. (1982). *In search of excel- lence.* New York: Harper and Row.

Wenger, E. (1998). *Communities of Practice. Learning as a social system.* Hämtat från Community Intelligence Labs: http://www.co-i-l.com/coil/knowledge-garden/cop/lss.shtml den 25 April 2013

Wenger, E. (2006). *Communities of Practice - a brief introduction.* Hämtat från Wenger-Trayner: http://wenger-trayner.com/wp-content/uploads/2012/01/06-Brief-introduction-to-communities-of-practice.pdf den 25 April 2013

Woodall, B., & Seetharaman, D. (den 5 August 2011). *Honda recalls 2.5 million vehicles on software issue*. Hämtat från Reuters: http://www.reuters.com/article/2011/08/05/us-honda-recall-idUSTRE77432120110805 den 30 April 2013