Kyrre Myrbostad

Å spille med åpne kort

En studie i kunnskapsflyt blant "Open Source"-utviklere

Trondheim, April 2006

Norges teknisk-naturvitenskapelige universitet
Fakultet for Informasjonsteknologi, matematikk og elektroteknikk
Institutt for datateknikk og informasjonsvitenskap

Masteroppgave
Studieprogram: Master i informatikk

Hovedveileder: Eric Monteiro, IDI

NTNU
Det skapende universitet

«There is no practical obstacle whatsoever now to the creation of an efficient index to *all* human knowledge, ideas and achievements, to the creation, that is, of a complete planetary memory for all mankind.»
H G Wells, 1937

«The new electronic interdependence recreates the world in the image of a global village.»
Marshall McLuhan, 1962

«Hello everybody out there using minix - I'm doing a (free) operating system
(just a hobby, won't be big and professional...»
Linus Torvalds - grunnlegger av Linux, 1991

1 Innledning

Min første nærkontakt med fenomenet Open Source fikk jeg for ca 6 år siden. Jeg hadde hørt rykter om et operativsystem, Linux, som visstnok var laget av en mengde frivillige dataentusiaster over hele verden. Ikke bare hadde disse menneskene laget systemet på sin egen fritid, de ga det faktisk bort gratis i tillegg, dog med noe som lignet et politisk budskap om åpenhet på kjøpet. Jeg lot meg umiddelbart fascinere av denne kuriøse blandingen av teknologi, kultur, etikk og politikk, og begynte å lete etter mer informasjon om Linux og Open Source.

Det første som slo meg med Open Source var omfanget og volumet av det hele. Det jeg trodde var en liten gjeng med dataentusiaster, viste seg å være en omfattende og mangesidig bevegelse, med røtter helt tilbake til den tiden en enkelt datamaskin gjerne fylte en etasje alene. Siden den gang har ulike grener av dette miljøet produsert en imponerende mengde åpen programvare, innenfor de fleste former for databehandling.

Jeg oppdaget at å installere Linux var noe helt annet enn å installere et operativsystem fra en av de store kommersielle tilbyderne, som Microsoft eller Apple. Det viste seg at Linux var bare en del av et enormt puslespill av programmer, som man selv kunne putte sammen til akkurat det operativsystemet man selv ønsket. En overveldende valgfrihet, og en voldsom oppgave i seg selv, derfor var jeg glad for å se at det fantes flere distributører av åpen programvare som kunne gjøre mange av disse valgene for meg. Jeg endte opp med å prøve en rekke ulike distribusjoner, fra giganten Debian GNU/Linux, med over 8000 tilpassede og testede programmer, til Damn Small Linux, som kan kjøres direkte fra en mini-CD på størrelse med et visittkort. Jeg lekte faktisk også med tanken på å sette sammen mitt eget unike system selv, om ikke annet bare fordi det var mulig.

Mine eskapader i Linux-verden var ikke uten frustrasjoner og problemer. I de aller fleste tilfellene fantes det imidlertid mange andre der ute som hadde slitt med lignende problemer, og som hadde klagd sin nød om dette i en av de utallige e-postlistene, diskusjonsforumene eller nyhetsgruppene på internett. Etter hvert som min kunnskap om Linux økte, og jeg ble mer familiær med de ord og uttrykk som var vanlig å bruke, ble jeg flinkere til å finne den informasjonen jeg trengte på internett, samtidig som min nysgjerrighet overfor miljøet som sådan økte. Følelsen av mestring, læring, og av å ha kontroll over min egen datamaskin, bidro også til å holde på min interesse.

Mitt møte med Open Source gjorde inntrykk på flere plan. Mengden av informasjon som fantes i

krinkler og kroker på internett, gitt frivillig og uten vederlag, var imponerende. Bredden av programvare, og hvordan disse fungerte sammen til utførelse av en rekke ulike formål, var også en øyeåpner for meg. Hvordan kunne dette miljøet ha vokst seg så stort og klart å produsere så mange komplekse informasjonsystemer?

Den nysgjerrigheten og undringen jeg fikk den gangen, ligger til grunn for min avhandling i dag. Jeg ønsker å forstå hvordan «Open Source»-utviklere faktisk arbeider; hvordan de kommuniserer og utveksler kunnskap seg i mellom. Kort sagt: Hvordan er det å spille med åpne kort?

1.1 Ordforklaring

Filformat: Når lyd og bilde blir lagret digitalt sitter man igjen med en mengde rådata. Disse rådataene blir knyttet til en fil som i tillegg til data også inneholder informasjon om dataene, for eksempel lengde og billedoppløsning. Det finnes mange konkurrerende filformater for lagring av multimediadata, for eksempel AVI fra Microsoft, OGG fra Xiph.Org og MOV fra Apple Computers.

Kodek: I mange tilfeller er det ikke ønskelig å operere med rå data, enten fordi det tar for stor plass å lagre, krever for mye datakraft å arbeide med, eller fordi det krever for mye båndbredde å distribuere. Dermed koder man dataene etter en bestemt metode slik at dataene blir lettere å behandle og distribuere. De kodede dataene kan så dekodes ved avspilling så lenge metoden er kjent og støttes av den aktuelle avspilleren. Programvaren som gjør dette mulig kalles en kodek. Det finnes to hovedtyper kodeker, de som mister informasjon i prosessen og de som beholder all opprinnelig informasjon om rådata. Den første varianten blir som regel foretrukket fordi den er mest effektiv og fordi mye av informasjonen som går tapt ikke er essensiell for opplevelsen av kvalitet. MP3-formatet benytter en lydkodek med tap, men siden mye av informasjonen som fjernes i liten grad er relevant for lydopplevelsen, brukes ofte kodeken til lagring og avspilling av lyd. Quicktime fra Apple Computers er et eksempel på en videokodek med tap som bl.a. blir mye brukt til videodistribusjon over nett. I tilfeller der multimediadata skal viderebehandles senere vil imidlertid de fleste foretrekke enten ubehandlede rådata eller en tapsfri kodek. Hvilken kodek som blir benyttet vil være angitt i filformatets metainformasjon.

1.1.1 Forkortelser

DVD – Digital Versatile Disc. Et lagringsmedium som ofte brukes til videodistribusjon

MP3 – En kodek basert på MPEG1-standarden. Brukes bl.a. i mange bærbare musikkspillere.

TCP/IP – Transport Control Protocol / Internet Protocol. To standarder som sammen danner grunnlaget for mesteparten av datatransport over internett.

URL – Uniform Resource Locator. En standardisert metode å adressere til en ressurs på internett.

HTML – Hypertekst Markup Language. En standard utviklet til hjelp for å lage internettsider med hypertekst, eller lenker om man vil, som kan vises i en nettleser.

HTTP – Hypertekst Transfer Protocol. En standard for å overføring av informasjon over internett, ofte i form av HTML-sider.

FTP – File Transfer Protocol. En standard for filoverføring på internett.

SMTP – Simple Mail Transfer Protocol. En standard som ligger til grunn for transport av e-postmeldinger fra tjener til klient.

W3C – the World Wide Web Consortium. En standardorganisasjon som arbeider med flere av de grunnleggende standardene på nett, inkludert URL, HTTP og HTML. (http://www.w3c.org)

IETF – Internet Engineering Task Force. En av de mest toneangivende standardorganisasjonene med internett som arbeidsfelt. Arbeider med blant annet TCP/IP og SMTP. (http://www.ietf.org)

ISO – International Standards Organisation. En organisasjon for utvikling og koordinering av en rekke standarder innenfor alle slags arbeidsfelt. Standarder her utvikles i samarbeid med store aktører i markedet og ledende akademiske institusjoner.

ITIL - IT Infrastructure Library. En samling av generelle arbeidsprosedyrer for IT- og telesektoren, utviklet av britiske myndigheter i samarbeid med en rekke store IT- og telefirma.

POSIX – Standard for konstruksjon av operativsystemer. (http://www.knosof.co.uk/posix.html)

UNIX – En betegnelse på en klasse av operativsystemer som er konform med POSIX-standarden.

ARPANET – Advanced Research Project Agency. Militært nettverk, forløper til dagens internett.

WIKI – Betegnelse på en nettside som kan enkelt redigeres og som inneholder historie om endringer. Brukes ofte for å samle generell informasjon rundt et emne, der alle kan bidra.

WIKIPEDIA – Verdens største nettbaserte leksikon, der alle kan endre og legge til artikler.

IRC – Internet Relay Chat. En tekstbasert måte å kommunisere i sanntid på, oppdelt i grupper.

MPEG – Motion Pictures Expert Group. En organisasjon for utvikling av multimediastandarder.

2 Litteratur

Første del av avhandlingen er viet litteratur om kunnskap og kommunikasjon generelt, og om Open Source spesielt, for å etablere en historisk og teoretisk kontekst rundt fenomenet.

2.1 «Open Source»-bevegelsen

«Open Source» (OS), eller åpen kildekode som det heter på norsk, har i løpet av noen få tiår gått fra å være et marginalt fenomen blant enkelte akademikere og spesielt interesserte, til å bli en teknologisk tungvekter, som har vært sentral i å forme hvordan vi i dag kommuniserer og samhandler. Jeg vil i denne delen gå nærmere inn på fenomenet OS; inkludert dets historie, arbeidsmetoder og verktøy.

2.1.1 Bakgrunn og historisk utvikling

For å finne røttene til den kulturen og praksisen som forbindes med bruken av åpen kildekode, må vi tilbake til programmeringens barndom. Etter andre verdenskrig, i kjølvannet av suksessen til de første store kodeknekkermaskinene[1], vokste det fram et fragmentert forskermiljø av ingeniører – med en spesiell interesse for datamaskiner og programmering. Inspirert av disse datapionerene ble det etter hvert dannet flere innovative datateknologi-miljøer rundt enkelte amerikanske universitet

[1]For eksempel ENIGMA

og forskningssentra, inkludert MIT og Xerox Parc (Raymond 1997). Forskerne som arbeidet her likte å kalle seg selv «hackere», som i positiv forstand viste til deres tekniske ferdigheter og evner til elegant problemløsning, samt forkjærlighet for programmering som aktivitet i seg selv. Den negative versjonen av en «hacker», en ulovlig inntrenger eller destruktiv programmerer, ble skapt av mediene for å beskrive datakriminalitet på 1980-tallet. Denne betydningen har festet seg i de flestes bevissthet, på tross av iherdige forsøk fra de opprinnelige hackerne på å etablere begrepet «cracker» som et alternativ (Hannemyr 1997).

Mot slutten av 1960-tallet og utover 1970-tallet var det to parallelle hendelser som sammen ble en sterk drivkraft i oppblomstringen av en egen kultur og et eget fellesskap blant enkelte programmerere. Disse to hendelsene var (1) utviklingen av ARPANET, forløperen til dagens internett, og (2) utviklingen av operativsystemet UNIX sammen med programmeringsspråket C. ARPANET var et prosjekt støttet av det amerikanske militæret, med det formål å konstruere et nettverk som ikke var avhengig av en sentral tjener, slik at det kunne overleve alvorlige skader i krigstid uten å bryte sammen. Løsningen ble å utvikle en svært åpen og fleksibel standard for nettverkstrafikk, TCP/IP. Ved hjelp av TCP/IPs udiskriminerende og åpne arkitektur, ble det etter hvert enkelt for universiteter og forskningssentra og tilknytte seg ARPANET. Tempoet og mengden informasjon som ble utvekslet over dette nettet slo i løpet av kort tid alle tidligere rekorder, og det amerikanske militæret trakk seg etter hvert ut av ARPANET, til fordel for en mer lukket og kontrollerbar nettverksløsning. Utviklingen av nettet fortsatte imidlertid ufortrødent, mye på grunn av UNIX, som var forholdsvis enkelt å tilpasse til nye maskinvarearkitekturer, og som hadde nettverkstøtte innebygd. UNIX-verktøy og applikasjoner ble dermed raskt spredd rundt i nettet, og etablerte seg som en de-facto standard for «hackere» over hele verden (Raymond 1999). An annen viktig årsak til den raske spredningen av teknologien, lå i den politiske stemningen som rådet, med 68-opprøret som en sterk katalysator. Mange så på det som en politisk og ideologisk oppgave å bringe nettverksteknologien ut til folket; et demokratisk prosjekt (Hannemyr 1997).

Ved MITs senter for kunstig intelligens var det et svært aktivt «hacker»-miljø på 60- og 70-tallet, som hadde en sentral rolle i spredningen og utviklingen av mange UNIX-verktøy. Det var utstrakt praksis å dele på informasjon og kildekode, som ble oppfattet som et felles gode på ligne med grunnforskning. Utover 80-tallet begynte dette miljøet gradvis å smuldre opp, mange forsvant til bedre betalende kommersielle firma eller ble mer restriktive med å dele sine oppdagelser. Richard Stallmann arbeidet ved MIT på denne tiden, og følte frustrasjon over den økende mangelen på åpenhet, som i hans øyne var et alvorlig hinder for miljøets vitenskapelige utvikling (Stallmann

1999). Dette skjedde også samtidig med en rennesanse for Taylors «scientific management» (Taylor 1911) innenfor systemutvikling og mange andre samfunnsområder. Med taylorisme fulgte et fokus på strenge hierarkiske beslutningsstrukturer, klart definerte arbeidsoppgaver og kontroll over informasjonsflyten i en organisasjon. Stallmann opplevde dette som om man at alle var opptatt med å bygge vegger mellom hverandre, i stedet for å samarbeide. Det hele toppet seg i det Stallmann ikke lenger kunne bruke sin egen skriver, fordi den hadde en feil i programvaren som styrte den, og leverandøren nektet å gi ham til gang til kildekoden – slik at han kunne fikse problemet. Stallmann sier han følte et moralsk ansvar for å utarbeide et alternativ til disse mer og mer lukkede systemene han måtte arbeide med. I 1984 dannet han «The Free Software Foundation» (FSF), som hadde til hensikt å støtte opp om utviklingen av åpen og allment tilgjengelig programvare. «Free» er ikke her ment i betydningen gratis, men i betydningen ikke lukket eller begrenset, eksemplifisert ved uttrykket «Free as in free speach, not as in free beer». Hovedprosjektet til FSF var å stable sammen et fullverdig operativsystem, der all koden var åpen for andre til å se på, endre og distribuere. Dette prosjektet ble kalt GNU, som sto for «GNU is Not Unix». Dette var en spøkefull henvisning til UNIX, som på mange måter var modellen og utgangspunktet for GNU-prosjektet. Forkortelsen GNU viser tilbake på seg selv, noe som skaper en uendelig løkke innover; en språklig underfundighet for de innvidde, det vil si programmerere.

For å sikre at koden som ble produsert av GNU-prosjektet skulle beholde sin status som åpen og tilgjengelig, utviklet Stallmann og hans likesinnede en egen lisensavtale kalt GPL-lisensen (GNU Public Lisence). Denne avtalen var mer opptatt av brukerens rettigheter enn begrensninger, og sikret at man kunne lese, modifisere og publisere hele eller deler av koden etter eget forgodtbefinnende, så lenge man publiserte resultatet under samme lisens. Med andre er lisensen som et intellektuelt og ideologisk virus, og hindrer at kode som en gang har vært fri, kan bli lukket og gjøres utilgjengelig (Stallmann 1999). Siden denne lisensen fokuserer på brukerens rettigheter i stedet for begrensninger, i motsetning til mer vanlige Copyright-lisenser, kalles den i dag gjerne Copyleft på folkemunne. Under GPL-lisensen ble det utviklet en lang rekke verktøy, som alle var svært modulære og kunne brukes sammen for å oppnå mange slags ulik funksjonalitet. Det var en rådende ideologi at en applikasjon ikke skulle prøve å gjøre mer en en ting av gangen, men at den skulle gjøre denne ene tingen svært bra. Modularitet var et viktig aspekt ved GNU-prosjektet.

Etter hvert fikk Stallmann knyttet til seg flere og flere utviklere under FSF-paraplyen, og mange GNU-verktøy utviklet seg til å bli en essensiell del av en hver UNIX-installasjon. Dessverre for Stallmann møtte FSF på et alvorlig hinder. GNU-prosjektet lyktes med å etablere en imponerende

mengde programvare for sitt operativsystem, men manglet kjernen, den innerste biten i operativsystemet som skal håndtere direkte kontakt med maskinvaren. GNU hadde en kjerne under utvikling, HURD, som skulle gjøre GNU fullstendig uavhengig av kommersielle UNIX-varianter. Utviklingen på HURD gikk imidlertid sent, og FSF gikk fra en flygende start, inn i en periode med dalende forventninger og interesse blant sine brukere. Kommersielle UNIX-varianter, som for eksempel BSD, fikk økt vind i seilene på bekostning av FSF sine alternativer. Det skulle likevel snart komme en hjelpende hånd til GNU-prosjektet, om enn fra svært uventet hold.

2.1.2 Linux og spredningen av internett

Historien om hvordan Linux kom til verden, er ofte brukt som selve kroneksemplet på hvordan OS kan fungere som konsept. Operativsystemet Linux er i dag også et av de aller største flaggskipene i OS-verden, og har hatt en viktig rolle i internettets historie. Slik begynte det:

En ung student ved universitetet i Helsinki, Linus Torvalds, hadde et kinkig problem. Han ønsket å benytte all den kraftfulle programvaren som fantes til UNIX-plattformen, inkludert GNU-verktøyene, men hadde ikke midler til å anskaffe den forholdsvis dyre maskinvaren som var påkrevd. Han bestemte seg derfor i 1991 for å starte arbeidet med å utvikle sin egen UNIX-variant, som skulle fungere på billige personlige datamaskiner av typen x86. Torvalds startet med å undersøke Minix, som var utviklet av professor Andrew S. Tanenbaum til undervisningsformål. Han fikk raskt tilgang til kildekoden til dette systemet og brukte det han lærte derfra til å starte arbeidet med å lage en et helt nytt operativsystem fra grunnen av. Som utgangspunkt benyttet Torvalds den samme standarden som Minix og mange andre UNIX-varianter var basert på, nemlig POSIX.

Etter å ha kommet i gang med å etablere den grunnleggende strukturen i sitt nye operativsystem, sendte Torvalds ut en mail til en nyhetsgruppe for minix-utvikling, der han i all ydmykhet beskrev hva han holdt på med, og spurte om noen hadde lyst til å være med og hjelpe til. Responsen viste seg å være overveldende, ettersom svært mange andre delte Torvalds ønske om å få tilgang til den kraftfulle UNIX-verdenen via sine forholdsvis billige personlige datamaskiner. I løpet av kort tid hadde prosjektet en mengde utviklere, som arbeidet med hver sin del av systemet parallelt, og som utviklet seg til et verdensomspennende fellesskap av programmerere, med Torvalds som koordinator. Ettersom FSF hadde gjort omfattende arbeid på mange områder unntatt selve kjernen i operativsystemet, kunne Torvalds, sammen med flere hundre programmere fra hele verden, konsentrere seg om kun denne delen og dermed få opp et fungerende system med et imponerende

programvareutvalg på rekordtid.

Allerede i 1993, to år etter Linus Torvalds annonserte sitt ”hobbyprosjekt”, var første versjon av operativsystemet GNU / Linux et faktum, der navnet Linux sto for den nye kjernen som Torvalds og hans hærskare av programmerere hadde utviklet (Raymond 1999). Etter hvert har navnet Linux på folkemunne blitt ensbetydende med hele operativsystemet, og ikke bare kjernen som styrer maskinvaren. Dette har vært, og er, en kilde til frustrasjon, både for GNU og for andre kompilasjoner som benytter Linux som kjerne. Jeg nevner dette skisma først og fremst for å understreke betydningen av *bredden* i hackermiljøet, for realiseringen av et åpent, ikke-kommersielt operativsystem. Linux var en brikke i et puslespill som mange hackere hadde jobbet med lenge, og selv om det var en svært viktig brikke, er det viktig å ikke underkjenne det arbeidet som lå i bunnen. Det som gjør Linux ekstra spennende, er måten det ble realisert på, med en åpen og nettbasert prosjektform. Samarbeid over nett var ikke et nytt fenomen, men det hadde ikke vært prøvd ut i så stor skala eller på et så omfattende prosjekt som det Torvalds ga seg ut på. Det ruvende problemet GNUs erfarne programmerere hadde slitt med i årevis, ble plutselig løst av en ung student, med en armada av tilfeldige programmerere fra alle verdenshjørner. En oppsiktsvekkende hendelse, som satte et toneangivende eksempel for senere utviklingsprosjekter innenfor hackermiljøet.

Etableringen av GNU / Linux åpnet for at mange flere hadde råd til å delta i det nettverket som hadde bygget seg opp rundt flere akademiske sentra rundt om i verden. Utover 1990-tallet ble maskinvaren for x86-plattformen stadig billigere, samtidig som ny nettverksteknologi og infrastrukturen la veien åpen for å spre nettverket ut til vanlige forbrukere. Flere og flere fikk tilgang til nye kommunikasjonskanaler, først og fremst i form av e-post og nyhetsgrupper. I 1993 kom Mosaic, en webleser i moderne forstand, som utnyttet hypertekst som en ny og banebrytende måte å navigere i informasjon. Med fremveksten av World Wide Web, fikk også kommersielle aktører øynene opp for nettets potensiale, selv om resultatene skulle komme tregt – og ikke uten alvorlige tilbakeslag[2].

Etter hvert som antallet brukere på nettet økte, oppsto et økende behov for å koordinere og enes om hvilke standarder man skulle benytte for ulike typer nettbasert kommunikasjon. Flere uavhengige frivillige organisasjoner ble dannet for å løse disse utfordringene. Organisasjonene fikk viktig drahjelp av et uformelt mandat fra «hackere» over hele verden. En av de mest innflytelsesrike av slike organisasjoner er IETF (Internet Engineering Task Force), som har ansvar blant annet for IP-

[2]Som ved «.com-bølgen» på slutten av nittitallet, der overdreven optimisme førte til mange konkurser .

standarden; internettets adresseringssystem. Her er deltakelse i organisasjonen frivillig og stemmerett basert på innsats. IETF lager forslag til nye standarder i form at utkast til RFC-dokumenter (Request for Change). Disse utkastene blir gjort offentlige på nett, slik at flest mulig kan delta i en debatt rundt utforming av dem. Deretter stemmes det over utkastene, og et ferdig RFC-dokument blir publisert. Uten disse organisasjonenes arbeid ville neppe internett ha vokst så raskt, eller kunnet tilby så mange tjenester som det gjør i dag. En av de store utfordringene for IETF i dag, er overgangen til en ny og utvidet versjon av IP-standarden (fra Ipv4 til Ipv6).

2.1.3 The Open Source Initiative

GNU / Linux ble stadig mer populært blant programmere og systemadministratorer, og ble oftere og oftere foretrukket som web- eller filtjenere foran mer kostbare proprietære systemer. FSF vokste seg større, og Richard Stallmann med flere oppnådde en opphøyet og glamorøs status i hackermiljøene. Alle var imidlertid ikke overbeviste. Den kommersielle programvareindustrien var jevnt over negativ til hele konseptet bak FSF og GPL-lisensen. De moralske betingelsene som FSF hadde brakt inn i systemutvikling ble oppfattet som blåøyd og av enkelte til og med som ren kommunisme. FSF sin kompromissløse holdning i forhold til GPL-lisensen og bruken av kode, ble oppfattet som unødvendig steil, også av enkelte innenfor hackermiljøene. Selve begrepet «Free Software» ble også oppfattet som uheldig av mange. Enten ble det oppfattet som gratis, noe som øyeblikkelig skremte bort eventuelle kommersielle interesser, eller så ble det riktig oppfattet som et politisk ladet begrep, noe som om mulig var enda mer skremmende for en eventuell kommersiell investor.

Midt oppe i denne begynnende interne striden i hackermiljøet, oppsto plutselig en svært uventet situasjon. Flere firma som utviklet nettlesere kjempet for tiden om kundenes gunst i et voksende marked. De to tyngste aktørene var uten tvil Micosoft Internet Explorer og Netscape Communicator. Det så lenge ut til at Netscape skulle vinne denne striden, men så valgte Microsoft å bygge sin nettleser som en integrert del av operativsystemet MS Windows, uten ekstra kostnad for installasjon. Microsoft hadde allerede rukket å bli nesten enerådende på hjemmemarkedet, og Netscape fikk dermed store problemer med å forsvare sin posisjon i markedet for nettlesere. Hvorfor skulle noen bry seg med å betale for en tjeneste de uansett får levert gratis med sitt operativsystem? For å kunne konkurrere med Microsoft måtte altså Netscape tilby et teknologisk klart overlegent produkt, for å forsvare den ekstra kostnaden for kunden. Samtidig tapte selskapet kontinuerlig penger, mens Microsoft sin pengebinge bare så ut til å vokse seg større og større.

Det neste skrittet Netscape foretok seg, kom uforberedt på hele programvareindustrien. Inspirert av den raske suksessen til GNU / Linux, webserveren Apache og flere andre prosjekter som opererte med åpen kildekode, annonserte Netscape den 22. januar 1998 at de ville frigi koden til sin nåværende versjon av Netscape[3]. På grunnlag av den frigitte koden ville de starte opp et eget parallelt prosjekt til Netscape, som skulle hete Mozilla. Dette prosjektet skulle operere med åpen kildekode, og i svært liten grad bli kontrollert eller styrt av ledelsen i Netscape (Raymond 1999).

Nyheten om Netscapes beslutning utløste en nærmest euforisk stemning blant hackere over hele verden. Et stort internasjonalt firma hadde tatt deres arbeidsform til seg, noe som hadde en sterk symbolverdi og ga et kraftig løft for hele miljøet. Nå var det ikke lenger like enkelt å avskrive hackerne som et perifert og sært fenomen, og mange øyne i programvareindustrien ble sperret opp av overraskelse eller til og med vantro. Nå skulle det vise seg at overgangen til åpen kildekode ikke var smertefri for Netscape. GPL-lisensen ble oppfattet som altfor streng, spesielt i forhold til måten den spredte seg til alt man brukte den til. I tillegg var ikke Netscape komfortabel med å bli assosiert med FSF, mye på grunn av sistnevntes politiske brodd og anti-kommersielle holdninger. Det var imidlertid krefter innenfor hackermiljøet som lenge hadde jobbet med å etablere et alternativ til FSF, som skulle være mer nøytralt politisk og mindre steil mot kommersialisering. Disse fryktet at både rekrutteringen og kontakten med markedet ville lide hvis man ikke etablerte et mer nøytralt og «lettfordøyelig» alternativ til FSF, men visste at de måtte trå varsomt for ikke å skape for mye splittelse i miljøet.

En av de mest anerkjente distributørene av GNU / Linux, Debian Software, hadde på egen hånd utarbeidet et sett med retningslinjer for utformingen av lisenser som de kunne godta i sin distribusjon. I tillegg hadde de sin egen definisjon av «Free Software», som var mer nøytralt og praktisk formulert enn det FSF opererte med. Disse dokumentene ble nå relansert under navnet «The Open Source Definition», og det ble gjort viden kjent at hvis man oppfylte disse kriteriene, kunne man smykke seg med tittelen «Open Source». Stallmann og hans FSF protesterte øyeblikkelig, og følte at hele hensikten med hacker-bevegelsen falt sammen, i det man fjernet den moralske og politiske dimensjonen fra systemutviklingen. For mange andre hackere, inkludert Linus Torvalds, var dette nøyaktig det man hadde ventet på. Endelig kunne man begynne å bygge bro mot den kommersielle programvareindustrien, uten å risikere å havne i uønskede diskusjoner om ideologi eller politikk. Bruce Perens og Eric Raymond startet etter hvert en egen organisasjon kalt «The Open Source Initiative» (OSI), som skulle forvalte retningslinjer og definisjoner, samt

[3] http://wp.netscape.com/newsref/pr/newsrelease558.html

fungere som talerør for Open Source-bevegelsen (Perens 1999).

Kort tid etter lanseringen av begrepet Open Source, kunne man observere en veritabel snuoperasjon blant mange kommersielle aktører i markedet. Det som før hadde vært avfeid som kommunisme, ble nå løftet opp som et nytt og friskt pust i bransjen, i mange tilfeller av de samme personene som hadde dømt det nord og ned (Raymond 1999). Netscape var intet unntak i så måte, og fikk, i samarbeid med OSI, utarbeidet en ny lisens for Mozilla. Denne lisensen, Mozilla Public Lisence, var skrevet i henhold til «the Open Source Definition», slik at Mozilla nå kunne kalle seg for et ekte «Open Source»-prosjekt. Utviklingen på Mozilla fortsatte siden ufortrødent, men uten at det store gjennombruddet kom. Noen av utviklerne begynte etter hvert å mislike veien som Mozilla tok, og hevdet man forsøkte å gjøre for mye på en gang. Mange følte at programmet var i ferd med å bli altfor stort og tungt – og dermed lite konkurransedyktig. Resultatet ble dannelsen av en ny variant av Mozilla, basert på mye av det samme grunnlaget, men med et klarere fokus på nettlesing. Denne varianten ble kalt Firebird (senere Firefox) og har i dag forlengst passert sin forfader Mozilla i antall brukere. Per i dag er faktisk Firefox den eneste nettleseren som jevnlig tar markedsandeler fra Microsoft Internet Explorer, selv om sistnevnte fortsatt er den klart mest brukte nettleseren på verdensbasis. Senere har også et annet Mozilla-relatert prosjekt, e-postklienten Thunderbird, begynt å slå igjennom som et alternativ til de proprietære tyngre programmene, som for eksempel MS Outlook og Lotus Notes.

Det finnes per i dag en god del OS-prosjekter som har greid å spre seg utover den interne kretsen av hackere og blitt en suksess blant vanlige sluttbrukere. Flere av de største Linuxdistributørene har greid å etablere seg som seriøse leverandører av både tjener og klientløsninger, for eksempel Red Hat, SuSe, Mandrake og Ubuntu. De siste årene har også OS begynt å etablere seg i markedet for kontorapplikasjoner, først og fremst gjennom programvarepakken OpenOffice, som er delvis basert på Sun Microsystems StarOffice, og som støttes økonomisk av samme organisasjon. OpenOffice har vunnet mye på at den støtter filformatene til Microsofts flaggskip i kontormarkedet, MS Office, slik at det blir enklere for brukerne å emigrere. Flere og flere OS-prosjekter ser nå fordelen med å gjøre overgangen fra proprietær programvare til OS så enkel og smertefri som mulig Et typisk eksempel på dette er linux-distribusjoner som Linspire[4], der man har satset mye på interoperabilitet med Microsoft sine produkter. Et annet eksempel finner vi ved Mozilla[5] sine to nyeste flaggskip innen nettlesing og e-post, Firefox og Thunderbird. Disse to har innebygde verktøy for å hente inn bokmerker og adresselister fra andre kommersielle programmer, og spør om brukeren vil gjøre dette

[4]http://www.linspire.com
[5]http://www.mozilla.org

ved installasjon. Inntil OS har fått et mer solid fotfeste hos sluttbrukerne, er nok dette en god strategi for å overleve i et marked preget av sterk konkurranse og hurtig utvikling.

2.1.4 Fra tekst til multimedia; kampen om standarden

Med den eksplosive veksten av internett, har både tilbudet i tjenester og infrastruktur økt dramatisk i løpet av de siste årene. Dette har skapt mange nye muligheter i forhold til hva slags innhold som kan presenteres over nettet, spesielt med tanke på rike medier. OS-miljøene har på sin side fokusert mest på tekstlig kommunikasjon, det være seg asynkront (e-post, diskusjonsgrupper) eller synkront (pratekanaler ala Freenet-prosjektet). Etter hvert som flere og flere kommersielle aktører har begynt å operere på nettet, har behovet økt for å kunne presentere noe mer enn bare statisk tekst og bilde. Gjennom utviklingen av portable medier som for eksempel CD-ROM og DVD-ROM, har det blitt mulig å pakke inn informasjonen på nye måter, der tekst, lyd og bilde utfyller hverandre, såkalt *multimedia*.

De tradisjonelle åpne standardene for kommunikasjon over nett (http, smtp, ftp med flere), er på sin side ikke utviklet med tidskritiske applikasjoner for øyet, og egner seg derfor i mindre grad til å overføre lyd og bilde, der kravene til hurtig og stabil overføring øker dramatisk. For å kunne levere stadig mer avansert multimediainnhold over internett, har derfor mange kommersielle foretak forsøkt å etablere sine egne løsninger som de-facto standarder. For eksempel så har både Microsoft og Apple valgt å inkludere videokonferanse som en innebygd mulighet i sine operativsystemer, der metodene for overføring ikke er basert på noen internasjonal standard, men i stedet utviklet internt i bedriften. Apple har til og med inkludert den nødvendige maskinvaren, i form av integrert mikrofon og minikamera, i sin siste serie med personlige datamaskiner.

Denne utviklingen skjer imidlertid ikke i et vakuum, og bransjen har over tid etablert sine egne standardorganisasjoner, som har klare likhetstrekk med organisasjoner som for eksempel IETF (Internet Engineering Task Force). Den mest innflytelsesrike av disse er utvilsomt Motion Picture Experts Group (MPEG), som har utarbeidet spesifikasjoner for mange av de mest brukte metodene for lagring, komprimering og distribusjon av lyd og bilde. MPEG-gruppen lager ikke applikasjoner selv, men angir rammeverk og beskrivelser av hvordan en implementering skal fungere. Både DVD og MP3 er basert på standarder som MPEG-gruppen har satt ned, for å nevne noen. MPEG-gruppen er offisielt medlem av ISO, den internasjonale standardorganisasjonen.

Det som skiller MPEG-gruppen fra for eksempel IETF, er først og fremst rekruttering og

distribusjon. Innenfor IETF blir medlem på individuelt grunnlag, og får medbestemmelse ut i fra hvor stor innsats man har lagt ned. I MPEG-gruppen er det ikke individuelle bidragsytere, men heller representanter for store bedrifter eller bransjeorganisasjoner som deltar. I tillegg er MPEG-gruppens standarder lisensbelagt, noe som betyr at man må betale for å kunne bruke standardene i en eventuell implementering. IETF, og de andre frivillige standardorganisasjonene fra hacker-miljøene, har ingen slike heftelser ved sine standarder.

For OS-bevegelsen representerer utviklingen av multimedia en utfordring på to fronter. For det første er man redd for at de åpne standardene man benytter på internett i dag, skal bli erstattet av proprietære formater. Et eksempel på dette er den stadig mer utbredte bruken av applikasjonen «Flash» fra Macromedia[6]. Dette er et program man kan installere på sin maskin, som så kan vise nokså avansert multimedia gjennom en kompatibel nettleser. Problemet er at dette programmet forholder seg i liten grad til de andre standardene som har vært med å bygget opp internett. Dermed opererer det i praksis som en informasjonstunnell over internett, som andre applikasjoner i liten grad kan samhandle med. Både brukere og utviklere risikerer å bli «låst» til en form for kommunikasjon.

Microsoft er en annen sentral aktør i denne utviklingen, i og med at de har satset sterkt på å gjøre sine metoder for overføring av lyd og bilde til de-facto-standarder. Her har imidlertid MPEG-gruppen spilt en viktig rolle, siden deres offentlige standarder har fått et bredere nedslagsfelt og støtte i bransjen, enn det microsoft har klart å etablere på egen hånd. Den andre utfordringen for OS-bevegelsen er nettopp knyttet til MPEG-gruppen og lignende standard-organisasjoner. Selv om slike standarder er å foretrekke, fremfor den ensidige avhengigheten av en de-facto-standard fra for eksempel Microsoft, er det likevel stor skepsis blant OS-utviklere til denne utviklingen. En standard i OS-terminologi skal være åpen for alle, uavhengig av finansielle ressurser, slik at det blir en lavest mulig terskel for å benytte disse i utviklingen av ny teknologi. På grunn av de forholdsvis kostbare MPEG-lisensene, vil det ikke være realistisk for den typiske OS-utvikler å basere seg på slike standarder, siden det sjeldent er mye penger involvert i startfasen av et OS-prosjekt. Med dette som utgangspunkt er det nå flere OS-prosjekter som prøver å etablere åpne alternativer for behandling av multimedia, for å sikre at neste generasjon av internett-tjenester innehar den samme åpne og inkluderende arkitekturen som i dag. Blant disse har jeg i denne oppgaven valgt ut to prosjekter, Xiph og VideoLAN, som jeg vil presentere nærmere i kapittel 4.

[6]Se www.macromedia.com

2.1.5 Arbeidsform og organisasjon

Det finnes i dag et enormt antall utviklingsprosjekter som velger å kalle seg selv ”Open Source”. Sourceforge.net, som er en av flere OS-portaler for prosjektsider og filtjenere, nærmer seg nå 200 000 registrerte prosjekter. Mange av disse har et bevisst forhold til hva det vil si å være OS, og har satt seg inn i OSI sine retningslinjer og lisensforslag. Andre prosjekter har et mindre bevisst forhold til begrepet, og bruker det kun i den rent bokstavelige betydningen, det vil si at de har lagt ut sin kildekode til allmenn nedlasting. Det er også store variasjoner i bruk av verktøy, organisasjonsform, antall utviklere, utviklingstempo, forretningsmodell (hvis relevant) og lederskap. Likevel kan man finne mange fellestrekk ved denne heterogene massen med utviklere som alle seiler under OS-flagget. I dette kapittelet vil jeg presentere noen av de kjennetegnene som er typisk Open Source, og som er med på å definere hva begrepet og bevegelsen går ut på. Denne fremstillingen fortsetter også i neste kapittel, der blikket rettes mer spesifikt mot hvilke *verktøy* som ofte foretrekkes i OS-utvikling.

Eric Raymond har gjennom flere artikler og taler fått status som en av de viktigste talsmennene for OS-bevegelsen. I sin artikkel ”The Cathedral and the Bazaar” (Raymond 1997), som har fått tilnavnet ”hacker-bibelen”, lister Raymond opp en rekke punkter han mener er typisk for god programmering innenfor OS og ellers. Jeg vil trekke fram enkelte av disse som jeg synes er spesielt relevante for å forstå hva OS er, og for å få et innblikk i hvilke mål og kriterier den selvbevisste OS-utvikler gjerne opererer med. Den første regelen er grunnleggende og sier mye om hvordan et typisk OS-prosjekt dannes:

1. *Every good work of software starts by scratching a developer's personal itch.*

Med andre ord kan opprinnelsen til mange OS-prosjekter spores tilbake til et reelt problem som en utvikler har, og som han ønsker å gjøre noe med. Dette gir grunnleggeren for prosjektet en klar motivasjon til å drive det framover, fordi det vil på et eller annet vis hjelpe ham i hans daglige arbeid.

2. *Good programmers know what to write. Great ones know what to rewrite (and reuse).*

3. *``Plan to throw one away; you will, anyhow.''*[7]

De to reglene over forsøker å beskrive OS-utviklernes holdninger til kildekode, med vekt på viljen

[7]Opprinnelig hentet fra «The Mythical Man Month» Fred Brooks (1975)

til å stole på andres arbeid, samt ydmykhet i forhold til egne bidrags verdi. Eierskapsfølelse overfor kildekode blir sett på som usosialt og uproduktivt.

> *4. If you have the right attitude, interesting problems will find you.*
>
> *5. When you lose interest in a program, your last duty to it is to hand it off to a competent successor.*

Her beskriver Raymond hvordan OS-prosjekter kan holdes i live over flere generasjoner med utviklere. Det er interessant å merke seg at denne metoden er avhengig av at grunnleggerne av et prosjekt er villige til å gi slipp på eierskapet av det. Selvsagt er ikke alltid dette tilfelle, og resultatet blir gjerne en intern konflikt, som kan være kime til nye alternative prosjekter eller i verste fall forsure fellesskapet og dermed drepe prosjektets initiativ og drivkraft. Den generelle holdningen innenfor OS-bevegelsen, nemlig at kildekode ikke er et personlig, men et felles gode, hjelper likevel mye i den potensielt vanskelige prosessen med å skifte lederskap for et prosjekt. Blant kritikerne av OS-modellen, har det ofte vært fokus på nettopp lederskapet i et OS-prosjekt, og problemer knyttet til dette. Hvis hoveddrivkraften og motivasjonen for arbeid kun er basert på personlige preferanser, risikerer man at alle jobber kun med de teknisk utfordrende områdene av utviklingen, mens de mer rutinepregede og mindre spennende arbeidsoppgavene blir nedprioritert og hengende etter (Bezroukov 1999).

> *6. Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging.*
>
> *7. Release early. Release often. And listen to your customers.*
>
> *8. Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone.*
>
> *(...)*
>
> *10. If you treat your beta-testers as if they're your most valuable resource, they will respond by becoming your most valuable resource*
>
> *11. The next best thing to having good ideas is recognizing good ideas from your users. Sometimes the latter is better.*

Disse reglene tar for seg det som Raymond kaller den ytre ringen (engelsk: «Halo») rundt de viktigste programmererne i et prosjekt (Raymond 1997). Dette består av brukere, betatestere, og utviklere med mindre og sporadiske bidrag. Regel nummer 8 har vært gjenstand for spesiell

oppmerksomhet og har blitt popularisert ved uttrykket ”With enough eyes all bugs are shallow” (kjent som «Linus' Law»). Kritiske røster har kalt denne forestillingen naiv, både fordi det i praksis sjeldent er mer enn noen få personer som i realiteten jobber med samme kode, og fordi enkelte programfeil krever rigid og systematisk testing for å avsløres. Torchiano og Morisio (2004) viser at OS-applikasjoner i de aller fleste tilfeller brukes på samme måte som kommersielle produkter. Selv om koden er tilgjengelig, betyr ikke det at en stor andel av brukerne velger å laste den ned eller endre den til egne formål, dette er heller unntaket enn regelen. Det er likevel ingen tvil om at terskelen for å delta i utviklingen av et OS-prosjekt er mye lavere enn ved et kommersielt produkt. Samtidig vil det være grunn til å anta at de fleste som orker å sette seg inn i kildekoden for å forsøke å lokalisere en feil, vil inneha betydelig kompetanse (ellers ville de neppe gått løs på en slik oppgave). Så selv om antall øyne ofte vil være færre enn det Raymond beskriver med «Linus' law», vil muligheten for ekstern hjelp alltid være til stede og oddsene gode for at eventuelle bidrag som mottas vil være av god kvalitet. De fleste som bidrar i OS-utvikling er erfarne og profesjonelle programmerere (von Hippel & von Krogh 2003).

Reglene 6-11 ovenfor er alle med på å beskrive et av de viktigste kjennetegnene ved OS-programmering, nemlig den nære kontakten mellom bruker og utvikler. Skal et OS-prosjekt overleve over lengre tid, er det avhengig av å knytte til seg dyktige utviklere, og en effektiv måte å gjøre dette på er å ta brukerne på alvor, gi dem jevnlige oppdateringer om hva som skjer og oppmuntre dem til å delta selv. Dette krever en aktiv leder, med gode egenskaper som motivator, samt evne til å fungere som et faglig og sosialt samlingspunkt. Raymond peker også på en av de viktigste kriteriene for å knytte til seg dyktige utviklere; selve produktet. Uten at man kan presentere et mer eller mindre fungerende førsteutkast, som er godt sammensatt og har potensiale til å bli noe bra, vil man neppe kunne forvente den helt store oppslutningen om eget prosjekt (Raymond 1997). Problemene med å skaffe dyktige utviklere til sitt prosjekt, samt bygge et dynamisk og produktivt OS-fellesskap, bringer oss inn i diskusjonen omkring rekruttering og motivasjon. Hva motiverer den typiske OS-utvikler? Hva er det som får så mange talentfulle programmerere til å gi så mye av sin tid og kompetanse?

Bergquist og Ljungberg (2001) beskriver OS-miljøene som en spesiell form for gaveøkonomi, der man gjennom å gi til fellesskapet kan bekrefte eller forsterke sin status innad i miljøet. Jo flere gaver man gir til fellesskapet (som blir godt mottatt, vel å merke), jo mer får man tilbake i form av faglig anerkjennelse og innflytelse i miljøet. Hvis man føler seg som betydningsfull i miljøet, vil dette styrke følelsen av sosial tilhørighet, noe som igjen vil motivere til ytterligere innsats. I den

klassiske teorien om gaveøkonomi, formulert i Mauss' «The Gift» fra 1955, er utvekslingen av gaver en bekreftelse av eksisterende maktstrukturer, der det å gi en gave symboliserer underdanighet overfor mottakeren. Innenfor OS-miljøer ser dette ut til å være snudd på hodet. Siden kildekode svært enkelt lar seg kopiere og distribuere, er det ikke lenger mottakeren, men giveren, som blir den viktigste part i gaveutvekslingen. Hvis mottakeren tar imot gaven, gir dette økt status til giveren, fordi det symboliserer hvor betydelig hans bidrag i fellesskapet er. Mottakeren er imidlertid ikke uten makt. Han kan fortsatt velge å forkaste bidraget, for der i gjennom å befeste sin egen posisjon på bekostning av giveren (Bergquist & Ljungberg 2001).

En annen sentral motivasjonsfaktor for deltakelse i et OS-miljø, er den faglige selvutviklingen som gjøres mulig, ved at man får testet ut sine ideer og tanker blant andre kompetente utviklere. Læringsgevinsten ved å delta i et OS-miljø kan være betydelig. I tillegg er det ofte personlige interesser inne i bildet, ved at utviklerne selv har behov for å bruke kildekoden på sin arbeidsplass eller i tilknytning til et eksternt prosjekt (von Hippel & von Krogh 2003).

Etter å ha samlet et knippe dyktige utviklere, er det fortsatt mye som gjenstår før det kan karakteriseres som et OS-prosjekt. Et typisk trekk ved OS-utvikling er fellesskapet som dannes omkring produktet som utvikles, og at nye medlemmer blir sosialisert inn i en kultur basert på deling og åpenhet. Georg von Krogh (2003) har studert utviklingen av FreeNet, et gruppebasert åpent kommunikasjonssystem, der samtalene foregår via direkteoverført tekst[8]. Han bruker begrepet «Joining script» til å beskrive den prosessen alle nybegynnere må igjennom, hvis de ønsker å bli en del av utviklergruppa. Hvordan disse skriptene arter seg, vil variere fra prosjekt til prosjekt, ut i fra hva slags aktivitetsnivå og hva slags type aktiviteter som forventes av lederen og de andre aktive bidragsyterne. Ved gjentatte bidrag av høy kvalitet, som harmonerer godt med miljøets forventninger, vil en nybegynner kunne få direkte tilgang til kodebasen og dermed bli tatt inn i varmen som en del av utviklergruppa (von Krogh 2003).

En viktig del av sosialiseringsprosessen for nye medlemmer, er tilpasningen til OS-miljøets kommunikasjonsform. Innenfor OS-tradisjonen har det lenge vært et betydelig fokus på høvisk og korrekt opptreden på nettet. Uansett hvilken arena for kommunikasjon man operer i, vil det finnes uskrevne (og av og til skrevne) regler for når man kan snakke og hvordan man uttrykker seg. I tillegg vil den enkelte organisasjon utvikle mer spesifikke varianter av disse reglene via deres egen praksis. I forbindelse med kommunikasjon over internett, snakker man gjerne om «nettikette» - en

[8]Internet Relay Chat (IRC). Dette kommunikasjonsverktøyet blir presentert mer utførlig i neste kapittel

omskriving av uttrykket «etikette», som forbindes med høvisk og respektfull opptreden[9]. Typiske kjøreregler innenfor «nettikette» går på hvor ofte man snakker, hva slags tone og språkbruk som er akseptert, og hvilke tema som kan tas opp. I tillegg kan selve skrivemåten bli gjenstand for negativ oppmerksomhet, hvis man for eksempel benytter svært sammentrekte ord og uttrykk («SMS-språk») eller bruker store bokstaver, effekter eller farger for å bli lagt merke til. Repetisjoner eller overdreven ordflom (såkalt «spamming»), blir som regel slått hardt ned på i de aller fleste OS-miljøer.

Ingen kan selvsagt gardere seg mot enkelte forstyrrende og uproduktive elementer i en organisasjon, så det vil alltid være en lederutfordring å sikre en god og produktiv stemning i fellesskapet. Alan Cox, en anerkjent programmerer innenfor OS-miljøet, trekker fram *uproduktive synsere* som et klassisk problem i en hver utviklingsgruppe. Han beskriver hvordan slike personer kan lamme et ellers dynamisk utviklingsmiljø og råder ledere til å «ta vare på de som spør 'Hvordan kan jeg...', og passe seg for de som sier 'Vi burde...'» (min oversettelse). En måte slike problemer blir taklet på i OS-miljøer, er rett og slett at flere av de viktige bidragsyterne begynner å ignorere en eventuell «vanskelig» person, noe som raskt skaper en presedens i miljøet og i praksis hindrer vedkommende i å delta i fellesskapet (Cox 1998). Det er med andre ord viktig i et OS-fellesskap at både ledelsen og hver enkelt bidragsyter innehar gode evner til å skille mellom usaklig støy og konstruktiv kritikk (Bergquist & Ljungberg 2001).

2.1.6 Verktøykassa

Det finnes en rekke verktøy til hjelp for programmering, samhandling og kommunikasjon innenfor systemutvikling. De fleste slike verktøy gir mulighet for arkivering av informasjonen de behandler, slik at viktige erfaringer og ny kunnskap kan gjøres nytte av i ettertid. OS er intet unntak i så måte, tvert imot er det en utbredt tradisjon innenfor OS at man benytter ulike informasjonsarkiver i sitt daglige arbeid. Jeg vil her presentere noen slike verktøy ut i fra fem grove kategorier av systemarbeid; *versjonskontroll*, *feilbehandling*, *distribusjon* og *kommunikasjon*:

Versjonskontroll:
Et prosjekt basert på tradisjonen rundt OS, tilknytter seg utviklere etter interesse og ferdigheter. Man kontrollerer i mindre grad hvem som får lov til å arbeide med hva, og det vil kunne utvikle seg parallelle versjoner av samme kodesnutt. For å holde koden til prosjektet intakt og fungerende, uten at utviklere til stadighet tråkker hverandre på tærne, trenger man et verktøy til administrasjon av

[9]For flere netiketteregler, se for eksemper http://www.learnthenet.com/english/html/09netiqt.htm

koden. CVS (Concurrent Versions System) og Subversion er to slike verktøy som brukes av svært mange OS-prosjekter.

I tillegg til å sikre kodens integritet ved flere utvikleres bidrag, etableres det gjennom verktøyene et arkiv over endringer, der man kan rekonstruere koden basert på for eksempel hvem som har endret, tidspunkt eller andre kriterier man definerer selv. Ved uenigheter om hvilken retning programvaren skal ta, kan man opprette ulike grener innad i kodebasen, slik at parallell utvikling ikke skaper problemer i forhold til versjonshåndteringen. I større prosjekter utnevnes gjerne noen utvalgte erfarne utviklere til portvakter, med ansvar for hver sin del av kodebasen, som vi vil se eksempel på i presentasjonen av mitt hovedcase. Dette gjøres blant annet for å hindre at prosjektet spriker i alt for mange retninger. Når nye bidrag blir lagt inn i kodebasen, følger det gjerne med en kommentar eller metatekst som beskriver hva bidraget går ut på. Disse kommentarene kan være interessante i forhold til å forstå arbeidsflyten og –fordelingen mellom utviklerne.

Feilbehandling:
Som nevnt tidligere trekker Eric Raymond fram feilsøking og behandling av problemer som et område der OS-modellen gjerne har et fortrinn i forhold til tradisjonell systemutvikling. Med utsagnet ”With enough eyes all bugs are shallow” (Raymond 1997), peker han på hvordan åpen kode og mange utviklere er med på å begrense muligheten for at feil slipper gjennom usett. For at dette skal være praktisk mulig, kreves det imidlertid at man har verktøy for å rapportere og behandle feil etter hvert som de blir oppdaget. Til dette formålet brukes ofte en database med et grensesnitt mot internett, der hvem som helst kan lete etter tidligere rapporterte feil, sjekke status og rapportere nye feil.

Mozilla Foundation er en organisasjon som ble opprettet for å lage en OS-versjon av nettleseren Netscape. For å holde styr på feilrapporter og andre forespørsler, ble det utviklet et rammeverk kalt Bugzilla[10], som benyttes av svært mange prosjekter, både med åpen og lukket kode. Typisk for disse rammeverkene er at alle feil og mangler, samt saksgangen tilknyttet disse, bli publisert fortløpende og automatisk av systemet. Dermed skal det være mulig for brukeren selv å sjekke om et eventuelt problem er kjent, om det finnes en midlertidig løsning eller generelt hva som er status på saken.

TRAC er et annet slikt verktøy som, i tillegg til feilbehandling, tilbyr moduler for prosjektstyring. Dette skjer via oppsetting av framdriftsplaner og milepæler, som skal gjøre det enklere å få oversikt

[10] http://bugzilla.org

over både den planlagte og de reelle fremgangen i prosjektet.

Hvis man er opptatt av hvordan et prosjekt håndterer endring, feil eller mangler, kan slike arkiver gi en god pekepinn. Faktorer som responstid, holdning til brukere, endringsvillighet, stabilitetskrav og sikkerhetsfokus kan skinne igjennom ved en nærmere kikk på saksgangen i en slik feildatabase.

Distribusjon:
Enkelte utviklingsprosjekter retter seg i første rekke mot andre utviklere, og tar derfor for gitt en anselig grad av kompetanse hos sitt publikum. Slike prosjekter vil gjerne legge liten vekt på å distribuere kompilert kode, men heller henvise sine brukere til kildekoden, ofte i form av en gjestekonto til et versjonskontrollsystem som CVS for eksempel. Hvis et prosjekt imidlertid retter seg mot andre brukere enn de mest teknisk kompetente, vil det være naturlig å distribuere kompilerte pakker for et eller flere operativsystemer. En vanlig praksis er å legge filer tilgjengelig til nedlasting på en nettside. Denne praksisen har etter hvert blitt standardisert via paraplynettsider som sourceforge.org og freshmeat.org. Disse nettsidene fungerer som portaler og tilbyr tjenerplass for filnedlasting og en standardmal for presentasjon av prosjektet. Enkelte tilbyr også prosjekthåndtering og informasjonstjenester som forum eller nyhetsgeneratorer.

De mest avanserte nettportalene kan prosessere og presentere statistikk om alle sine prosjekter til brukerne. Eksempler på slik statistikk kan være aktivitetsnivå, antall utviklere og antall kodelinjer. Hvis man ønsker å danne seg en oversikt over hvordan det generelle OS-landskapet ser ut, er slike portaler et godt utgangspunkt. Når det er sagt må man ikke overvurdere verdien av den statistikk som aggregeres fra slike portaler, ettersom det kun er en av flere kanaler for distribusjon. Det er heller ingen grunn til å ta for gitt at informasjonen som legges inn på slike nettsider holder en høy kvalitet eller representerer prosjektet på en treffsikker og oppdatert måte.

Kommunikasjon:
I internettets barndom, før hypertekst og multimedia, var elektronisk post og diskusjonsgrupper de rådende verktøy for samhandling og kommunikasjon. På tross av en rivende utvikling innen tjenester og multimedia på internett, har disse verktøyene beholdt sin funksjon – om enn ikke uforandret. Ved å opprette epostlister man abonnerer på, kan forskere og studenter over hele verden på en enkel måte holde en faglig eller triviell diskurs gående med svært mange ”tilhørere” samtidig. All informasjon i en slik liste blir lagret på en sentral server og man kan dermed hente ned innlegg enkeltvis, i magasinformat på for eksempel en måned av gangen, eller søke i arkivene etter

spesifikke tema. Fortsatt i dag er dette den primære formen for kommunikasjon i mange OS-prosjekter, og på samme tid den primære kilden til informasjon for både utviklere og brukere. Det er imidlertid en tendens til at epostlister blir utviklernes arena, mens brukerne flytter til et annet beslektet medium: diskusjonsforumet.

USENET satte standarden for mange-til-mange asynkron elektronisk kommunikasjon, med viktige funksjoner som gruppemedlemskap, tråding av innlegg i trestrukturer, lokal synkronisering og søking i arkiv. På 1990-tallet spredte bruken av internett seg ut over universitetsmiljøene og til den vanlige forbruker via de første nettleserne og fremveksten av World Wide Web (WWW). Bedre nettverksteknologi og billigere hjemmedatamaskiner bidro nok også til å akselerere denne utviklingen. Resultatet har blitt en dalende aktivitet på USENET til fordel for diskusjonsforum over WWW. Disse forumene ligner på USENET i at de støtter en del av den samme funksjonaliteten, men vises ikke i en egen klient, men heller gjennom en nettleser over http-protokollen. Dette gjør dem mer fleksible i forhold til funksjonalitet og lettere tilgjengelig for mindre teknisk anlagte brukere, noe som kan være med på å forklare hvorfor de er blitt så populære. Mange OS-prosjekter bruker slike forum som kontaktpunkt til brukerne, der man kan finne svar på ofte spurte spørsmål (FAQ), diskutere faglig relevant stoff og spørre om hjelp eller råd. De aller fleste av disse kanalene er fullstendig åpne for publikum. Det vil som regel være moderatorer i slike kanaler som tar kontroll hvis noen ikke følger etablerte sosiale spilleregler, men hver enkelt kan også bruke egne filtre for å unngå eventuelle uønskede og plagsomme personer.

Sanntidskommunikasjon, som IRC (Internet Relay Chat) og MSN (MicroSoft Messenger), har blitt stadig mer populært ettersom internett har vokst i omfang og tilgjengelighet. Mens MSN og lignende programmer (ICQ, IChat, Jabber m.fl.) i første rekke er laget for kommunikasjon mellom to eller flere enkeltbrukere, er IRC bygget opp etter tildels de samme prinsippene som USENET, der grupper er blitt erstattet med sanntidskanaler. De aller fleste OS-prosjekter har enten en egen kanal på IRC, eller deler en kanal med andre tilsvarende prosjekter. Det er som regel ikke samme interne krav til kvalitet i det som presenteres på disse kanalene i forhold til epostlister og forum, ettersom kommunikasjonen er i sanntid, og minner mer om direkte tale i formen (Paolillo 1999).

Organisasjonens «ansikt utad» blir typisk representert over internett ved hjelp av en offisiell hjemmeside. Hvis prosjektet er forholdsvis lite eller nylig oppstartet, er det ganske vanlig å bare benytte en paraplyside noe ala sourceforge.net, der man kan presentere grunnleggende informasjon om prosjektet (kontaktinformasjon, deltakere, nyheter m.m.). Ved mer omfattende prosjekter er det

normalt å i tillegg ha en egen hjemmeside, med mer inngående beskrivelser av prosjektets innhold og status. Det er også vanlig å gi korte svar på spørsmål som enten er blitt stilt ofte, eller som prosjektet antar vil komme til å bli stilt (som regel kalt en FAQ = Frequently Asked Questions). Som en utvidelse av denne tankegangen har det etter hvert blitt mer og mer vanlig å samle all relevant informasjon om et prosjekt i et standardisert rammeverk for publisering på nett. Dette rammeverket kalles en Wiki (av uttrykket wiki-wiki, som betyr «raskt» eller «kjapt» på Hawaii, og er hentet fra ruteplakatene til bussene på den internasjonale flyplassen i Honolulu). Det finnes mange varianter av Wiki-er, men de aller fleste har disse fellestrekkene: a) man kan enkelt editere innholdet via et webgrensesnitt, b) det er brede rettigheter til å endre innholdet (enkelte har ingen begrensninger), c) full historikk på endringer er tilgjengelig og websidene kan enkelt gjenopprettes tilbake til en eldre utgave.

Et interessant eksempel på en Wiki er det raskt voksende prosjektet Wikipedia. Dette er et flerspråklig og tematisk ubegrenset nettbasert oppslagsverk, der alle har fulle rettigheter til å legge inn eller endre artikler. Wikipedia ble startet i 2000 og inneholder per i dag over tre og en halv million artikler (hvorav over en million av disse er i den engelske utgaven) noe som gjør det til verdens største oppslagsverk. I januar 2005 var det 30,000 forfattere med fem eller flere artikler på samvittigheten, noe som vitner om hvor mange bidragsytere dette prosjektet har tiltrukket seg på forholdsvis kort tid. For de fleste organisasjoner vil man imidlertid ha en tematisk begrensning og et ønske om større kontroll på det som skal presenteres, slik at mengden innhold på Wiki-en vil være kraftig begrenset i forhold til Wikipedia. Alle som setter opp en Wiki må nødvendigvis ta en vurdering i forhold til kontroll kontra åpenhet, ut i fra hvilken hensikt Wiki-en er ment å utføre – på en skala fra ensidig presentasjon til offentlig debattarena.

2.2 Tradisjoner innenfor IT-forskning

Forskingen på informasjons- og kommunikasjonssystemer kan deles inn tre hovedretninger, Datateknikk (CS - Computer Science), Systemutvikling (SE - System Engineering) og Informasjonssystemer (IS – Information Systems). Dette skillet er ikke universelt, i den forstand at du vil neppe finne det samme mønsteret på alle universiteter eller i alle forskningsmiljøer rundt om i verden. Enkelte steder er det overlapping, på andre er det ikke forsøkt delt, eller en eller flere deler mangler helt. Glass, Ramesh & Vessey (2004) har kategorisert et stort antall publikasjoner i forhold til tredelingen CS/SE/IS, og på dette grunnlaget bidratt til å tydeliggjøre skillet mellom de tre grenene av IT-relatert forskning:

CS finner man som regel i de teknisk-naturvitenskapelig orienterte fagkretsene og temaer her omhandler i første rekke maskinvarearkitektur, operativsystemers oppbygning, og konstruksjon av algoritmer og datastrukturer. Typisk for denne genren er et fokus på hvordan det ideelle system bør arte seg, der bruk av matematikk og/eller fysikk er sentralt i bevisføringen. Design- og implementeringsprosesser, eller reell bruk av et system, blir svært sjeldent tatt opp som tema.

SE har tradisjoner innenfor samme fagkrets som CS, men beveger seg ut fra den rene tekniske sfæren og mer inn på det feilbarlig menneskelige aspektet ved systemutvikling. Hovedtyngden av publikasjoner herfra omhandler metoder og teknikker for utvikling av datasystemer, inkludert design, modellering, «best-practice», testing og bruk av hjelpeverktøy. I tillegg har SE også et ben innenfor «management»-miljøene, der prosjektstyring, evaluering, og risikohåndtering er sentrale tema (Glass, Ramesh & Vessey 2004). Det mest kjente bidraget fra denne tradisjonen er nok Royces ”Fossefallsmodell” fra 1970, revidert av Boehm seks år senere. Denne forholdsvis enkle lineære modellen har hatt stor betydning for design og prosjekthåndtering av IT-prosjekter både i offentlig og privat sektor. Fordelene med modellen var blant annet at man fikk bedre kontroll over utviklingsprosessen ved å dele den opp i klart definerte faser (David, Bershoff & Comer 1998). Det finnes mange måter å løse dette på, men de fleste forholder seg til følgende faser:

1. Innhenting og fastsetting av spesifikasjoner
2. Design og planlegging av systemets komponenter, samt koblingene mellom disse.
3. Implementering av det planlagte systemet
4. Testing og debugging
5. Utrulling og vedlikehold

Det er imidlertid flere svakheter, eller fallgruver, knyttet til denne modellen. Den gir for det første lite støtte til håndtering av endringer i spesifikasjoner underveis. For det andre har den som premiss at kunden er i stand til å gi nøyaktige spesifikasjoner på forhånd, noe som sjeldent er tilfelle. Kunden har gjerne ikke den nødvendige tekniske eller organisatoriske oversikt som kreves for å gi systemutvikleren presise nok spesifikasjoner. Til sist er modellen lite fleksibel i forhold til å jobbe parallelt, og kan føre til uproduktive pauser fordi en fase må ferdigstilles før den neste kan begynne (Pressman 1997).

Det har i senere år dukket opp flere forsøk på å møte utfordringene knyttet til fossefallsmodellen.

En ofte brukt tilnærming er å bruke prototyper, det vil si at man lager en enkel fremstilling av hvordan det ferdige systemet vil arte seg, og tester dette ut på den aktuelle brukergruppen. Brooks (1995) beskriver bakgrunnen for bruken av prototyping ganske treffende:

> *"The management question, therefore, is not whether to build a pilot system and throw it away. You will do that. The only question is whether to plan in advance to build a throwaway, or to promise to deliver the throwaway to customers…"*

Prototyping gir altså en åpning for tilbakemeldinger fra brukere på et tidlig stadium i prosessen. Dermed kan man hindre at feilskjær i designfasen blir brakt videre inn i det ferdige systemet. Her finnes det flere tilnærminger til hva man velger å prototype, ut i fra hva slags generell utviklingsmodell man benytter. Innenfor tradisjonell lineær systemutvikling vil man gjerne prototype de komponentene man er mest usikre på, for der i gjennom å komplettere de eksisterende designplanene man har utarbeidet på forhånd.

En mer radikal retning innen systemutvikling, såkalt *evolusjonær prototyping*, har et ganske annet utgangspunkt. Her starter man med å prototype kjernefunksjonene i et nytt system, på et så tidlig stadium som mulig. Tanken er at man begynner med å få opp basisfunksjonalitet og deretter bygger ut nye funksjoner etter hvert som behovene blir klarere og systemets ryggrad blir mer moden og robust. Systemet blir ikke definert og planlagt fullt ut på forhånd, men vokser i takt med kundens skiftende behov og bruksmønster. Dette gjøres gjennom korte utviklingsykluser og mange mindre oppdateringer og endringer, slik at funksjonaliteten økes i små steg av gangen.

David, Bershoff & Comer (1998) har sammenlignet ulike modeller for systemutvikling i forhold til hva som logisk sett bør gi det beste resultat for både bruker og utvikler. Viktige parametre i undersøkelsen var blant annet hvor lang tid man bruker på å få en fungerende versjon på beina, og hvor raskt systemet kunne tilpasses skiftende behov etter utrulling. De fremholder at systemutvikling basert på evolusjonær prototyping vil ligge nærmere kundens behov til alle tider i systemets livsløp enn ved en klassisk lineær modell. Tiden fra planlegging til første versjon av systemet er i kundens hender vil være kort, og kunden vil i de fleste tilfeller være mer fornøyd med et begrenset system som fungerer, enn med et omfattende system som ikke treffer hans eller hennes behov. I tillegg vil den evolusjonære tilnærmingen åpne for at nye funksjoner som introduseres er tilpasset kundens reelle behov per i dag, i stedet for å være bundet til statiske designplaner utarbeidet på forhånd (David, Bershoff & Comer 1998).

Det er flere trekk ved typiske Open Source-prosjekter som minner om evolusjonær prototyping. Det er en utbredt tradisjon innenfor OS at man gir ut kode på et tidlig stadium i prosessen, ofte før den har noen form for stabil funksjonalitet. Deretter gis det ut nye versjoner hver eneste natt, basert på siste endringer. Disse nattlige utgivelsene blir ikke annonsert som offisielle utgivelser, men er ment for å gi folk muligheten til å teste de siste endringene og nyeste funksjonene til programvaren. De offisielle utgivelsene vil komme med lengre mellomrom, avhengig av hvert enkelts prosjekts utviklingstempo og interne krav til testing og stabilitet. Offisielle utgivelser deles ofte opp i en stabil utgivelse, og en eller flere nyere versjoner (beta, alpha etc) med økt funksjonalitet og/eller bedre ytelse, men med større risiko for uoppdagede feil i koden. Dermed har brukerne av OS-programmer selv muligheten til å ta en vurdering i forhold til hvor stor risiko de er villige til å ta for å oppnå ny funksjonalitet eller ytelse.

Den historisk sett ferskeste grenen av IT-forskningen omhandler informasjonssystemer (IS). Her har den *reelle bruken* av IT, og konsekvenser knyttet til denne, kommet frem i mye større grad enn ved tidligere forskning. Typisk for IS-publikasjoner er også at de benytter seg av teorier og metoder fra andre fagfelt, langt oftere enn hva tilfellet er for CS og SE. Spesielt enkelte teorier innenfor humanistiske vitenskaper som sosiologi, psykologi og antropologi har fått et fotfeste innenfor IS-tradisjonen. Denne avhandlingen faller naturlig inn under IS-grenen av IT-forskning, ettersom jeg fokuserer på mange av de ikke-tekniske aspektene ved fenomenet som undersøkes. Dette innebærer ikke at litteratur fra de to andre grenene har blitt ignorert, men at jeg fra forventer å finne mest relevant stoff innenfor IS, og derfor vil konsentrere meg om denne tradisjonen i den videre fremstillingen.

2.3 Hvordan kunnskap kan kommuniseres

Med fremveksten av massemedier og internett har mengden informasjon den enkelte må forholde seg til økt drastisk. Uttrykket «informasjonsrevolusjon» forsøker å fange opp dette fenomenet, som virker å ha satt sitt spor innenfor de fleste industrilands samfunnsområder. For svært mange kommer dette klart til uttrykk på jobben, ved at arbeidsoppgavene blir mer kunnskapsintensive, og kravene til omstillingsevne og faglig utvikling øker. Et stadig mer komplekst samfunn bidrar samtidig til å øke behovet for tverrfaglige løsninger, noe som igjen tvinger fram et fokus på effektiv kommunikasjon og gruppedynamikk. Det er derfor ikke overraskende per i dag at en rekke publikasjoner innenfor IS-tradisjonen velger å se på hvordan teknologi kan støtte opp om kunnskapsarbeid i organisasjoner, der ikke-tekniske faktorer, inkludert blant annet kommunikasjon

og læring, blir gitt en fremtredende rolle.

2.3.1 Kunnskap via teori eller praksis

Hva er kunnskap? Det finnes sannsynligvis like mange definisjoner som det finnes forskere og filosofer. Kunnskap kan for eksempel være ...

> *..."information combined with experience, context, interpretation, and reflection. It is a high-value form of information that is ready to apply to decisions and actions." (T. Davenport et al., 1998)*

Med denne definisjonen av kunnskap forankrer Davenport begrepet tydelig i et praktisk fremfor et rent teoretisk perspektiv. Informasjon blir ikke kunnskap før det (potensielt) kan omdannes til konkrete beslutninger og handlinger i en relevant kontekst. Vektleggingen av erfaring og kontekst er viktig i denne definisjonen, fordi det forhindrer at man kan redusere kunnskap til en ren teoretisk øvelse. For mange vil dette intuitivt virke litt merkelig. Kunnskap blir ofte forbundet med boklig lærdom, i motsetning til erfaring og praksis, som man opparbeider seg ute i felten. Nyere studier har vist at det svært ofte er et betydelig avvik mellom den informasjon som er beskrevet i manualer og plandokumenter, og det arbeidet som faktisk gjøres i en organisasjon. I tillegg er det et svært sjeldent at ledelsen er villig til å revurdere sine planer ut i fra avvikende adferd «på gulvet» (Hatlig & Sørensen 1998). Denne tankegangen har vokst frem, sakte men sikkert, etter at man for alvor begynte å skille mellom de som planlegger og de som utfører arbeidet. Under middelalderen hadde man byggherrer ute i felten, som fulgte arbeidsprosessen og tok ad-hoc beslutninger basert på de erfaringene som ble gjort. Etter hvert ble det imidlertid mer og mer vanlig å trekke byggherren ut av den praktiske implementeringsfasen, og dermed redusere denne til en ren planlegger; en arkitekt (Turnbull 1993). Et kognitivt skille mellom teoretisk og praktisk kunnskap vokste med andre ord frem parallelt med vitenskapens oppblomstring i kjølvannet av middelalderen. Samtidig mistet den praktiske kunnskapen mer og mer av sin status, siden den i økende grad ble assosiert med de lavere sosiale klassene i samfunnet.

For å illustrere hvordan denne kløften mellom teoretisk og praktisk kunnskap påvirker oss i dag, kan vi ta for oss byggingen av en fiktiv moderne bro. Her foregår gjerne prosessen i to atskilte faser. Først en ren teoretisk fase, der arkitekten regner seg fram til hvordan broen skal se ut, og lager detaljerte tegninger og planer for byggingen. Deretter kommer byggefasen, den praktiske fasen, der planene skal settes ut i livet. Det er nettopp i denne siste fasen, når teori møter praksis, at man støter

på de tyngste utfordringene, med det største potensialet for læring og tilegnelse av ny kunnskap. Turnbull (1993) mener vi må begynne å tenke på arbeidsplassen som et laboratorium, og anerkjenne det vitale kunnskapsarbeid som finner sted i praktisk utførelse av en jobb. Ved å legge for stor vekt på planlegging og detaljstyring, risikerer man å kneble potensialet for innovasjon som ligger latent på enhver arbeidsplass. Nonaka & Takeuchi (1998) kritiserer vestlig forskning og arbeidsliv for å fokusere for mye på teoretisk, eller *eksplisitt,* kunnskap, der tall, regler og plandokumenter blir satt i høysetet. De hevder at vesten ofte overser den «tause» kunnskapen, som er like grunnleggende og like verdifull som den eksplisitte. Taus kunnskap kan i følge Nonaka & Takeuchi opptre i form av subjektiv innsikt, intuisjon eller magefølelse, og har dype røtter i personlige handlinger og erfaringer (”know-how”), samt verdinormer, ideer og følelser. Bygningsarbeideren, fra eksempelet over, vil gjennom byggingen av broen opparbeide seg mye slik kunnskap, som ikke uten videre lar seg dokumentere på en tilfredsstillende måte. De som arbeider tett sammen med bygningsarbeideren vil likevel snappe opp mye av hans innsikt gjennom (1) å høre på hans fortellinger fra byggefasen, eller gjennom (2) konkrete eksempler fra hans arbeidspraksis. Disse måtene å overføre kunnskap på vil jeg gå nærmere inn på i de to neste kapitlene.

2.3.2 Fortellinger som ressurs

Der mennesker lever i samkvem, vil man finne fortellinger. Fra tilsynelatende banale hverdagshistorier til utsvevende eventyr, sagn og fremtidsvisjoner. Den anerkjente franske språkforsker og litteraturkritiker Roland Barthes (1915-1980) hevdet at fortellinger er «international, transhistorical, transcultural: it is simply there, like life itself» (Phillips & Hardy 2002). Fortellinger er overalt rundt oss og fyller mange ulike funksjoner, som underholdning, propaganda, oppdragelse og identitetsbygging for å nevne noen. Gjennom hele livet blir vår kulturelle identitet og kompetanse dannet gjennom de fortellinger vi tar til oss og gjør til våre egne; fra eventyrene vi fikk høre når vi var små, til Dagsrevyens katastrofer og reklamens små suggestive glimt av materiell lykke.

Når det gjelder kunnskapsoverføring i organisasjoner, er det viktig å identifisere fortellinger som den ressursen de faktisk er. Historiene som kollegene utveksler over lunsjbordet eller underveis i arbeidet, må ikke avspises som uproduktive, selv om de er tilsynelatende ustrukturerte og lite målfokuserte. Orr (1996) viser i sin studie av kopimaskinreparatører hvordan uformelle tekniske diskusjoner oppstår gjennom daglig sosial omgang. Han kaller disse fortellingene for «krigshistorier» (engelsk: war stories), med henblikk på reparatørenes opplevelser av å måtte

kjempe mot gjenstridige maskiner. Tidligere erfaringer blir her presentert i en anekdotisk form, med kontekst og tekniske detaljer innvevd. Hvor mye detaljer og kontekst som tas med, vil avhenge av hvilken situasjon fortelleren befinner seg i, men som absolutt minimum vil fortellingene alltid inneholde tre typer informasjon: *Hvor* et problem har oppstått, *hva* problemet gikk ut på, og *hvordan* problemet ble løst. Ved å utveksle denne typen fortellinger, enten over lunsjbordet eller under selve feilsøkingen, sikrer reparatørene at deres kunnskap blir spredt til sine kolleger. Dette er spesielt viktig i forbindelse med overføring av kunnskap mellom eldre, erfarne reparatører og ferske rekrutter i organisasjonen (Orr 1996).

Et annet viktig aspekt ved fortellinger, er deres funksjon som sosialt lim og kulturbygger. Fortellinger kan formidle lett gjenkjennbare situasjoner, i den hensikt å oppnå trøst eller forståelse fra sine likemenn (felles skjebne, felles trøst). Situasjoner som opptrer som gjengangere kan bli gjenstand for humoristisk oppmerksomhet, noe som igjen styrker den kulturelle tilknytningen mellom de som arbeider sammen. For enkeltindivider i en arbeidsgruppe kan fortellinger være en måte å få vist seg fram på, som et verdig medlem av gruppa. Fortelleren søker sosial og faglig bekreftelse gjennom sin historie. Slik bygger fortellingene tette bånd mellom mennesker som arbeider sammen, og gir dem viktig kulturell ballast til å kunne utveksle både eksplisitt og taus kunnskap seg i mellom. Ofte vil det for utenforstående være vanskelig å se verdien og meningsinnholdet i slike fortellinger, fordi de er mettet med kulturelle symboler og henvisninger til andre fortellinger[11]. Orr (1996) presiserer på sin side at historiefortelling bør sees på som arbeid *i seg selv* – et arbeid som man gjør dumt å undervurdere betydningen av.

OS-utviklere som er aktive på IRC, diskusjonsfora og e-postlister bruker fortellinger aktivt i sitt daglige arbeid. Det er særlig i de kanaler der formen på samtalen er løs og lite definert at man kan se dette i praksis. Jo mer rom det er for uformelle, spontane samtaler og diskusjoner, jo oftere vil man se kunnskap bli overført på denne måten.

2.3.3 Kunnskap gjennom eksempler og maler

Taus kunnskap, opparbeidet gjennom praktisk arbeid, kan altså fanges opp i de fortellingene som utveksles på arbeidsplassen. Dette er imidlertid ikke den eneste måten å overføre taus kunnskap på. Hvis vi går tilbake til bygningsarbeideren i eksemplet mitt, kan vi merke oss hvordan han bruker konkrete eksempler som verktøy for å takle nye problemer, eller for å formidle generell kunnskap.

[11]Orr (1996) kaller dem *elliptiske* fortellinger; det vil si at de viser til andre fortellinger som tilhørerne forventes å kjenne til.

Ved å vise til hvordan den forrige broen ble bygget, og relatere dette til en ny kontekst (en ny bro for eksempel), kan arbeideren gi seg selv og sine kolleger viktig innsikt i byggeprosessen, og potensielt unngå en del av de fallgruvene som ble oppdaget sist. Ofte vil den nye konteksten sette tidligere erfaringer i nytt lys, og dermed generere ny kunnskap blant arbeiderne. I mange tilfeller vil tidligere erfaringer bekreftes, også i den nye konteksten, og bygningsarbeideren vil opparbeide seg ett sett med tommelfingerregler. Disse reglene vil være knyttet til konkrete eksempler, samt bli gjenstand for konstant revisjon gjennom det daglige arbeidet, ettersom reglene blir forsøkt strukket og kombinert i prosessen med å løse nye praktiske problemer. Jo større frihet bygningsarbeideren har til å eksperimentere slik i sitt daglige virke, jo større blir potensialet for utvikling av ny kunnskap i form av nye tommelfingerregler, eller *maler* om man vil.

Turnbull (1993) har studert hvordan de store gotiske katedralene i Europa ble bygget på 13- og 14-hundretallet. For mange historikere fremstår disse monumentale byggverkene som arkitektoniske mysterier, først og fremst fordi vitenskapen ikke var kommet langt nok på den tiden til å utføre de beregninger som trengtes for å konstruere byggverk av en slik størrelse. Hvordan kunne arkitektene unngå at byggverkene rett og slett ramlet sammen, når de ikke hadde den nødvendige kunnskapen? Turnbull hevder svaret er enkelt, det fantes ingen arkitekter i moderne forstand, og det fantes i liten grad detaljerte tegninger eller planer. Ved å ta gotiske byggverk nærmere i øyesyn, ser man fort at disse i liten grad kan være et produkt av en samlet plan. Variasjonen i stilart og form er slående, og man finner bokstavelig talt flere lag med byggeskikker utenpå hverandre. Dette kom av at katedralene ble til gjennom et utstrakt samarbeid mellom et stort antall byggmestere, som alle hadde sine sett med erfaringer og tommelfingerregler. Disse reglene ble gjerne formulert som geometriske maler, som man anvendte på en rekke konkrete problemer i løpet av byggefasen. Av og til gikk det galt, men da slo man hodene sammen og eksperimenterte med ulike løsninger inntil det fungerte. Resultatet av slike problemer ble ofte nye maler, som kunne tas i bruk på de neste store byggeprosjektene.

Slik ble altså malene et sentralt verktøy for de gotiske byggmestrene, fordi de gjorde det mulig å overføre taus kunnskap mellom arbeiderne på tvers av språk og kultur. Denne prosessen, der taus kunnskap gjøres eksplisitt, kaller Nonaka og Takeuchi (1998) for *eksternalisering*. De fremhever dette som en svært viktig prosess for å kunne kommunisere kunnskap ut over det lokale arbeidsfellesskapets grenser. Innenfor dagens IT-dominerte verden, kan vi kjenne igjen denne prosessen blant annet i utviklingen av lokale, nasjonale og internasjonale *standarder*. Eksempler på slike standarder i dag kan være ISO-standarder for kvalitetssikring på arbeidsplassen, ITIL-

standarder for telecom og IT, og de mange standardene fra IETF som har lagt mye av grunnlaget for oppblomstringen av internett. For OS-bevegelsen har utviklingen av gode standarder vært enormt viktig. Uten disse verktøyene for koordinering og konsolidering av kunnskap, ville vi neppe sett den samme effekten av alle OS-utviklernes individuelle bidrag. I tillegg er det en utstrakt praksis innenfor OS-utvikling at man går svært direkte og konkret til verks når man kommuniserer problemer og løsninger. Generelle problemer blir som gjerne møtt med konkrete eksempler på relaterte løsninger. Linus Torvalds blir ofte sitert på sitt enkle mantra: «Show me the Source», som oppsummerer mye av innfallsvinkelen OS-bevegelsen har overfor systemutvikling, problemløsning og kommunikasjon.

Jeg vil nå gå videre inn på diskusjonen omkring taus kunnskap; hvordan den oppstår og spres i en organisasjon. En svært viktig faktor i den sammenheng er hvordan kunnskap blir *kommunisert,* på hvilken måte og gjennom hvilke kanaler.

2.3.4 Uformell kommunikasjon; kaffepraten

Et viktig aspekt med måten vi kommuniserer på, er hvor formelle eller uformelle rammer man opererer med. Formell kommunikasjon blir som regel benyttet ved rutinemessige oppgaver og gjerne i form av planlagte møter, med fast agenda og i mange tilfeller faste samtaleregler. Uformell kommunikasjon blir atskillig mer fremtredende ved uforutsette begivenheter, hastesaker eller kompliserte problemer, alt dette hendelser som opptrer forholdsvis ofte i svært mange organisasjoner. Uformell kommunikasjon oppstår gjerne ved kaffemaskina, i korridoren eller under felles utførelse av arbeid, og er mye mer dynamisk og interaktiv enn formell kommunikasjon. Dette gir rom for spontane fortellinger som kan fungere som bærere av taus kunnskap.

Relasjonene mellom de som kommuniserer vil i stor grad påvirke graden av formalitet. Her vil lederstil, hierarki og sosiale eller kulturelle faktorer være med på å sette premissene for kommunikasjonen. Hvor hyppig man kommuniserer vil også ha betydning. Hvis man ser hverandre gjentatte ganger daglig, vil det for de fleste være unaturlig å opprettholde en høy grad av formalitet. Videre vil arenaen for kommunikasjon spille inn på grad av formalitet. I et møtelokale vil man typisk oppleve en mye mer formell omgangsform enn ved et treningssenter. Hva slags medium man kommuniserer gjennom vil også være en viktig faktor. Et brev man får i posten vil typisk ha en mye strengere formell karakter enn for eksempel en samtale over telefon (Kraut, Fish, Root & Chalfonte 1990).

Med fremveksten av nye IT-verktøy har vi fått flere spennende mellomkategorier av medier med ulik grad av formalitet, inkludert elektronisk post, forum og pratekanaler. Innenfor disse nye mediene er det OS-utviklerne trives best; her er det brorparten av kommunikasjonen foregår, mens de mer tradisjonelle mediene som regel blir ignorert. Jeg vil etter hvert diskutere de nye medienes reelle og potensielle rolle i organisasjoners kunnskapsarbeid, men for å tilnærme seg dette emnet er det hensiktsmessig å etablere hvilken *kontekst* ulike medier kan operere i, fra små lokale grupper til store globale nettverk.

2.3.5 Praksisfelleskap og praksisnettverk

Uformell kommunikasjon danner grunnlag for etablering av fellesskapsfølelse og tilhørighet overfor den gruppen mennesker man til daglig arbeider sammen med. Desto tettere man arbeider sammen og jo flere arbeidsoppgaver som kan løses i fellesskap, desto større er sjansen for at det etableres et lokalt praksisfellesskap (Engelsk: Community of Practice - CoP). Et praksisfellesskap kan betegnes som en hyppig besøkt arena for faglig kommunikasjon, koordinering og sosial kontakt. Disse dannes typisk gjennom lokal samhandling over tid, der man utvikler felles problemforståelse, praksis og gruppeidentitet. Det finnes per i dag en rekke studier som benytter praksisfelleskap som analyseenhet, og som viser hvordan fungerende praksisfelleskap kan gi anselig gevinster i form av læring, kunnskapsspredning, innovasjon, produktivitet og jobbtilfredshet (se bl.a. Fontaine & Millen 2003, Frost & Shoen 2004, de Laat & Broer 2004, Kolbotn 2004).

I de nevnte studiene av praksisfelleskap er det lagt anselig vekt på hvordan intern kommunikasjon foregår, og i hvilke *medier* den opptrer. De mest brukte mediene for kommunikasjon blir gjerne rangert etter hvor ”rike” de er, og rikdom blir her brukt som indikasjon på hvor effektive de er til å endre samtalepartnerens forståelse. Muntlig samtale blir typisk nevnt som den rikeste, og synkron media som generelt sett rikere enn asynkron (Shenkel 2004, Kraut, Fish, Root & Chalfonte 1990). De rike mediene er i teorien velegnet til kompleks kommunikasjon, for eksempel ved interne forhandlinger (Wijayanayake & Higa 1999) eller ved oppstarten av et nytt praksisfelleskap, der det fortsatt eksisterer forholdsvis mye usikkerhet rundt arbeidsform og samhandling. Uten tilgang til rik media, helst i form av fysiske møter, vil det være vanskeligere å få i gang en læringsprosess i møte med nye og usikre situasjoner (Shenkel 2004). Undersøkelser har også vist klare sammenhenger mellom fysisk nærhet og samarbeidsvillighet. De fleste utvikler størst faglig og sosial respekt for de personene de oftest «dumper borti» i sitt daglige virke, og samarbeider bedre med disse enn med andre, mer geografisk perifere, medarbeidere (Kraut, Fish, Root & Chalfonte 1990).

I mange tilfeller er det imidlertid ikke praktisk mulig eller rasjonelt å organisere et praksisfelleskap med stabil tilgang på rike medier. Geografiske avstander, for eksempel, kan gjøre det vanskelig å arrangere fysiske møter på jevnlig basis. For å overkomme dette hinderet har mange organisasjoner forsøkt å implementere ulike typer gruppevare, det vil si programvare som er designet for å støtte opp om virtuelt samarbeid. Resultatene har dessverre ofte uteblitt, på tross av store investeringer i slike systemer. Jonathan Grudin (1989) hevder de fleste gruppevareprosjekter mislykkes på grunn av tre forhold. For det første er det ofte et gap mellom hvem som legger mest arbeid inn i systemet og hvem som høster flest fordeler. Ledelsen vil gjerne ha mest mulig informasjon om det daglige arbeidet, for dermed å kunne generere aggregerte data om situasjonen for hele arbeidsgruppa. Den enkelte bruker av systemet vil derimot oppleve innleggingen av den samme data som en ekstra byrde i en allerede travel hverdag. Dette vil over tid skape generell misnøye og unøyaktig innrapportering til systemet. For det andre er det svært sjeldent at ledelsen har en godt utviklet intuisjon for hva som er god og dårlig gruppevare. Dette henger sammen med at verktøyet skal kunne benyttes av mange brukere med ulike behov, og dermed vanskelig kan forstås fullt ut av en enkelt leder. I tillegg er det mange utfordringer knyttet til å evaluere slike verktøy, siden brukerne gjerne vil oppfatte verktøyet ulikt ut i fra deres arbeidssituasjon. Dette øker vanskelighetene med å skille klinten fra hveten i mylderet av applikasjoner som tilbys ledelsen (Grudin 1989).

Enkelte typer gruppevare forsøker bevisst å skape virtuelle arenaer som etterligner fysiske arenaer, ved hjelp av audiovisuelle hjelpemidler. Hensikten er da gjerne å etablere muligheter for spontan uformell kontakt på tvers av geografiske avstander. Kraut et.al. Beskriver et system der ble det forsøkt å sy sammen pauserommene til to geografisk atskilte avdelinger i organisasjonen. Dette ble gjort ved hjelp av en stor skjerm midt i pauserommene, som til enhver tid viste hva som foregikk i den andre avdelingens pauserom. Dette fungerte bare til en viss grad, ettersom den uformelle kontakten ikke så lett oppsto på tvers av videoskjermen. Den samme forpliktelsen til å inngå en samtale som man opplever i det fysiske rom, fant ikke sted i like stor grad (Kraut, Fish, Root & Chalfonte 1990).

Det finnes imidlertid en annen innfallsvinkel til problemet med geografisk spredning av ansatte. I stedet for å forsøke å skape nye altomfattende verktøy for samhandling, eller å kopiere den uformelle kommunikasjonen i et praksisfelleskap, kan man velge å sette andre mål og visjoner for hvordan samhandlingen skal gå for seg. For de fleste OS-prosjekter vil man måtte ta stilling til hvordan man samhandler over store avstander, ettersom rekrutteringen til disse i hovedsak foregår over internett og i mindre grad er knyttet til geografisk eller nasjonal tilhørighet. Her er det ikke

vanlig å bruke rike medier i det hele tatt, men i stedet satser man på enkle tekstbaserte verktøy, som e-postlister eller pratekanaler. I den kommersielle sfære har man også begynt å se fordelene ved å etablere slike enkle systemer. Ved Cap Gemini, et multinasjonalt konsulentfirma, ble det bevisst lagt til rette for opprettelsen av praksisfellesskap over internett, i første rekke ved hjelp av felles e-postlister for alle ansatte med noenlunde samme arbeidsområder. Dette ble et populært tiltak blant de ansatte, og ved nærmere utspørring kunne man observere en styrket tro på egen kunnskap og faglig utvikling blant de som deltok aktivt i det virtuelle fellesskapet som ble dannet (Teigland & Wasko 2004).

På tross av mangelen på fysisk kontakt og rike medier, er begrepet «fellesskap» (engelsk: community) godt etablert i OS-kretser, og man finner ofte en sterk gruppeidentitet og følelse av tilhørighet blant de som er aktive i et prosjekt. Det virtuelle nettverket ser ut til å ha tatt over praksisfellesskapets rolle. Vi kan med andre ord løfte blikket fra de lokale *praksisfelleskap*, over til de bredere *praksisnettverk* - eller mer presist; *elektroniske praksisnettverk* (engelsk: Electronic Networks of Practice – ENoP). Logikken bak opprettelsen av et EnoP kan være en erkjennelse av følgende: Det man mister av rikdom i kommunikasjonen, i forhold til et lokal praksisfelleskap, vinner man tilbake gjennom den bredden og det nedslagsfeltet man oppnår. Ved å åpne for dannelsen av et stort nettverk, med mange flere deltakere enn en lokal arena kan bære, øker muligheten for alle deltakerne til å komme i kontakt med nye personer og ny kunnskap (Teigland & Wasko 2004).

Hvordan kommuniserer man i så fall i et EnoP, hva er det viktigste mediet? Janaka Wijayanayake og Kunihiko Higa har foretatt en større empirisk undersøkelse av medievalg hos folk som arbeider sammen på tross av geografisk spredning. Deres resultater viser at de mest brukte mediene er telefon og e-post. Dette gjelder også der avanserte og rikere medier har vært tilgjengelig. Et litt overraskende funn er at e-post, selv om det ikke er et spesielt rikt medium, hyppig blir benyttet til kompleks kommunikasjon som diskusjoner og forhandlinger. I følge teorier om rikdom i medier er ikke e-post særlig egnet til denne typen kommunikasjon, likevel viser altså undersøkelsene at det blir flittig brukt, uten at ledelsen har lagt noe press i den retning (Wijayanayake og Higa 1999). Det er også en tendens til at deltakerne i et ENoP tilpasser bruken av mediet over tid, slik at begrensninger som er iboende i mediet ikke lenger blir like relevante. For eksempel så ser man at deltakerne i en e-postliste vil etter hvert finne fram til en del felles kjøreregler (både formelle og uformelle) som skal hindre støy og misforståelser. Mange finner også at bruken av et asynkront medium gir dem større rom for refleksjon og formulering, enn det som er tilfelle ved rikere medier

som for eksempel fysiske møter (Kock 1998). Via e-post utveksler med andre ord alt fra korte beskjeder, lengre fortellinger og eksempler fra egen arbeidshverdag, alt etter som hvem man adresserer. Alle disse innleggene kan enkelt logges og gjøres tilgjengelig for deltakerne i nettverket, og dermed utgjøre en felles kunnskapsbase for alle involverte. Ute blant bedrifter og organisasjoner blir dessverre ofte e-post ignorert av ledelsen i utarbeidelsen av planer og strategier for kunnskapsarbeid (Bontis, Fearon & Histon 2003). Her virker OS-miljøene mer bevisste på potensialet i e-postmediet, med tanke på den utstrakte bruken av e-postlister til både kommunikasjon og arkivformål. .

2.3.6 Kunnskapsledelse

Det er ingen nødvendighet i at et praksisfellesskap skal gi alle de positive effektene for læring og kunnskapsoverføring som man ønsker seg. Frost & Shoen (2004) har studert mange ulike praksisfelleskap ved Siemens AG og identifisert fem generelle forhold som må legges til rette for at et praksisfelleskap skal fungere:

1. *Organisering og støtte for aktiviteter i fellesskap*. Konferanser, uformelle møteplasser og felles IT-verktøy er eksempler på tiltak som danner grunnlaget for et praksisfelleskap.

2. *Opprette relasjoner mellom mennesker og kunnskapen de besitter.* Ved å legge til rette for at de som arbeider innenfor samme domene har samlingspunkter der både de menneskelige og teknologiske ressursene er samlet, støtter man opp om praksisfelleskapet.

3. *Finne et felles fokus*. For å få et praksisfelleskap til å løfte i samme retning, kreves det at man etablerer et felles fokus. Dette kan skje via inspirerende og målrettede ledere, og via gode prosesser for å kontinuerlig revurdere og «spisse» fokuset.

4. *Virke med fellesskapets ytre og indre miljø.* Et dynamisk praksisfelleskap bør være fleksibelt i forhold til endringer i det indre miljøet, inkludert en eventuell overordnet ledelse, og nye potensielle deltakere. Det bør også ha et bevisst forhold til hva som skjer i verden rundt seg, både i forhold faglige og organisasjonelle spørsmål.

5. *Leve ut fellesskapets verdier og normer.* Et praksisfelleskap er avhengig av at dets egne normer og verdier blir fulgt opp av alle som deltar i det. Innholdet i disse vil variere fra sted til sted, men det finnes også noen som er universelle; *tillit* og *åpenhet*. Her har ledelsen

(eller dominerende deltakere) et spesielt ansvar i å gå foran med et godt eksempel og bidra til å sette en best mulig standard.

Disse kriteriene for et fungerende praksisfelleskap lar seg enkelt overføre til elektroniske praksisnettverk. Forskjellen ligger først og fremst i at de sosiale og mellommenneskelige aspektene ved samhandlingen blir noe vanskeligere å fremme, siden det ofte vil være upraktisk og kostbart å samle deltakerne i nettverket rent fysisk. Det vil imidlertid være enklere å få til en mer samkjørt bruk av teknologiske verktøy, ettersom disse vil være de primære formene for kommunikasjon og samhandling i et elektronisk praksisnettverk (i motsetning til et lokalt praksisfelleskap, der fysiske møter og uformell spontan kontakt spiller en større rolle). Hvis vi igjen tar e-post som en mulig kandidat til primær kommunikasjonskanal i et EnoP, ser vi raskt viktigheten av at man har et gjennomtenkt forhold til hvordan dette mediet benyttes. I en undersøkelse av e-postbruk i telesektoren, poengterer Higa med flere (Higa, Sheng, Shin & Figueredo 2000) hvor viktig det er med støtte og veiledning fra ledelsen, for at e-post skal bli brukt til sitt fulle potensiale. De hevder at man med en god kunnskapsstrategi og ledelse omkring bruken av e-post, kan oppnå mange av de samme positive effektene som vanligvis forbindes med mer rike medier.

Etter å ha blitt bedre kjent med OS-bevegelsen, dens historie, arbeidsform og verktøy, kan vi trygt si at denne tradisjonen kan beskrives ved hjelp av konseptet elektroniske praksisnettverk. Vi har videre sett at man i mangelen på rike medier innenfor et EnoP, kan adoptere mer «fattige» medier, i form av blant annet e-post, og benytte disse som basis for kunnskapsarbeid i organisasjonen, uten at man mister evnen til å ha en effektiv og kunnskapsrik kommunikasjon. For at dette skal kunne realiseres, må noen nøkkelfaktorer være på plass, som diskutert over. Innenfor OS-miljøene blir derfor grunnleggeren for et prosjekt særdeles viktig. Jo flere av de fem kriteriene som grunnleggeren makter å legge til rette ved starten av et prosjekt, jo bedre er sjansene for at prosjektet kan vokse seg større, og at andre deltakere over tid bidrar til å støtte oppunder den videre utviklingen. Her har OS-bevegelsen en styrke ved at de har tilgang til en rekke tradisjonsrike verktøy og metoder for dannelse og drift av elektroniske praksisnettverk, som beskrevet i kapittel 2.1.6.

Siden OS-utvikling i all hovedsak er basert på frivillig arbeid, og kan beskrives som en form for gaveøkonomi (kapittel 2.1.5) oppstår det en ekstra lederutfordring: Hvor mye kan man forme og strukturere kommunikasjon og samhandling i et OS-prosjekt når alle deltar frivillig? Holck & Jørgensen (2005) tar opp dette temaet i sin undersøkelse av OS-prosjektene FreeBSD og Mozilla.

Forfatterne formulerer denne lederutfordringen ved uttrykket «å balansere mellom anarki og kontroll». På den ene siden ønsker man en mest mulig åpen og fri organisasjon, tilsynelatende anarki, for å sikre en stor brukerbase, god tilgang på nye utviklere og spredning av kunnskap. På den andre siden ønsker man å sikre kvaliteten både i prosess og produkt, gjennom å innføre ulike former for kontroll. Forfatterne hevder at deres to case har funnet en god balansegang mellom disse tildels motstridende hensyn, gjennom å utnytte de teknologiske verktøyene som OS-bevegelsen har til rådighet. Dette inkluderer programvare for versjonskontroll (CVS / Subversion), websider med kontinuerlig oppdatert statusinformasjon om prosjektet, e-postlister og nyhetsgrupper, feilrapporteringssystemer og verktøy for prosjekthåndtering (ala TRAC). Denne balansegangen er en pågående prosess innad i organisasjonen, som gjerne pendler litt fra side til side ut i fra hva som er det mest åpenbare behovet til en hver tid (Holck & Jørgensen 2005). For en effektiv kunnskapsledelse i et OS-prosjekt er det med andre ord viktig å være seg denne balansegangen bevisst og bidra til at pendelen svinger i en retning som er i samsvar med organisasjonens behov som helhet.

3 Metode og verktøy

I dette kapittelet vil jeg ta for meg hva slags metodisk utgangspunkt denne avhandlingen bygger på, beskrive forskningsprosessen og fortelle om de ulike hjelpeverktøyene jeg har utarbeidet.

3.1 Å måle eller fortolke

Forskning på informasjonssystemer har lenge vært delt mellom en europeisk og en amerikansk tradisjon. Dette henger sammen med forskjeller i vitenskapsteoretisk fokus på hver sin side av Atlanterhavet. I USA har det lenge vært en sterk positivistisk tradisjon, basert på tankegods fra i første rekke August Comte. Babbie & Mouton (2001) oppsummerer Comtes filosofi slik:

> *"Comtes idealsamfunn er fundert på vitenskapelige prinsipper. For å oppnå dette idealet må humanistiske vitenskaper få like god kontroll over sitt domene som det naturvitenskapen har klart. Det er derfor logisk at den beste strategi for humanistiske vitenskaper er å følge samme metodologi som naturvitenskapen. Dette innebærer at målet for begge vil være å etablere universelt gyldige, kausale lover om menneskelig aktivitet." (min oversettelse)*

Med dette som filosofisk bakteppe ble den amerikanske IS-tradisjonen raskt dominert av studier der man forsøkte å etablere generelle lover basert på kvantitative data, for eksempel i form av store spørreundersøkelser eller annet statistisk materiale. Dette skilte seg klart fra den europeiske IS-tradisjonen, som i større grad var influert av en annen vitenskapsfilosofi med hovedsete i Tyskland. Max Weber demonstrerte med sin avhandling om protestantismens etikk en annen måte å arbeide vitenskapelig på, som ikke forholdt seg til de rigide krav som positivismen stilte opp (Pather & Remenyi 2004). Fremfor å søke universelle lover, var fokuset i stedet satt på å utforske og fortolke virkeligheten rundt seg. Dybden i beskrivelsene, sammen med den logiske oppbygningen av argumentasjonen, var grunnlaget for de vitenskapelige funn. Funnene ble heller ikke presentert som absolutte størrelser, men som informerte beskrivelser av mulige sammenhenger. Dette satte en standard som hadde, og har, stor innflytelse på europeisk forskning generelt, inkludert IS-forskning.

I USA vokste det imidlertid fram en lignende alternativ tradisjon, representert ved Chicago School of Sociology, som har inspirert mange forskere i ettertid til å bevege seg utenfor positivismens rammer (Pather & Remenyi 2004). På tross av denne skolens arbeid finner vi fortsatt i dag en markant forskjell mellom IS-forskning i USA og i Europa. Det er likevel en tendens til at gapet mellom tradisjonene blir mindre og at man i større grad klarer å utnytte hverandres forskningsresultater (Evaristo & Karahanna 1997). Dette håper jeg vil fortsette i tiden som kommer, ettersom jeg ikke kan se noen god grunn til at IS-forskere skal la eventuelle ideologiske baktepper hindre en potensielt fruktbar dialog.

3.1.1 Begrensninger ved tradisjonelle naturvitenskapelige metoder

Forskning på informasjonssystemer (IS) har tradisjonelt vært dominert av empirisk-analytiske metoder (Galliers & Land 1987). Innen forskning på informasjonssystemer ser vi tendenser til økt fokus på kvalitative metoder, spesielt i de domener der mennesker møter teknologi, men det er fortsatt et marginalt fenomen; under 5% av artiklene i ledende forskningspublikasjoner benytter slike metoder i følge Orlikowski & Baroudi (1991). Argumentasjonen for å holde seg innenfor den kvantitative metodikken, har i første rekke handlet om et krav om nøyaktighet og målbarhet. Fokuset har vært på analyse av statistisk materiale, i den hensikt å avdekke sammenhenger mellom et begrenset antall strengt definerte avhengige og uavhengige variabler. Datainnsamling går i slike tilfeller som regel for seg gjennom bruk av representative utvalg eller via laboratorieforsøk der man forsøker å ha kontroll over alle relevante faktorer i problemstillingen. Dette er velprøvde og vel anerkjente metoder for innhenting av data, men de er begge heftet med noen innebygde

begrensninger, som jeg har valgt å belyse gjennom å stille følgende to spørsmål:

1. Hvor mye av problemområdet lar seg kvantifisere?

Kravet om målbarhet fører til en nødvendig innskrenking av hva som kan tas med i undersøkelsen. Mange faktorer lar seg vanskelig kvantifisere, spesielt når det gjelder måling av menneskelig aktivitet. Ved å velge bort de fleste vanskelig målbare faktorer, kan man risikere å få systematiske skjevheter. Slik oppstår en fare for at man over tid etablerer vitenskapelige «sannheter», som overser en eller flere signifikante forhold. Myers (1994) har undersøkt implementeringen av IT-systemer hos myndighetene i New Zealand. Han finner at tidligere forskning har vært altfor snever og mekanistisk i metodebruken, og hevder videre at implementering av IT-systemer bare kan bli forstått som en del av en større sosial og organisatorisk kontekst. Nonaka & Takeuchi (1998) bekrefter, i sin studie av kunnskapsoverføring, at de vanskelig målbare variablene kan være essensielle for å forstå en organisasjon. Den viktige tause kunnskap vil i følge dem ikke kunne oppdages ved bruk av kvantitative metoder, ettersom disse krever at kunnskapen blir formulert eksplisitt og entydig. Når Nonaka & Takeuchi (1998) videre hevder at taus kunnskap som regel utgjør hovedtyngden av en organisasjons samlede kompetanse, utgjør det et klart argument for å ta alternative kvalitative metoder på alvor.

2. Kan et laboratorium gjenspeile dagliglivet på en tilfredsstillende måte?

For å kunne ha kontroll over faktorene man ønsker belyst ved bruk av laboratoriemetoder, må man etablere en situasjon der man eliminerer de faktorer man ikke tror er viktige. Selv om man har gode argumenter for hvilke faktorer man eliminerer, vil det alltid være en fare for at man eliminerer en faktor som ville hatt en subtil effekt ute i felten. Forskjeller mellom laboratoriet og den reelle situasjonen man imiterer, kan i verste fall gi resultater som reproduserer den samme skjevheten man introduserte i det man foretok utvelgelsen av faktorer til undersøkelse. Ved å fjerne konteksten til det man måler, gir man uttrykk for at man selv som forsker vet nøyaktig hvor grensen mellom konteksten og fenomen ligger. Dette kan være et problematisk og kanskje litt naivt utgangspunkt.

3.1.2 Fortolkning som metode

Hvis man så godtar innvendingene mot den tradisjonelle naturvitenskapelige metode, gjenstår det å etablere et alternativ. Hvis man ikke kan *bevise* eller *falsifisere,* hva er da hensikten med drive forskning? Et mulig svar på dette kan man finne ved å skifte fokus fra å spekulere, til å beskrive. Den hypotetisk-deduktive metode, der man først formulerer et utsagn for så å teste om det holder

mål i «virkeligheten», kan beskrives som en *spekulerende* metode. Som en motsats til dette har det blitt fremmet metoder som har *beskrivelsen* som mål i seg selv (Galliers & Land 1987). Utforskingen av ulike fenomen er i dette perspektivet ikke en øvelse i å teste ferdigformulerte forutsigelser, men en øvelse i å forstå og tolke det man observerer. Et prinsipp innen fortolkende forskning er at kunnskap om virkeligheten baserer seg på sosiale konstruksjoner som språk, bevissthet, delte meninger, dokumenter, verktøy og andre artefakter. Man prøver å forstå fenomener gjennom den betydningen mennesker tilegner dem.

En sentral utfordring ved å drive fortolkende forskning, er å etablere gode kriterier for kvalitet i forskningen. Hvis virkelighet og sannhet i stor grad blir redusert til subjektive størrelser, hvordan kan da forskeren klare å etablere en vitenskapelig metode som utgjør noe mer enn en subjektiv oppfatning, eller ren synsing om man vil? I motsetning til ved tradisjonelle kvantitative metoder, har man her i liten grad hjelp fra matematikken eller statistikken, men må i stedet gå andre veier for å sikre at forskningsprosessen holder en høy vitenskapelig standard.

Kleyn & Myers (1999) har studert flere forskningsprosjekter innenfor en fortolkende tradisjon, og utarbeidet syv metodiske prinsipper for denne typen forskning. Det første prinsippet er knyttet til en vitenskapsfilosofisk tenkemåte, og kalles (1) «det fundamentale prinsippet om den hermenautiske sirkel». Den hermenautiske sirkel er en konseptualisering av hvordan forståelse og innsikt oppnås gjennom å stadig skifte fokus fra del til helhet, fra dybde til bredde. Detaljene ved et fenomen får økt mening ved å se dem i forhold til en større helhet. Helheten trer deretter tydeligere fram ved at nye detaljer blir undersøkt. Prinsippet er fundamentalt, i den forstand at det gjelder som en forutsetning og et bakteppe for alle de andre prinsippene for fortolkende forskning. Jeg har i tråd med prinsippet om den hermenautiske sirkel, prøvd å benytte verktøy og metoder som lar meg gå nok i dybden til å fange opp interessante detaljer. Ved hjelp av et annet sett av både praktiske verktøy og teoretiske byggverk, har jeg videre forsøkt å etablere en forståelse av helhetsbildet, som kan justeres fortløpende etter hvert som nye, oppklarende detaljer blir funnet.

For å oppnå en god oversikt over helhetsbildet knyttet til fenomenet man ønsker å undersøke, er det viktig at man har et bevisst forhold til hva slags kontekst det opererer innenfor. Dette prinsippet kaller Klein & Myers for (2) *kontekstualisering,* og handler om å sette ting i en sosial og historisk sammenheng, slik at man kan tilegne situasjonene som blir beskrevet mening ut ifra konteksten. Forskeren må se på mennesker som aktive formere av historien, ikke bare som produkter. En metodisk strategi som fanger opp mye av essensen i dette prinsippet, er case-studiet.

3.2 Case-studier og innsamling av empiri

En case-studie kan defineres som en forskningsstrategi, der man undersøker et nåværende fenomen *i sin reelle kontekst*. Dette er et viktig kjennetegn som står i skarp motsetning til eksperimentelle metoder, der man forsøker å skille fenomenet ut fra den konteksten det opererer i. Grensene mellom hva som er kontekst og hva som er fenomenet selv, vil ofte også være flytende og gjenstand for kontinuerlig revisjon i løpet av forskningsprosessen (Yin 1981). Case-studier kan være av både kvantitativ og kvalitativ art, men kvalitative metoder har tradisjonelt vært i overvekt her, sannsynligvis på grunn av case-studiet sitt utspring fra en fortolkende tradisjon.

Jeg har i denne avhandlingen valgt å fokusere på to konkrete case, VideoLAN og Xiph, som begge er aktive i dag, og som har en relativt kort og oversiktlig historie (mindre enn ti år). Disse casene vil bli mer utførlig presentert i kapittel 4. Mine kilder for empiri har vært e-postarkiver fra OS-prosjektenes hjemmesider, log fra IRC-kanalene jeg har deltatt i, personlig intervju over IRC, offentlige websider og diskusjonsforum, samt prosjektenes åpne kildekode. Dette utgjør en svimlende mengde materiale, som jeg ville ha måttet brukt år på å gå igjennom i detalj. Jeg har derfor valgt flere ulike innfallsvinkler til behandlingen av empirien, ut i fra hvilken kilde den stammer fra.

3.2.1 Bevegelser i kodebasen

For å få noenlunde oversikt over aktiviteten innad i prosjektene, med hensyn til selve skrivingen av kildekode, utviklet jeg et enkelt nettbasert verktøy for å systematisere informasjon om bevegelser i kodebasen. Kilden til denne informasjonen var prosjektenes egne nettbaserte kodevisninger fra Subversion. Her kunne man få skrevet ut en liste over alle endringer som var gjort i basen, med dato, brukernavn og kommentar. Denne listen var imidlertid så massiv at den var lite verdt som analytisk verktøy alene. Ved hjelp av et nettbasert skript fikk jeg overført informasjonen i disse listene over til min egen database, slik at jeg kunne behandle informasjonen mer strukturert. Appendiks A har flere detaljer om hvordan dette ble gjort.

Jeg må presisere at grafene og tallene som ble laget ut i fra disse dataene kun var laget for å få en grov oversikt over utviklingstempo og deltakelse i det pågående arbeidet med kodebasen. Siden utviklerne kan ha ulik praksis i forhold til hvor mye og hvor ofte de legger inn endringer, vil det kunne ligge skjevheter innebygd i disse tallene. Ergo må dette brukes varsomt, som en av flere innfallsvinkler til en diskusjon, heller enn som endelig «bevis» eller konkluderende materiale.

3.2.2 Katalogisering av e-postinnlegg

Når det gjelder e-postlistene og loggene fra samtalene på IRC, hadde jeg et anselig stykke katalogisering og analyseringsarbeid foran meg. For å håndtere dette lot jeg meg inspirere av «grounded theory» (GT), en integrert metode og teknikk som har sine røtter innen sosiologien. Strauss og Glaser utviklet «grounded theory» som en reaksjon på andre samtidige sosiologers store og «luftige» teoribygging. De ønsket seg tilbake til en sosiologi som i større grad var fundamentert (grounded) direkte i empirien, så man fikk et mest mulig stabilt grunnlag for eventuelle teorier. Fremgangsmåten innenfor GT kan beskrives i tre steg, der den første er (1) åpen koding. Her forsøker forskeren å beskrive empirien ved å sette mange ulike merkelapper på dataene. I denne fasen skal forskeren forsøke å være minst mulig styrt av teori eller forventning, men heller fokusere på å holde seg nært empirien. Når man så sitter med en mengde ulike merkelapper, begynner fase (2), aksial koding. Nå tar forskeren for seg haugen med merkelapper, og forsøker å aggregere dette opp til noen mer generelle kategorier, samt oppretter funksjonelle relasjoner mellom disse. Siste fase, (3) selektiv koding, blir en finspikking av de eksisterende kategoriene, der man forsøker å trekke ut hovedkategoriene og gjennom disse danne grunnlaget for ny teori (Goede & de Villiers 2003).

Det er imidlertid enkelte alvorlige problemer heftet med denne metoden. For det første er den svært omfattende, og krever store mengder koding. I tillegg oppdaget jeg tidlig at det objektive idealet knyttet til åpen koding var svært vanskelig å leve opp til. Med den teoretiske diskusjonen i bakhodet var det nesten umulig å unngå at merkelappene ble relatert til de temaene jeg ønsket å belyse, selv om jeg bevisst forsøkte å ikke gjøre dette. Jeg vurderte en stund å droppe metoden helt på grunn av dette, siden jeg gjennom min ubevisste praksis så ut til å bryte med den fundamentale forutsetningen for teorien. Det jeg derimot innså, var at jeg gjennom å utføre trinn en og to i metoden, kunne bygge meg opp en systematisk oversikt over empirien – noe som hadde en definitiv verdi i seg selv. Ved å kategorisere data i et hierarkisk mønster på linje med aksial koding, ville jeg få en oversikt over hvordan dataene så ut, samt på en enkel måte kunne gå i dybden og lete etter interessante detaljer innenfor en hel rekke ulike tema. Ergo gikk jeg fra å vurdere grounded theory som en mulig hovedmetode, til å se på den som et av flere verktøy innenfor en mer tradisjonell kvalitativ og hermeneutisk analyse. Dermed ble faktisk min subjektivitet i den åpne kodingen heller en fordel enn et problem, ettersom jeg nå kunne bruke kategoriene som veiskilt til interessante empiriske detaljer.

Jeg fant ut at e-postlistene var godt egnet til å bli systematisert ut i fra den GT-inspirerte

framgangsmåten, siden disse var delt opp i klart adskilte enheter som kunne behandles separat. Loggene fra IRC-kanalene var ikke like godt egnet til dette følte jeg, så disse ble behandlet på en annen måte, som jeg kommer tilbake til i neste delkapittel. For e-postlistene opprettet jeg altså en database med et nettbasert grensesnitt, der jeg kunne registrere enkeltinnlegg samt opprette kategorier (merkelapper) som innleggene ble knyttet til. Etter hvert fikk jeg en nokså stor mengde med kategorier, og begynte å se sammenhenger mellom enkelte av dem. Systemet ble da bygget ut til å kunne behandle hierarkier av kategorier, samtidig som jeg foretok fortløpende revisjoner av de eksisterende kategoriene. Resultatet ble en database over e-postinnlegg fra mine to case, katalogisert ut i fra innholdet i hver enkelt. Dette ga meg to nyttige hjelpemidler til analyse: (1) en oversikt over den generelle bruken av listene, og (2) en enkel metode for å gå i dybden og hente ut detaljer om spesifikke tema. Se appendix B for ytterligere detaljer om dette verktøyet.

3.2.3 Deltakelse i pratekanaler

Klein & Myers tredje prinsipp for fortolkende forskning omhandler interaksjon mellom forskeren og hans studieobjekter. Forskeren må plassere seg selv og sine forskningsobjekter inn i den sosiale og historiske konteksten. Data sees på som et resultat av sosial interaksjon mellom forsker og forskningsobjekt, og følgelig er forskeren en del av konteksten som det forskes i. Fortolkende forskere må erkjenne at forskningsobjektene, såvel som forskeren, fortolker og analyserer og dermed bevisst eller ubevisst farger historien. Dette prinsippet lå til grunn for en av innfallsvinklene jeg søkte overfor de prosjektene jeg studerte, i den forstand at jeg ikke hadde til hensikt å prøve å skjule min egen rolle i forskningsprosessen, tvert i mot. Jeg ville bruke meg selv som et eksempel på hvordan det er å komme utenfra og forsøke å komme inn i disse miljøene. Gjennom dette håpet jeg å få en ekstra innsikt i de sosiale og kommunikative prosessene i casene.

I starten gikk jeg gjennom offisielle kanaler, ved å sende e-post til alle offisielle adresser med informasjon om mitt prosjekt, men dette ga meg så godt som ingen respons. Jeg forsøkte deretter å ta kontakt via e-postlistene, uten at dette ga noe særlig mer resultater. For mitt ene case, Xiph, så jeg ikke på dette som et stort problem, siden jeg hadde store mengder empiri om arbeidsprosessen fra deres utstrakte bruk av e-postlister. For mitt andre case, VideoLAN, var ikke det samme tilfelle. Her fant jeg lite stoff på e-postlistene, og det virket som utviklerne ikke brukte denne kommunikasjonskanalen i særlig grad. I motsetning til Xiph hadde imidlertid VideoLAN en særdeles aktiv og høyt trafikkert IRC-kanal, der både brukere og utviklere holdt til. Jeg begynte derfor å bli en fast deltaker i kanalen, først som en tilbaketrukket observatør, men etter hvert også

som en deltaker i de pågående samtaler og diskusjoner. Min egen bakgrunn fra multimedia og streaming[12], ble en viktig innfallsport for å komme nærmere dette miljøet. Ved å ha noen egne historier og erfaring å dele, kunne jeg lettere komme i dialog med de andre utviklerne. Jeg deltok ikke i selve utviklingen, men i de generelle faglige diskusjonene og støtten til nye brukere.

Jeg introduserte meg selv tidlig som forsker, både via offisielle kanaler og direkte på IRC. Jeg tror likevel ikke forskerhatten har vært spesielt fremtredende i mitt virke på IRC, ettersom tempoet er høyt i denne kanalen, med svært mange deltakere, og man forsvinner fort i mengden. Det var heller ikke mulig for meg å observere noen endret holdning eller stil hos gjengangerne i kanalen fra tiden før jeg introduserte meg til tiden etter.

Mesteparten av denne avhandlingen ble skrevet fra min arbeidsstasjon hjemme, som har to skjermer tilknyttet. Jeg gjorde det derfor til en vane å alltid ha IRC gående på den ene skjermen, uansett om jeg satt og jobber med avhandlingen eller gjorde andre ting med arbeidsstasjonen. Dette følte jeg var en inspirerende måte å arbeide på, ettersom det ga meg en direkte link til det jeg forsøkte å beskrive. Samtidig ga det rom for nødvendige kognitive pauser, ved at jeg inni mellom kunne legge til side det jeg holdt på med for å delta i en eller annen urelatert diskusjon på IRC.

All min aktivitet og observasjon på IRC-kanalene ble logget på min arbeidsstasjon. Jeg var først usikker på den etiske siden av dette, men ble snart overbevist om at dette ikke var problematisk. I flere tilfeller så jeg andre fortelle at de loggførte sine samtaler. I et tilfelle ga til og med en utvikler uttrykk for at han trodde alle loggførte sine aktiviteter på IRC. Loggingen ble foretatt i en periode fra nyttår 2006 frem til levering i slutten av april samme år. Dette utgjorde en nokså avskrekkende mengde materiale, og jeg slo fort fra meg tanken på å få det inn i datasystemet jeg hadde benyttet til e-postinnlegg tidligere. Grunnen til dette var først og fremst formen på kommunikasjonen over IRC. Temaene var mange, fragmenterte og overlappet hverandre hele tiden. Å skille disse fra hverandre som enkeltinnlegg i en debatt, samt sette fornuftige merkelapper på dem, virket som en lite rasjonell måte å nærme seg disse dataene på. I stedet landet jeg på en mye mer prosaisk metode for å katalogisere empirien, nemlig å gjøre det manuelt ved hjelp av utskrifter og gule lapper. I motsetning til e-postlistene var disse loggene faktisk for det meste stoff jeg allerede hadde sett, og det var dermed enklere for meg å finne igjen detaljer og episoder som hadde festet seg i bevisstheten. Resultatet ble ikke like skalerbart og visuelt som for e-postlistene, men godt nok til at jeg brukte det flittig som et oppslagsverk til detaljer og episoder i kunnskapsarbeidet til VideoLAN.

[12]Hovedsakelig fra deltakelse i studentproduksjoner via Student-TV i Trondheim

3.2.4 Generalisering ut i fra kvalitative data

For å relatere detaljer og episoder til en større helhet er det nyttig å ta i bruk abstraksjoner og generelle konsepter vedrørende menneskelig samhandling. Klein & Myers fjerde prinsipp om abstraksjon og generalisering er knyttet til dette. Prinsippet handler om å relatere sine forskningsresultater til teoretiske og generelle konsepter som forklarer menneskelig forståelse og sosial handling. I motsetning til positivistisk forskning gjøres ikke dette først og fremst for å bevise eller motbevise teorier, men for å sette ting i en sammenheng og se verden en på bestemt måte.

Kravet om generaliserbarhet står sterkt i den positivistiske tradisjonen, men er gjerne nedtonet i fortolkende tekster. Blant de mest aktive forkjempere for kvalitativ metode er det enkelte som hevder dette kravet grenser mot å være irrelevant. De hevder at menneskelig aktivitet ikke kan generaliseres hverken statistisk eller analytisk, utenom i sjeldne tilfeller der innsikt om menneskets natur kan brukes som analogier til andre relaterte fenomener[13]. Jeg vil ikke fremholde en så bastant holdning til generaliserbarhet i denne avhandlingen, men heller benytte abstrakte teorier og konsepter som analogier i en diskusjon rundt de tendenser og episoder som jeg finner i min empiri. Hensikten blir altså ikke å teste eller falsifisere disse teoriene, men å bruke dem som konseptuelle rammeverk til en fortolkning og dypere forståelse av case.

3.2.5 Kritisk tolkning

En konsekvens av et fortolkende perspektiv på vitenskapelig arbeid, blir at man må reflektere over hvordan man selv og forskningsobjektene innehar ulike virkelighetsoppfatninger og tolkninger av konkrete situasjoner. For forskerens del betyr dette at man må gå i seg selv og tenke over hvordan ens egne holdninger og kulturell ballast påvirker forståelsen av det man observerer. Klein & Myers (1999) kaller dette for prinsippet om dialektisk resonnering. Det innebærer at man må reflektere omkring egne forutintatte holdninger, hvilke «briller» man ser ting gjennom og mulige motsetninger mellom dette og faktiske funn.

En utfordring for meg er med andre ord å møte OS-miljøene på en mest mulig åpen og lite forutinntatt måte. Jeg har latt meg fascinere av historien og de politisk-etiske dimensjonene ved OS-bevegelsen, og må derfor være varsom for ikke å lage en «solskinnsversjon» av den virkeligheten jeg prøver å fortolke. I tillegg til å forsøke å være mine egne «briller» bevisst, har jeg også gått noen runder utenfor casene, i de tilfellene der det har vært eksterne som har ytret misnøye med den

[13]Se for eksempel Archers beskrivelse i Cornford & Smithson (1996:41)

generelle konsensus i miljøet. Dette har jeg gjort for å få flest mulig versjoner av de hendelser som jeg mener er relevante for min fremstilling. Et eksempel på dette kan være mine besøk på arenaer der jeg finner de interne utviklerne sammen med andre utviklere på «nøytral grunn», for eksempel det uavhengige diskusjonforumet på hydrogenaudio.org. Klein & Myers sjette prinsipp om multiple fortolkninger handler om nettopp hvor viktig det er å få flere synsvinkler på et og samme fenomen, siden tolkningene kan variere fra person til person og fra gruppe til gruppe.

Klein & Myers syvende og siste prinsipp handler om mistenksomhet. Forskeren må være oppmerksomme på mulige fordreininger av virkeligheten. Dette kan være bevisst eller ubevisst fra forskningsobjektenes side. I tillegg til å forstå selve dataene, må forskeren kunne lese og tolke det som skjuler seg bak de uttalte ordene. Her er det viktig å ha en god oversikt over hvilken kontekst ting blir sagt i, samt ha en formening om hvilke mulige agendaer som kan ligge bak ulike utsagn eller handlinger. For å få et mer direkte innblikk i hva som skjer mellom utviklerne i VideoLAN, har jeg intervjuet en av de mest aktive utviklerne. Fra ham har jeg fått flere synspunkter på det jeg selv har observert gjennom deltakelse på IRC-kanalene. Med flere slike korrigerende intervjuer, ville nok min forståelse av det som går for seg kunne blitt enda klarere.

4 Case: Open Source og multimedia

Mine case retter seg mot multimedia av flere årsaker. For det første er multimedia et forholdsvis ungt og dynamisk fagområde, der både standarder og bruksmønstre endres ofte. Tempoet i utviklingen skaper dermed et press på de som vil delta i kappløpet. Man bør ha et godt miljø for læring og kunnskapsoverføring, ellers vil faren for å falle av lasset være overhengende. Derfor tror jeg multimedia er et felt der fordelene og ulempene ved OS-modellen, med hensyn til kunnskapsarbeid, vil ha gode muligheter for å tre frem ved nærmere undersøkelser.

For det andre kreves det høy kompetanse for å utvikle multimedia-applikasjoner, en kompetanse som også innebærer en god porsjon erfaringsbasert skjønn, ettersom kvalitet i lyd og bilde ikke er en fast størrelse, men til en viss grad er avhengig av personlige preferanser. Dette burde åpne for hyppige faglige debatter og kontroverser underveis i arbeidet, noe som vil kunne bidra til en intensivering og tydeliggjøring av mekanismene for kommunikasjon og samhandling.

Den tredje årsaken til valget av multimedia som fokus for case, henger sammen med den

standardkrigen som nå pågår. OS-bevegelsen har ikke klart å befeste seg på denne arenaen, ikke i samme grad som de har lykkes på å etablere tekstbaserte standarder for kommunikasjon over nett. De som arbeider med dette har med andre ord tøff konkurranse, men kan potensielt få en eventyrlig mengde brukere. Multimedia har potensialet til å bli morgendagens standard for kommunikasjon, så den som lykkes å etablere seg her vil kunne få stor innflytelse på måten vi snakker og arbeider sammen på i framtida.

Min personlige interesse for open source og multimedia er også en grunn til valg av case, ettersom jeg føler meg trygg på sjargong og etikette innenfor Open Source generelt og multimedia spesielt. Uten denne bakgrunnen tror jeg risikoen ville vært atskillig større for å falle som offer for eventuelle misforståelser eller feiltolkninger, på grunnlag av manglende kulturell ballast. Det er imidlertid også en mulig ulempe ved å ha kjennskap og interesse for det aktuelle tema som skal undersøkes, i den forstand at framstillingen kan bli for mye farget av mine personlige preferanser og holdninger.

Jeg har valgt to multimediaprosjekter som fokus for denne oppgaven. Det første, *Xiph*, utvikler rammeverk for digital behandling av multimedia. Målgruppen er her først og fremst andre utviklere som ønsker å lage multimediaapplikasjoner uten å måtte betale dyre lisenser for patentert teknologi. Det andre, *VideoLAN*, lager videorelaterte applikasjoner for sluttbrukere, med fokus på video over nettverk. Disse to er valgt fordi de begge har klart å levere avanserte tekniske løsninger gjennom å benytte seg av metoder og verktøy typisk for Open Source-bevegelsen. Sammen utgjør de også et OS-rammeverk for multimedia, med mange muligheter for behandling av lyd og bilde, uten å måtte ta i bruk proprietære formater eller lisenser. I tillegg har disse to prosjektene nokså ulik bakgrunn, målgruppe og historie, noe som åpner for interessante sammenligninger og gir større bredde til undersøkelsen som sådan.

4.1 Multimedia-scenen

Multimedia har gått skrittet fra å være en tjeneste man distribuerte via lagringsmedier som CD-ROM og senere DVD, til å blitt en mer og mer vanlig tjeneste over internett – som en følge av økt båndbredde hos den enkelte bruker. Per i dag er dette i mest form av korte filmklipp, trailere og lignende. Vi ser imidlertid stadig flere tilbydere av mer omfattende former for multimedia over nett, i form av for eksempel direkte visning av spillefilmer (tilbys av blant annet av nettleverandøren NextGenTel i Norge, der man kan leie tilgang til direkte overførte filmer, der betaling går over nettregningen). For å ta et annet hjemlig eksempel, så har Norsk Filminstitutt begynt å utvikle et

eget filmarkiv, der man kan kjøpte tilgang til direktevisning av en rekke enkeltproduksjoner[14].I begge disse tilfellene har Microsoft sin videostandard, Windows Media, blitt foretrukket, i likhet med svært mange lignende tjenester på internett per i dag.

På det bærbare musikk-markedet, er fortsatt MP3 det mest brukte formatet, mens både Windows Media og Apple gjør store innhogg. Spesielt Apple har fått fart på sakene gjennom sin svært populære portable mediespiller, iPod, og nettbaserte betalingstjenester for nedlasting.

Apple har også blitt en viktig aktør gjennom inkorporeringen av MPEG-gruppens H.264-standard i de nyeste versjonene av Apples operativsystem, OS X. Ved hjelp av denne kodeken kan operativsystemet vise video i lavkvalitet, til bruk i enkle videokonferanser, og i svært høy kvalitet til visning av for eksempel spillefilmer. Denne kodeken er planlagt implementert i etterfølgeren til DVD-standarden, selv om de her er i sterk konkurranse med Sonys nye lagringsformat (Blue-Ray).

Jeg har sett på to OS-prosjekter som fortsatt er små aktører i markedet, men likevel begge har det tekniske potensialet til å kunne sammen utgjøre en åpen standard for multimedia over nett. Disse er Xiph, som først og fremst utvikler kodeker, og VideoLAN som utvikler løsninger for avspilling og direkte distribusjon over internett.

4.2 CASE1; Xiph

Xiph = Xiphophorus (en fiskeart, samt gresk for «sverdbærer»)

4.2.1 Bakgrunn og utbredelse

Xiph.org er en internettbasert organisasjon av systemutviklere som benytter OS-metoder og verktøy til å lage filformater og kodeker for multimedia. I første rekke er organisasjonen kjent for å ha utviklet filformatet OGG og lydkodeken Vorbis, begge utviklet under GPL-lisensen. Dette arbeidet ble startet av Christopher Montgomery i 1998, som en reaksjon på innføringen av lisensavgift på det svært populære MP3-formatet. OGG Vorbis skulle være et alternativt rammeverk for lagring og distribusjon av digital lyd, som kunne brukes som grunnlag i utviklingen av nye applikasjoner for lydarbeid, uten å være avhengig av å skaffe dyre lisenser fra ulike patentholdere.

OGG er et uttrykk hentet fra et dataspill kalt «Netrek», og har fått følgende betydning:

[14]http://www.nfi.no/arkivbibliotek/digitalt.html

«To do anything forcefully, possibly without consideration of the drain on future resources. "I guess I'd better go ogg the problem set that's due tomorrow." "Whoops! I looked down at the map for a sec and almost ogged that oncoming car."» (http://jargon.watson-net.com/jargon.asp?w=ogg)

Bakgrunnen for valget av navnet, var at arbeidet med kodeken ble begynt på en tid hvor maskinkraften ville hatt problemer med å håndtere det ferdige produktet. Programvaren «ogget» maskinvaren, slik Montgomery uttrykker det på Xiphs hjemmesider. Navnet Vorbis kommer fra en karakter i Terry Pratchetts bøker.

Arbeidet med selve lydkodeken har i hovedsak vært gjort av Montgomery selv, som i 2002 kunne annonsere versjon 1.0, som ble svært godt mottatt blant programmerere og lydentusiaster. Etter 2002 har utviklingen stagnert noe, frem til 2004 da Montgomery annonserte at han ville gå over til en mer administrativ rolle i Xiph. På siste nettmøte annonserte imidlertid Montgomery at han nå hadde muligheten til å delta mer aktivt i kodingen igjen, noe som lover godt for den videre utviklingen. Både filformatet OGG og kodeken Vorbis er i dag støttet av mange multimediaspillere, deriblant den populære WinAmp. OGG Vorbis høster også lovord for lydkvalitet i komparative tester[15]. Spillindustrien har etter hvert omfavnet OGG Vorbis i sine produksjoner. Flere større spillprodusenter benytter både formatet og kodeken i sine utgivelser[16]. Eksempler på titler som benytter OGG Vorbis for musikk og lydeffekter inkluderer bestselgende spillserier som Doom, Unreal, Myst, Halo, Hitman, FarCry og Harry Potter. Også i onlinespill-bransjen blir OGG Vorbis benyttet, for eksempel i Lineage, Tribes og EVE Online. Ved spørsmål om hvorfor OGG Vorbis blir foretrukket, er svaret som oftest at man får samme eller bedre kvalitet i forhold til kommersielle konkurrenter, samtidig som man slipper å betale lisenser.

Etter hvert som OGG Vorbis ble mer utbredt og støttet av flere mediespillere, oppsto det et behov for raskere videreutvikling av lydkodeken. Montgomery fikk knyttet til seg et knippe med utviklere, og dannet organisasjonen xiph.org som rammeverk for sitt arbeid. Under denne paraplyorganisasjonen vokste det fram flere andre multimediaprosjekter, med et felles mål om å etablere en fullverdig lisensfri multimediaplattform basert på åpen kildekode. En av de mest aktive prosjektene, Theora, er konsentrert rundt utviklingen av et OS-alternativ til de mange proprietære og patentbelagte videoløsningene som verserer på internett (i første rekke mpeg-baserte løsninger (mpeg4, H.264 m.fl.), Apples Quicktime og Microsofts Windows Media). Hensikten er, på lik linje med OGG Vorbis, å lage en kodek som ikke benytter eksisterende patenter, og dermed ikke krever

[15] For en utførlig lyttetest med flere ulike formater se http://www.maresweb.de/listening-tests/mf-128-1/results.htm
[16] Se http://wiki.xiph.org/index.php/Games_that_use_Vorbis for en liste over spill flere som bruker OGG Vorbis

kostbar lisensiering.

Arbeidet med å lage en patentfri videokodek ble startet for alvor etter at Xiph ble kontaktet av en kommersiell aktør i det digitale videomarkedet, nemlig On2. Dette firmaet hadde lenge jobbet med digital videoteknologi, og ønsket å inkorporere en av sine eksisterende kodeker, VP3, i filformatet OGG. VP3 ble relisensiert under LGPL-lisensen og Vorbis-teamet kunne dermed starte arbeidet med å bygge et fullverdig OS-alternativ til proprietære løsninger for koding og distribusjon av digital video. OS-versjonen av VP3 ble omdøpt til Theora (hentet fra kultserien «Max Headroom»). On2 har støttet dette arbeidet kontinuerlig frem til dagens alphaversjon av kodeken, som ble lansert i november 2005. Det finnes flere multimediaspillere som støtter både Vorbis og Theora, deriblant VLC, Mplayer, Xine, Helix og ffmpeg. Ved hjelp av små programtillegg kan man også få støtte for Vorbis og Theora i Windows Media Player og Realplayer.

I tillegg til Vorbis og Theora, har xiph.org knyttet til seg flere andre prosjekter, deriblant to som begge behandler lyd, men til forskjellige formål. Den ene kalles Speex, og er utviklet for utelukkende å behandle tale mest mulig effektivt i forhold til ressursbruk, båndbredde og lagringsplass. Den andre kalles FLAC og er en tapsfri lydkodek med komprimering. Filer laget med FLAC tar mer plass og krever mer ressurser enn for eksempel OGG Vorbis, men til gjengjeld er kvaliteten like god etter koding som de originale rådata. Et annen populær prosjekt som har etablert seg under Xiph-paraplyen, er Icecast, et OS-alternativ til den mye brukte SHOUTcast-protokollen (som benyttes blant annet i mediespilleren WinAmp for direkte overføring av lyd over internett).

Jeg vil i den videre fremstillingen konsentrere mine undersøkelser til de to av de mest toneangivende prosjektene til Xiph; Vorbis og Theora. Grunnen til dette er at det fort blir svært komplekst å dra alle underprosjektene inn i diskusjonen, med deres små variasjoner og særegenheter. Dette betyr ikke at de andre prosjektene er blitt ignorert, men at de ikke vil bli trukket frem, med mindre de innehar avvikende trekk som er relevante for den pågående fremstillingen. Ved å fokusere på det mest toneangivende av prosjektene i Xiph, mener jeg å kunne fange inn de viktigste trekkene ved organisasjonen – både når det gjelder arbeidsform og intern struktur.

4.2.2 Organisasjonsstruktur

Xiph.org omtaler seg selv som et ”velmenende diktatur” på sine offisielle hjemmesider. Grunnleggeren Christopher «Monty» Montgomery har siste ordet i alle saker, men har delegert bort

mange oppgaver til en komité bestående av enkelte sentrale utviklere i organisasjonen. I tillegg er det utnevnt ansvarspersoner for hvert sitt delprosjekt innenfor xiph-paraplyen. Disse er som regel enten initiativtakere til et prosjekt eller i det minste aktive og sentrale bidragsytere, og står rimelig fritt i forhold til paraplyorganisasjonen Xiph ved spørsmål som gjelder eget prosjekt. Hvem som blir ansvarspersoner er i liten grad bestemt av komiteen, men utnevnes uformelt ut i fra frivillighet, interesse og innsats, som denne lille ordvekslingen viser eksempel på:

> «
> *> Do we have a build system maintainer?*
> You can yell at me if you want.
> »
> *(http://lists.xiph.org/pipermail/vorbis-dev/2005-January/017889.html)*

Diskusjoner og konflikter vedrørende Xiph-prosjekter, tas som regel på hvert enkelt prosjekts IRC-kanal eller e-postliste. Større diskusjoner havner som regel på e-postlistene, mens mindre omfattende problemer håndteres fortløpende via IRC. Hver måned arrangeres det i tillegg et åpent møte på IRC for hele Xiph, med forhåndsannonsert agenda. På disse møtene oppdaterer de ulike delene av organisasjonen hverandre på status og framdriftsplaner, og man diskuterer fremtidig veivalg og arbeidsfordeling i en forholdsvis formell tone. I tillegg inviteres det av og til eksterne kontakter til diskusjoner omkring konkrete saker. Både møteplaner og logg fra selve møtene er åpent tilgjengelig på Xiph sine nettsider.

4.2.3 Arbeidsflyt og verktøy

Ettersom det er i hovedsak en person som har ansvar for en enkelt modul, er det sjeldent at utviklerne er nødt til å arbeide sammen. Som regel vil hver enkelt utvikler fikle med sin modul, og legge dette direkte inn i kodebasen ved hjelp av Subversion. Hvis endringene som skal gjøres kan potensielt ha konsekvenser for flere andre moduler, er det vanlig praksis å opprette en alternativ gren i kodebasen. Alle endringer som gjøres i Subversion blir automatisk postet i en egen e-postliste kalt «commit», i tillegg blir samme informasjon annonsert automatisk i prosjektets IRC-kanal. Eksterne bidrag blir gjerne sendt direkte til e-postlistene slik som i eksempelet under:

> here is a patch that fixes various bugs/problems with avi2vp3:
>
> 1. open() is not called with O_BINARY , this causes the program to
> crash or give undesired results on win32 (and maybe other
> platforms?), as binary mode is not default.
>
> 2. win32 does not understand chmod flags, ie:
> (S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)

```
3. buffer = malloc(AVI_max_video_chunk(avifile));
for some reason, AVI_max_video_chunk(avifile) returns 0, i've
set an arbitrary high value for it until someone fixes it
correctly.
```

(http://lists.xiph.org/pipermail/theora-dev/2005-February/002680.html)

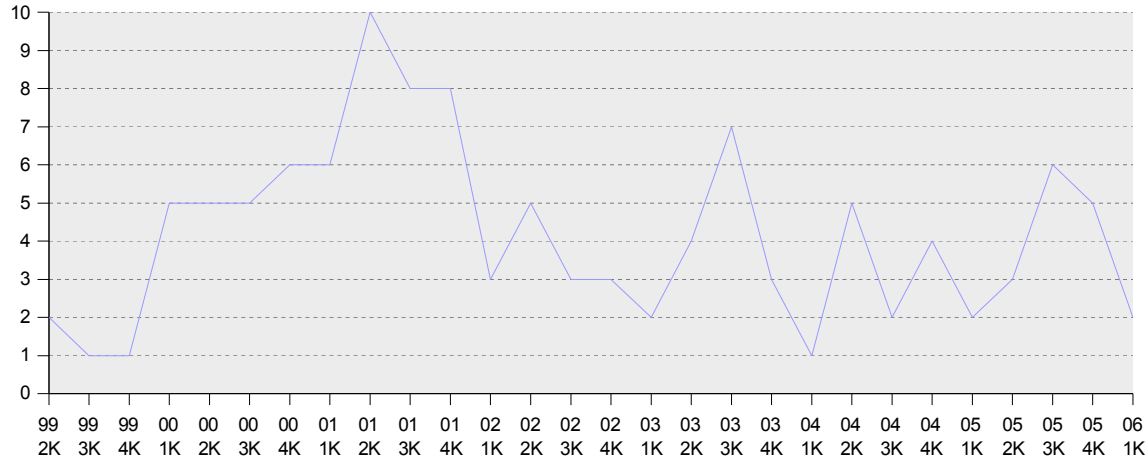
Ved gjentatte gode bidrag vil sannsynligvis noen i prosjektet gi den eksterne utvikleren skrivetilgang til Subversion, hvis denne selv ber om det.

Problemer, feil og mangler blir logget ved hjelp av TRAC, der man også kan finne oversikt over fullførte og planlagte milepæler for hvert enkelt prosjekt. Dette er et forholdsvis nytt verktøy for mange, i forhold til e-postlister og IRC, ergo er det ikke alle som bruker det like mye.

4.2.4 Utviklingen i tall

For å få et mer detaljert bilde av hvordan utviklingen av de to mest omtalte Xiph-prosjektene (Vorbis og Theora) har foregått, har jeg valgt å hente ut noen nøkkeltall fra kodebasen i Subversion. Hvis vi starter med det eldste prosjektet, lydkodeken Vorbis, kan vi i figur 1 se hvor mange aktive utviklere som har bidratt direkte med endringer i kodebasen de siste seks-syv årene. X-aksen viser her tidslinjen oppdelt kvartalsvis, mens Y-aksen viser antall utviklere som gjorde endringer i koden i angitt periode.

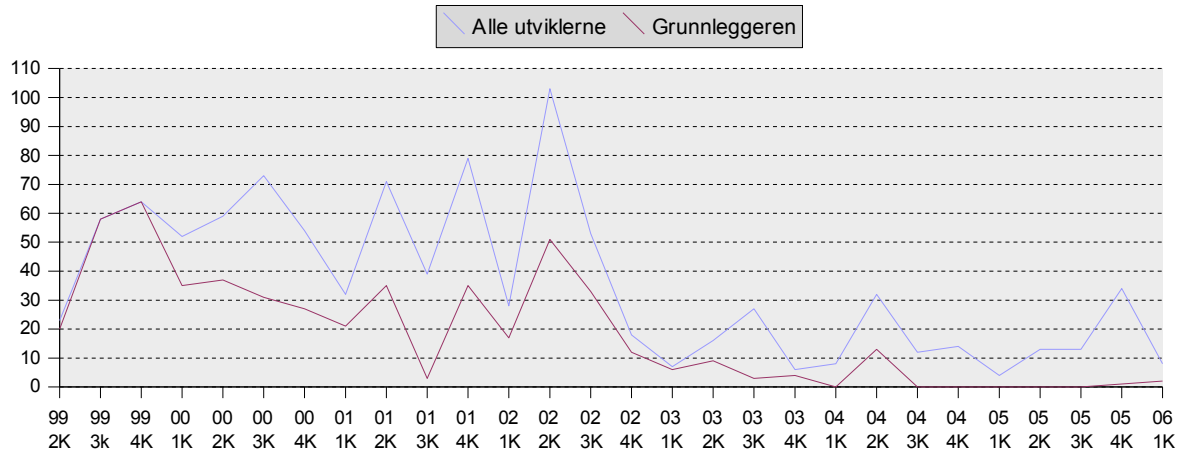
Fig.1: Antall utviklere som gjorde endringer i Vorbis, per kvartal



Vi kan i figur 1 observere en stigning i antallet aktive bidragsytere frem til utgivelsen av beta versjon 4 av Vorbis, og den offisielle dannelsen av Xiph Foundation som en «non-

profit»-organisasjon, i første kvartal 2001. I kjølvannet av dette dobles nesten antallet utviklere, før det igjen faller ned på omtrent samme nivå som før lanseringen. Etter dette holder antallet seg på et rimelig stabilt nivå frem til i dag, med unntak av noen større topper i 2003 og 2005. Bevegelsene mot slutten av 2003 bør man ikke legge særlig vekt på, siden det generelle utviklingsnivået var nokså lavt, og et par nye utviklere leverte kun ett eller to bidrag før de forsvant fra prosjektet igjen. Toppen i 2005 skyldes til dels at Vorbis fikk nye utviklere med fokus på Windows-plattformen, som har fortsatt å bidra. Fallet i 2006 er sannsynligvis spuriøst, siden tidsperioden ikke er avsluttet enda. Det virker altså som at Vorbis tiltrakk seg jevnt flere utviklere, med høydepunkt rundt lanseringen av beta 4, for deretter å falle litt ned på et nivå på tre til fire aktive utviklere.

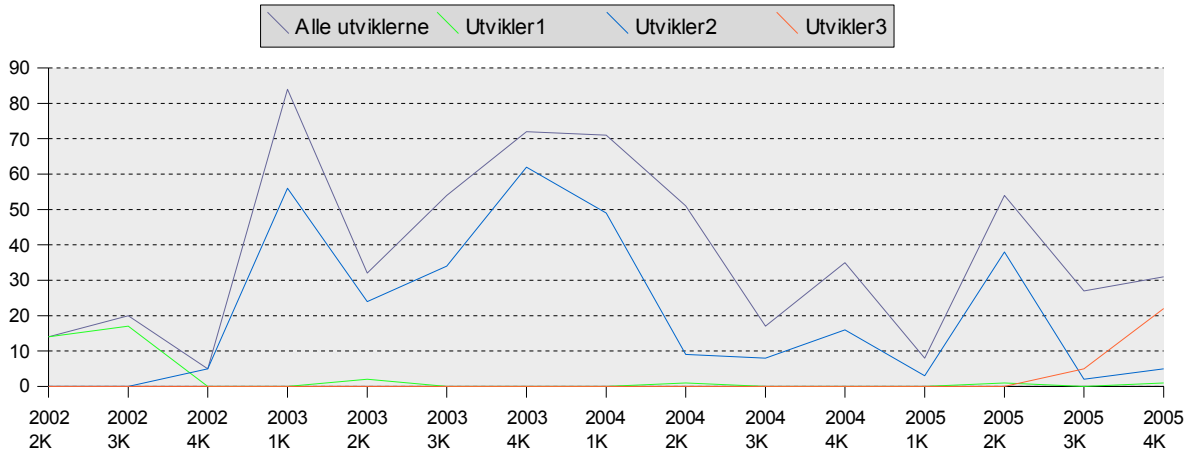
Fig.2: Antall endringer i kodebasen til Vorbis, per kvartal



Vi ser i figur 2 hvor mange endringer som har blitt gjort i kodebasen per kvartal. Jeg har valgt å vise alle utviklerne sammenlagt, samt å vise grunnleggeren av Vorbis som en egen graf. Dette illustrerer hvor viktig grunnleggeren i dette prosjektet har vært, og er, for prosjektets fremdrift. Grunnleggeren utfører til en hver tid ikke mindre enn halvparten av endringene i kodebasen, som regel mer. Vi ser også at veksten i antall utviklere før 2002 (fra figur 1) i liten grad er med på å bestemme hvor stor trafikk det er i kodebasen, her er grunnleggerens bevegelser igjen avgjørende. Etter flere perioder med vekslende aktivitet, ser vi et tydelig løft frem til midten av 2002, som topper seg med lanseringen av Vorbis 1.0 i juli. Deretter synker aktiviteten hos grunnleggeren nesten ned mot null, før den opphører helt høsten 2004 – i det han annonserer en overgang til en mer administrativ rolle. En medvirkende årsak til at utviklingen bremset opp etter 2002, kan være enkelte uoverensstemmelser innad i utviklergruppa, der det var uenighet om noen av valgene som var gjort med hensyn til programarkitektur. Samtidig gjorde Montgomery i liten grad forsøk på å gi fra seg

lederskapet for Vorbis, selv om han selv hadde liten tid til å bidra. Dette førte etter hvert til at alternative versjoner av Vorbis ble lansert, ettersom behovet for å optimalisere kodeken økte. En japanske utvikler lagde sin egen optimaliserte versjon av både kodeken og enkoderprogrammet til vorbis, kalt «aoTuV[17]». Denne ble foretrukket av mange på grunn av bedre ytelse enn vorbis sine egne versjoner, der utviklingen gikk tregere. Lenge var det lite åpning for å inkludere aoTuV i den offisielle vorbis-versjonen[18], men i 2005 ble likevel aoTuV sine endringer inkludert i lanseringen av vorbis-versjonene 1.1 og 1.1.1.

Fig.3: Antall endringer i kodebasen til Theora, per kvartal



Når det gjelder utviklingen av Theora, har det vært noen perioder med høy aktivitet, men jevnt over har det vært få utviklere som har bidratt. Det har sjeldent vært mer enn 5 aktive, og av disse har som regel en person stått for mesteparten av endringene. I figur 3 har jeg listet opp endringer i kodebasen, med grafer for alle utviklerne samlet, og for tre viktige utviklere enkeltvis. Den første, Utvikler1, er den samme som grunnla OGG Vorbis, og vi ser av figuren at han står for nesten all aktiviteten i løpet av 2002, før han forsvinner ut av bildet. Dette korresponderer med utgivelsen av Vorbis 1.0, og bekrefter at grunnleggeren faset ut av flere Xiph-prosjekter i denne perioden. Vi ser imidlertid at en annen utvikler tar over stafettpinnen, Utvikler2, i motsetning til det som skjedde i tilfellet med Vorbis. Denne utvikleren står videre for brorparten av aktiviteten helt frem til i dag, hvor en ny utvikler, Utvikler3, begynner å vise tegn til å kunne ta over prosjektet ved en senere anledning[19]. Vi ser med andre ord at selv om hverken tempoet i utviklingen eller mengden utviklere er særlig imponerende, har Theora-prosjektet likevel klart å holde det gående, på tross av skifte av

[17]Egen hjemmeside på http://www.geocities.jp/aoyoume/aotuv/
[18]Se avklaringen på http://lists.xiph.org/pipermail/vorbis/2004-June/025181.html
[19]Her har det ikke vært foretatt noe formelt skifte, jeg bare registrerer at en utvikler har vist ekstra innsats i senere tid.

pådriver, eller hovedbidragsyter om man vil.

4.2.5 Status

Ogg Vorbis har hatt liten utvikling siden lanseringen av versjon 1.0 i 2002, kun mindre oppdateringer og fiksing av feil. Dette kan henge sammen med at hovedutvikleren ikke lenger kunne bruke så mye tid på prosjektet, men heller ikke forsøkte å finne en klar etterfølger. I tillegg var det uenigheter om hvordan programmet skulle bygges opp som virket demotiverende på utviklingen. Dette har gjort at OGG Vorbis har tapt en del terreng til andre kommersielle varianter, spesielt i forhold til overføring over nett med lav bitrate[20]. Siden april 2006 har imidlertid Montgomery signalisert at han igjen har tid og overskudd til å gjøre en større innsats overfor utviklingen med Vorbis. Jeg har allerede registrert en økende aktivitet på IRC og e-postlistene, der han forsøker å gjøre seg ajour med hva alle de andre holder på med, og hva som har blitt gjort.

Den kommersielle suksessen som man hadde håpet var innom rekkevidde, etter at vorbis raskt fikk et fotfeste innen spillindustrien, har i stor grad uteblitt på andre deler av markedet. MP3 er fortsatt toneangivende, mens Windows Media Audio og AAC fra Apple begynner har vokst seg litt større. En av grunnene til dette kan ligge i usikkerhet rundt bruken av patenter i vorbis. Det har gjentatte ganger vært stilt spørsmålstegn ved Xiph sine påstander om at OGG Vorbis ikke er heftet med patenter. En lengre teknisk diskusjon tok plass på hydrogenaudio.org i løpet av 2003, der flere konkrete patenter ble trukket inn i diskusjonen[21]. Ingen offisielle representanter fra Xiph tok aktivt del i debatten, etter råd fra sine advokater, noe som skapte en usikkerhet rundt patentspørsmålet. Dette kan ha virket dempende på kommersielle interessenter, ettersom det kan føles tryggere å betale for en lisens til et produkt som oppgir hvilke patenter det bruker (for eksempel MP3), enn å benytte et produkt som påstår det ikke bruker noen patenter.

Theora har ikke klart å etablere seg som en konkurransedyktig videokodek enda, sett i forhold til de kommersielle konkurrentene. I en test på en uavhengig nettside for videoentusiaster, ble ikke Theora regnet som god nok til å kvalifisere til testens siste del[22]. Dette er forsåvidt ikke

[20]Lav bitrate vil si at man forsøker å spille av lyden med minst mulig data per sekund med lyd; høy komprimering.
[21]http://www.hydrogenaudio.org/forums/index.php?showtopic=13686
[22]Se http://www.doom9.org/index.html?/codecs-quali-105-3.htm

overraskende, med tanke på at Theora fortsatt er i tidlig betafase. Enkelte av utviklerne av Theora har begynt å arbeide ved Fluendo (www.fluendo.com), et selskap som baserer seg på å selge komplette videoløsninger over nett, kun ved hjelp av OS-programmer (i første rekke OGG Vorbis og Theora). Dette kan både gi dem en kanal for bedre kontakt med potensielle brukere, og en mulighet for større økonomisk støtte og trygghet.

4.3 CASE2; Videolan

VideoLAN = Video over LAN (Local Area Network, lokalnett på norsk)

4.3.1 Bakgrunn og utbredelse

VideoLAN ble startet som et studentprosjekt ved Ecole Centrale Paris i 1996. Utgangspunktet var todelt; studentene ønsket å kunne se tv-sendinger på egne pc-er, samtidig som skolens nettverksdrift ønsket en oppgradering av lokalnettet, og derfor var på utkikk etter et prosjekt med høye krav til båndbredde. Prosjektet ble etablert som et valgbart emne for andreårs-studenter ved skolen, og tiltrakk seg i løpet av kort tid et knippe med interesserte studenter som la ned atskillig flere arbeidstimer enn det som som var vanlig blant de andre skoleprosjektene i dette årstrinnet. Enkelte av studentene ble involvert gjennom eget initiativ allerede fra første årskull, og mange valgte også å bidra til prosjektet etter at den obligatoriske aktiviteten var fullført.

Prosjektet ble fra starten av delt inn i en videotjener (VideoLAN server – VLS) og en videoklient (VideoLAN Client – VLC), og lykkes i 1998 med å sende og motta video i standard kringkastings-kvalitet (MPEG2). De to delprosjektene arbeidet videre parallelt, med hver sin studentgruppe som utviklere. Etter lang tids press fra studentene ble prosjektene i 2001 lisensiert under GPL-lisensen og åpnet for eksterne utviklere. Dette resulterte i en eksplosiv vekst både i antall utviklere og i utviklingsrate. Både VLC og VLS var opprinnelig skrevet for Linux, men VLC var fra starten av konstruert med portabilitet for øyet, det vil si at det skulle være enklest mulig for programmerere å lage både klienter og tjenere for andre operativsystemer. Dette ga ekstra vind i seilene til VLC, i forhold til VLS som ikke i like stor grad klarte å tiltrekke seg nye utviklere og brukere.

Allerede seks måneder etter lisensiering til GPL ble den første versjonen av VLC for Microsoft Windows lagt ut, som skapte grobunn for nye brukere. VLC for Windows ble ikke umiddelbart en suksess, sannsynligvis på grunn av stor konkurranse med andre etablerte mediespillere som Windows Media Player, WinAmp m.fl. På Apples operativsystem MacOSX var det imidlertid få

store konkurrenter, spesielt på videosiden, og når VLC ble portet dit fikk den i løpet av kort tid stor utbredelse blant Mac-brukere. I dag utgjør likevel MS Windows den største brukerbasen for VLC, sannsynligvis på grunn av dette operativsystemets nesten enerådende markedsandel blant vanlige forbrukere.

I løpet av 2003 oppdaget noen av studentene i VLC-gruppa en enkel metode for å utvide klienten til også å kunne fungere som videotjener. Dette gjorde at VLS ikke lengre var nødvendig for å utføre de fleste tjeneroppgaver, og studentene søkte seg nå til VLC for videre utvikling. Utviklingen på VLS fortsatte en stund, om enn i et mye lavere tempo enn VLC, og med atskillig mer begrenset funksjonalitet. Per i dag er det svært liten aktivitet rundt VLS, og VideoLAN anbefaler på sine nettsider å benytte VLC til både tjener- og klientformål. Jeg har derfor valgt å fokusere på VLC i den videre fremstillingen av VideoLAN.

En av hovedårsakene til VLCs popularitet blant sine brukere har nok vært på grunn av dens tekniske finesser. Allerede fra tidlige versjoner har VLC kunnet skryte av stor bredde i funksjonalitet, sammenlignet med sine konkurrenter, og har vært raskt ute med støtte for en lang rekke standarder, kodeker og filformater for lyd og bilde. I motsetning til de fleste andre mediespillere, har VLC valgt å inkludere egne kodeker i spilleren, i stedet for å bruke de kodekene som er installert i operativsystemet. Dette har gjort det enklere for sluttbrukeren, ettersom VLC dermed kan spille de fleste videoformater direkte, uten å måtte bry seg med å laste ned mange ulike kodeker. Dette har også vært gjenstand for kritikk, siden VLC med denne arkitekturen tar fra brukerne kontrollen over kodekene. For de aller fleste brukere vil nok likevel fordelene ved VLC sin modell lett overskygge den potensielle ulempen det innebærer å ikke kunne håndtere kodeker uavhengig av spilleren.

4.3.2 Organisasjonsstruktur

Videolan har kun noen få offisielle formelle roller i organisasjonen, inkludert en prosjektleder, en finanssjef, en sjefskonsulent / maskot og to lærere fra Ecole Centrale Paris. Det finnes i praksis lite hierarkisk beslutningsmyndighet i VideoLAN. Ukentlige møter blant studentene på Ecole Centrale Paris var før en viktig arena for beslutninger, både av teknisk og organisatorisk art. I dag er imidlertid hovedvekten av utviklerne eksterne, og disse møtene har dermed mistet sin funksjon i så måte. Per i dag ser alle saker ut til å bli tatt opp via offentlig tilgjengelige kanaler, der enighet når gjennom mer eller mindre opphetede diskusjoner på VideoLANs IRC-kanal eller e-postlistene (som regel på førstnevnte). Her har alle i utgangspunktet lik rett til å hevde sine synspunkter, men vekten av hver enkelts argumenter vil gjerne bli veid opp i mot deres fartstid og generelle bidrag til

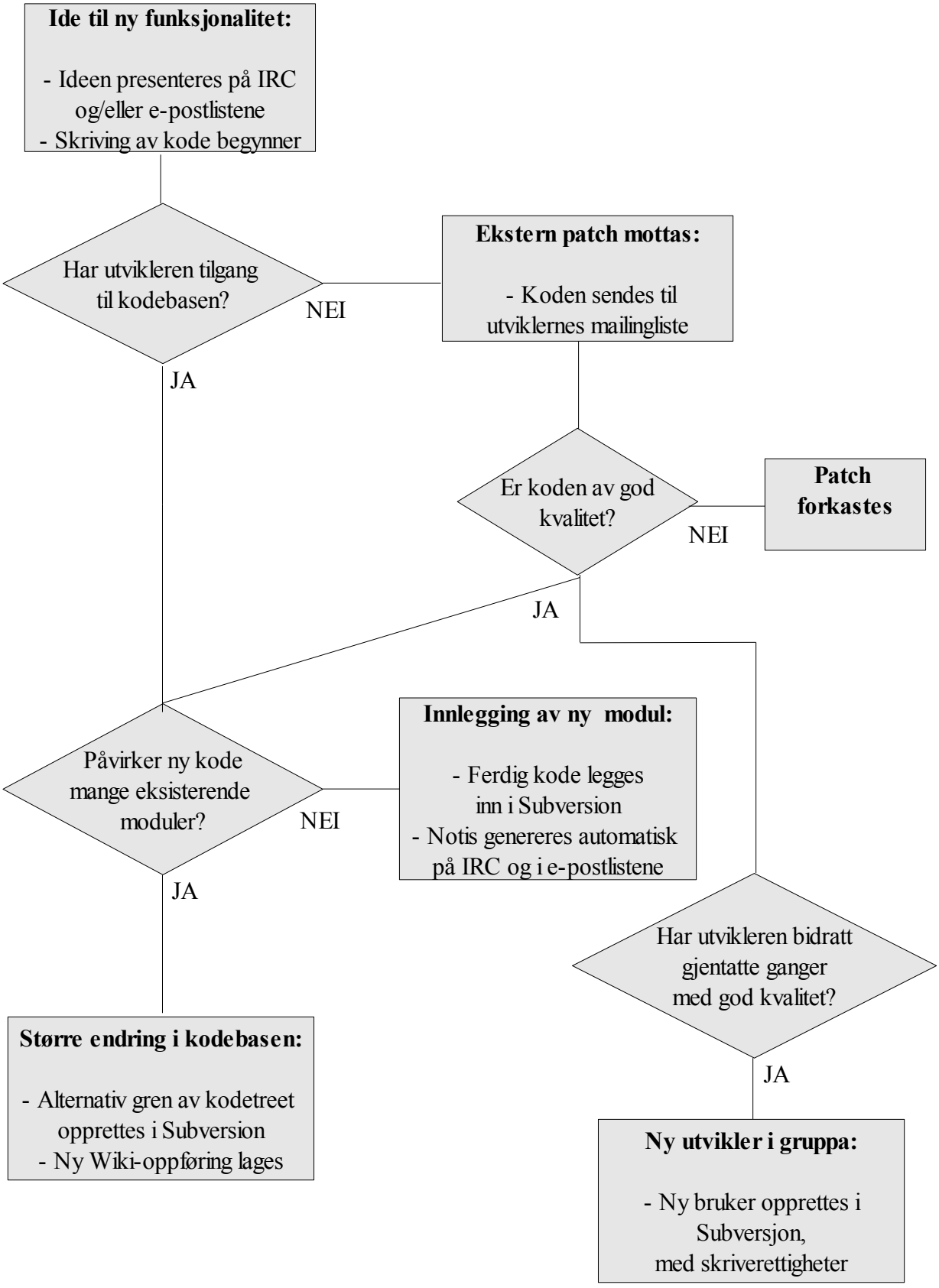
organisasjonen. I tillegg vil hver enkelt utvikler ha en naturlig autoritet i forhold til spørsmål som angår sin modul. Kunnskap og erfaring, demonstrert gjennom kodeprestasjoner, er med andre ord viktige faktorer for hvor mye autoritet den enkelte opererer med.

De viktigste arenaene for kommunikasjon er IRC, e-postlister, nettbaserte forum og Wiki-er. Da hovedvekten av utviklerne fortsatt var ved Ecole Centrale Paris, ble det holdt ukentlige møter mellom utviklerne, der både tekniske og ikke-tekniske tema ble tatt opp. I dag er hovedvekten av utviklerne eksterne, og det er derfor i hovedsak de nettbaserte arenaene som blir benyttet til tekniske spørsmål. Endringer av VLC blir her evaluert ved hjelp av en uformell prosess, der alle utviklerne får gi sin vurdering. Jo flere deler av VLC som påvirkes av en eventuell endring, jo flere av utviklere tar del i denne prosessen, som foregår fortløpende på IRC – den primære kommunikasjonsarenaen for utviklergruppa. Juridiske spørsmål, samt kontakt utad mot media eller kommersielle aktører, blir tatt hånd om lokalt ved Ecole Centrale Paris. Her det også etablert en større testplattform. Kontakten med sluttbrukerne foregår enten via offentlige forum, via dedikerte e-postlister for brukere (som de fleste av utviklerne abonnerer på), eller direkte gjennom VideoLANs IRC-kanal på freenet.

4.3.3 Arbeidsflyt og verktøy

Hvis en utvikler hos Videolan får en ide til ny funksjonalitet, vil han som regel starte dette arbeidet på egen hånd og fortelle hva han driver med til resten av utviklergruppa på IRC. Når koden begynner å ta skikkelig form, vil utvikleren legge koden inn i en felles kodebase ved hjelp av Subversjon, sammen med en kommentar som beskriver hva dette er. Subversjon genererer deretter automatisk en e-post med beskrivelse av koden, som blir sendt til en intern epostliste som alle utviklerne abonnerer på. Hvis utvikleren ikke er en fast bidragsyter hos VideoLAN, kan han sende koden sin direkte til utviklernes epostliste. Denne koden blir så evaluert internt, og lagt til i Subversjon hvis blir funnet god nok. Ved gjentatte bidrag av høy kvalitet vil vedkommende bli gitt en egen konto på Subversjon med fulle skriverettigheter og er dermed blitt en del av VideoLAN-gruppa.

Skjema 1: Fra ide til ferdig kode, forenklet oversikt:



Ny kode som legges inn i Subversjon blir testet rutinemessig ved Ecole Centrale Paris. De andre utviklerne vil deretter komme med en uformell evaluering av den nye koden, enten basert på testingen som har blitt gjennomført, eller på generelt grunnlag. Hvis den nye koden er avhengig av, eller overlapper med, andres arbeid, vil de berørte også komme med sine synspunkter. Alt dette foregår som regel fortløpende på IRC, med unntak av svært store og omfangsrike ideer, som i noen tilfeller vil få sin egen Wiki-oppføring. Hvis det blir laget en egen Wiki, vil denne bli gjenstand for kontinuerlig revisjon av utviklere som er direkte involvert, eller som har sterke meninger om VLC generelt. Hvis endringene griper om seg mange ulike deler av VLC, vil det som regel også bli opprettet en egen «gren» i Subversjon. Koden kan så senere integreres i «hovedtreet», som utgjør den offisielle VLC-koden, etter at den har blitt testet skikkelig og utviklergruppa er blitt enige om at det nye bidraget holder mål.

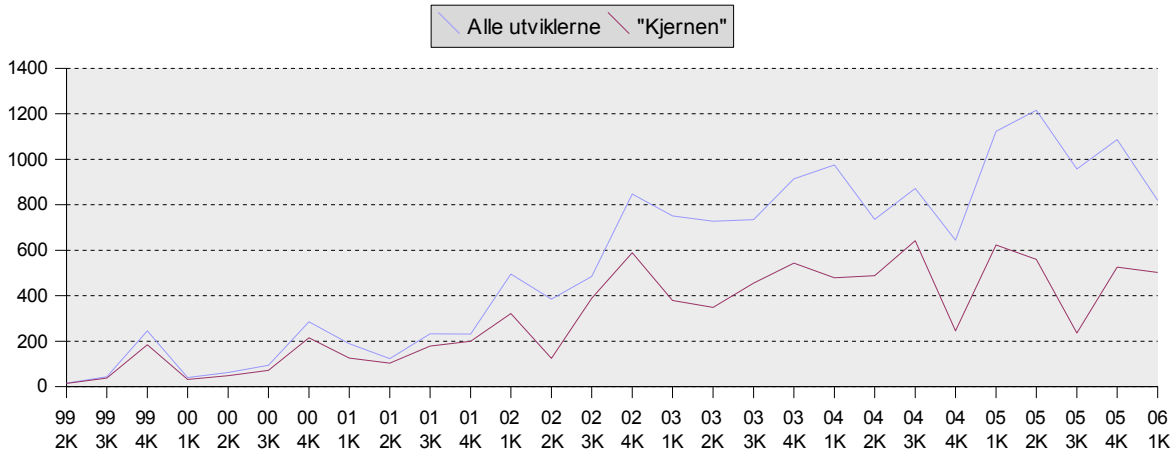
Selv om VLC er satt sammen av svært mange utviklere, trenger ikke dette bety at utviklerne arbeider på samme funksjonalitet. Tvert i mot er det svært uvanlig at flere utviklere arbeider på samme delen av VLC, det vanlige er heller at en person har ansvaret for én modul, og utvikler denne på egen hånd. Behovet for samhandling kommer først og fremst når ulike moduler skal fungere sammen, gjerne ved innføring av et nytt grensesnitt til en modul, eller ved utvidelse til å støtte nye formater eller standarder. I slike tilfeller vil utviklerne i første omgang sjekke hverandres kildekode og se om man klarer å løse eventuelle problemer på egen hånd. Hvis ikke koden er selvforklarende vil utviklerne ta kontakt med hverandre, som regel på IRC, og utveksle den informasjonen som er nødvendig for å løse det aktuelle problemet. Hvis den opprinnelige forfatteren av en kodesnutt ikke lenger er tilgjengelig, vil noen måtte dykke ned i koden og forsøke å gjøre seg kjent med den på egenhånd.

Etter hvert som flere nye moduler blir lagt til i kodebasen, testet og funnet gode nok, vil utviklergruppa diskutere seg fram til en felles målsetting for neste offisielle versjon av VLC. Innholdet i denne målsettingen blir konkretisert som milepæler i TRAC, som blir fortløpende oppdatert etter hvert som de ulike utviklerne gjør fremgang på hver sin modul. Når gruppa er enig om at målsettingen er nådd, blir det laget en egen utgivelsesgren i kodebasen, som blir frosset for nye endringer intill den er behørig testet. Etter noen runder med nye utgivelseskandidater, testing, og fiksing av eventuelle feil, blir utviklerne til slutt enige om å utgi en ny offisiell versjon av VLC. Denne blir så distribuert ganske enkelt ved å gjøres tilgjengelig for nedlasting fra VideoLANs hjemmesider.

4.3.4 Utviklingen i tall

Ved å se nærmere på bevegelsene i kodebasen, kan man danne seg et bilde av hvordan utviklingsarbeidet har gått for seg siden første versjon av VLC ble lastet inn i Subversion. Grafene i figur 4 og 5 er basert på endringer i kodebasen, registrert gjennom Subversion. Figur 4 viser hvor mange endringer som har blitt gjort fra første oppføring, 2.kvartal 1999, til skrivende stund, 1.kvartal 2006. X-aksen viser tidslinjen, oppdelt kvartalsvis, mens Y-aksen viser antall endringer. Den grå grafen viser data for alle utviklerne samlet, mens den lilla viser data for det jeg har kalt «kjernen». Det viste seg nemlig, ved nærmere ettersyn, at de tre til fire mest aktive utviklerne sto for en anselig andel av endringene. «Kjernen» i figur 1 har blitt beregnet per kvartal, og består av alle utviklere som alene har stått for 10% eller mer av det aktuelle kvartalets endringer. «Kjernen» utgjorde mellom to og fem utviklere, med tre som det vanligste i de fleste kvartalene. Her gikk mange av de samme personene igjen som de mest aktive. Den andre figuren viser antallet utviklere som var inne og utførte endringer i den samme tidsperioden som figur 4, igjen med kvartalsvis inndeling.

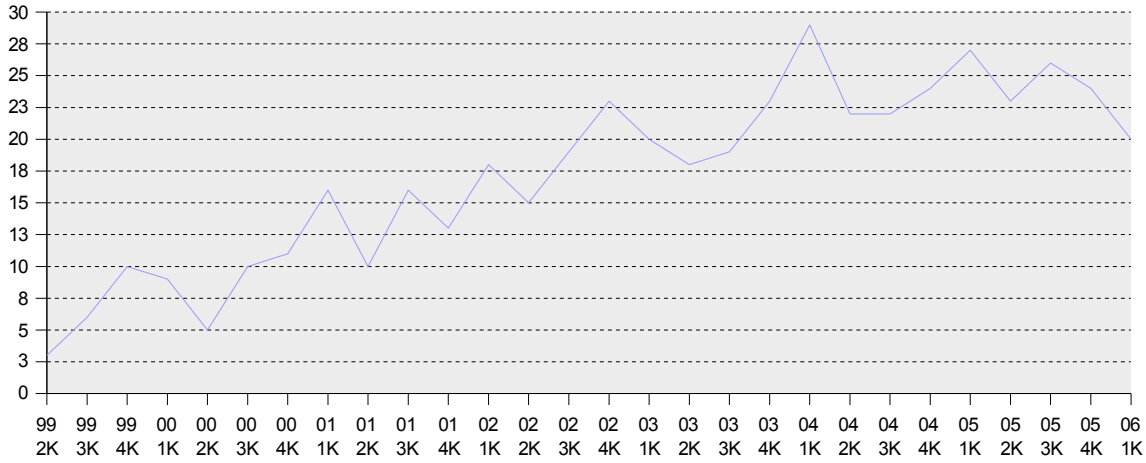
Fig.4: Antall endringer i kodebasen, per kvartal



Vi kan se av figurene at frem til 2002 er det den innerste kretsen, «kjernen», som utfører nesten alle endringene i kodebasen. Selv om antallet aktive utviklere nesten fordobler seg fra 2000 til 2001, tar det altså enda et år før de mer perifere utviklerne begynner å komme mer med. I løpet av 2002 ser vi en ny økning i antallet utviklere som er aktive, og nå ser vi for første gang at «kjernen» ikke lenger er like enerådende i forhold til endringer. Dette bildet holder seg frem til i dag, der de tre til fire mest aktive utviklerne står for omtrent halvparten av endringene i kodebasen. Det er interessant å merke seg at dette mønsteret holder seg nokså stabilt fra og med 2002, uavhengig av hvor mange

utviklere som er aktive til en hver tid. Dette bekrefter bildet av OS-utvikling som bestående av en liten kjerne av utviklere, med en bredere krets (som Raymond (1997) kaller «halo») av deltakere som i hovedsak tester og fikser småting (se også Mockus, Fielding & herbsleb 2000).

Fig.5: Antall utviklere som gjorde endringer i kodebasen, per kvartal



Hele utviklingsperioden, sett under ett, viser en klar økning i antallet aktive utviklere. Dette er interessant, også med tanke på at VideoLAN i svært liten grad har drevet noen form for promotering eller egenreklame. Det har vært en del utskiftninger underveis, som den taggete kurven i figur 5 indikerer, men uten at dette har hatt drastisk innvirkning vis man ser perioden under ett. I siste halvdel av 2004, og midtveis i 2005, ser vi et fall i generell aktivitet i kodebasen. I det siste tilfellet er også den relative innsatsen til «kjernen» noe lavere enn ellers. Dette kan nok tilskrives at to av de mest aktive utviklerne måtte kutte ned på sin deltakelse i henholdsvis 2004 og 2005, på grunn av andre forpliktelser. Fallet i første halvdel av 2006 er nok sannsynligvis grunnet manglende data, siden denne tidsperioden ikke er over enda.

4.3.5 Status

VideoLAN sine hjemmesider rapporterer automatisk hvor mange nedlastinger av VLC som er foretatt siden februar 2005. Denne telleren er nå oppe i over 11 millioner nedlastinger, med et gjennomsnitt på ca to nedlastinger i sekundet. I tillegg til dette tallet kan man legge til et anselig antall nedlastinger fra populære nedlastingssider som tucows.com eller download.com, pluss majoriteten av linuxbrukerne, siden disse som regel vil laste ned sine versjoner gjennom de ulike distribusjonenes oppdateringsverktøy (for eksempel apt-get for Debian og portage for Gentoo). Alt i

alt vitner dette om en imponerende markedsandel med tanke på prosjektets beskjedne finansielle styrke. Det har heller ikke vært ført noen aktiv promotering av produktet, utover sporadisk deltakelse på enkelte messer med linux eller multimedia som tema. At VLC likevel har fått en slik spredning, kun gjennom jungeltelegrafen, er en vitnesbyrd om den tekniske kvaliteten som utviklergruppa har oppnådd.

Den nåværende offisielle versjonen av VLC er 0.8.4. Med andre ord er ikke VLC å regne som et fullført produkt enda, men som en langt fremskredet betaversjon. Med de krav som utviklerne har stilt til VLC så langt, kan man nok forvente mange år med kontinuerlig utvikling før det er sannsynlig at versjon 1.0.0 ligger klar til nedlasting. Det vil i så fall bli en stor milepæl i VLCs historie. Dagens versjon er imidlertid ikke sterkt preget av å være uferdig, hvis man ser på funksjonalitet og stabilitet. Det de fleste brukerne reagerer på er derimot det grafiske grensesnittet, som enkelte finner lite intuitivt, rotete og vanskelig å finne frem i. VideoLAN har også kommentert disse manglene og søker konkret etter nye designere på sine egne rekrutteringssider.

Etter GPL-lisensieringen har VLC opplevd et merkbart skifte i tyngdepunkt når det gjelder utviklere, fra en overvekt av studenter ved Ecole Centrale Paris, til en mer internasjonal utviklergruppe, der studentene er i klart mindretall. Per i dag er det kun to aktive studenter som deltar i prosjektet, hvor begge disse er i sitt siste år. Ingen studenter fra lavere årskull er involvert for øyeblikket. Det er imidlertid noen tidligere studenter som fortsatt deltar i utviklingen, selv om de ikke lenger har noen formell tilknytning til universitetet. Dette har forandret måten gruppa arbeider og kommuniserer på. Det som før ble formidlet gjennom fysiske møter, enten planlagte møter eller tilfeldige sammenkomster, er i dag flyttet til VideoLANs IRC-kanal på freenet, offentlige diskusjonsforum og til dels e-postlistene. Den fysiske arenaen er mer eller mindre borte, til fordel for en virtuell arena.

Det er nå snart syv år siden prosjektet VideoLAN ble igangsatt. Med alderen kommer dessverre også enkelte nye utfordringer til overflaten. Utviklerne opplever stadig oftere å måtte hanskes med gammel kildekode, der de opprinnelige forfatterne ikke lenger er tilgjengelige – eller har svært liten tid til å hjelpe. De siste årene har VLC i tillegg mistet to av sine mest aktive og erfarne utviklerne, som har måttet trappe ned sitt engasjement på grunn av andre forpliktelser. Dette kan bli et problem for VLC, som er avhengige av kontinuerlig innovasjon for å kunne hevde seg blant pengesterke konkurrenter.

4.4 Xiph vs. Videolan

Jeg har i min fremstilling valgt å se på to organisasjoner som begge utvikler programvare med åpen kildekode, og som begge benytter metoder og verktøy typiske for denne stadig voksende tradisjonen innenfor systemutvikling. Det er likevel interessante forskjeller mellom disse to, spesielt med hensyn til hvordan organisasjonene arbeider innad og hvordan de forholder seg til omverdenen.

En åpenbar forskjell ligger i måten de organiserer kommunikasjon på, både internt og eksternt. Hos Xiph har e-postlistene høy prioritet som arena for kommunikasjon mellom utviklerne, der de viktigste bidragsyterne er godt representert. Hos VideoLAN har e-postlistene derimot en mer perifer rolle i utviklingsarbeidet, og fungerer stort sett som en kanal for automatisk annonsering av oppdateringer i Subversjon. I stedet foregår brorparten av utviklernes kommunikasjon fortløpende via IRC. Også i Xiph er IRC en etablert kommunikasjonsarena, men fungerer her først og fremst som en sosial arena og et supplement til e-postlistene, ikke som et substitutt. Et unntak fra dette er Xiphs månedlige fellesmøter, som faktisk blir holdt over IRC. Noe tilsvarende finnes ikke i VideoLAN lengre, etter at den fysiske forankringen til Ecole Centrale Paris ble brutt, der man før hadde ukentlige møter på campus.

De to organisasjonene skiller seg også fra hverandre når det gjelder ekstern kommunikasjon, i forhold til sluttbrukere og andre interesserte. Her har VideoLAN valgt å opprette både egne e-postlister og diskusjonsforum for brukere, der også utviklerne deltar. For brukerne av Vorbis og Theora er det opprettet egne e-postlister, men ingen diskusjonsfora. Siden VideoLAN også har generelt større aktivitet på IRC, er det som regel enklere å få svar på spørsmål og problemer her, enn det som er tilfelle på Xiph sine IRC-kanaler. Dette kommer nok delvis av at VideoLAN har et større utviklingsmiljø, men i tillegg kan man spore en større vilje til å komme nye brukere i møte enn det som er tilfelle for Xiph, der man gjerne forventer et høyere kompetansenivå av de som deltar i kanalene.

For nye utviklere med ambisjoner om å delta i utviklingsarbeidet kan det derimot være enklere å finne ut av hva som har blitt gjort og hva som er behovet i dag hos Xiph enn hos VideoLAN. Siden Xiph er ivrige brukere av både e-postlister og Wiki-er, som blir arkivert og publisert på nett, er det mulig for eksterne å få tilgang til en anselig mengde informasjon om utviklingsprosessen. Xiph har også etablert «dusører», det vil si egne oppføringer på nett, der de mest etterspurte utvikleroppgavene blir listet opp. Med VideoLANs hovedvekt på bruk av IRC som kommunikasjonsarena for utviklerne, blir slik informasjonen svært tidsbestemt, og i liten grad

arkivert på en systematisk måte. Dermed blir nye utviklere mer avhengige av å få direkte kontakt med de etablerte utviklerne for å få svar på spørsmål i tilknytning til kildekoden.

Xiph er altså flinkest til å arkivere det de driver med, mens VideoLAN er generelt mer åpne i forhold til eksterne. Svært forenklet kan man si at Xiph er utviklerorientert, mens VideoLAN er mer brukerorientert. Dette kan kanskje spores tilbake til opprinnelsen til prosjektene. Xiph ble jo dannet som en reaksjon mot proprietære formater, der den moralske og politiske siden av Open Source var et viktig element fra starten av. VideoLAN på sin side, var ikke Open Source i begynnelsen, men hadde et mer pragmatisk utgangspunkt, der funksjonalitet for brukerne (studentene) var den kanskje viktigste drivkraften for utviklerne.

5 Kunnskapsarbeid i Xiph og VideoLAN

I denne delen vil jeg prøve å analysere de observasjoner jeg har gjort vedrørende mine to case, i forhold til hvordan kunnskap blir kommunisert eksternt og internt. I de første kapitlene stiller jeg meg på utsiden og ser etter spor av eksplisitt kunnskap, før jeg trer inn i casene og forsøker å følge den tause kunnskapens bevegelser.

Avslutningsvis vil jeg fokusere på hvordan de nevnte forskjellene mellom casene har skapt behov for ulike endringer i organisasjonene, samt diskutere hvilke muligheter man har for å drive en aktiv kunnskapsledelse i elektroniske praksisnettverk generelt, og i OS-prosjekter spesielt.

5.1 Kontakt med omverden

I beskrivelsen av opprinnelsen til fenomenet Open Source, var vi innom GNU-prosjektet og dets opphav. Her så vi hvordan Richard Stallman dannet Free Software Foundation som en protest mot rådende ledelsesstrategier, der man i følge Stallman var i ferd med å skape kunstige vegger mellom arbeidsfolk. Han var opprørt over et stadig økende fokus på hierarkiske beslutningsstrukturer, der kommunikasjon og samhandling ble kontrollert ovenfra, med lite rom for horisontal, uformell kommunikasjon og kunnskapsutveksling. Denne holdningen har etter hvert blitt et kjennetegn ved OS-bevegelsen, på den måten at man har hatt en iboende uvilje til å sette ned formelle direktiver for hvordan folk skal arbeide sammen og en enda større motvilje mot å innføre begrensninger i hva man kan utveksle av informasjon. Ulempen ved dette kommer i form av en generell nedprioritering

av formelle kanaler for kommunikasjon, som for eksempel offisielle nettsider, prosjektbeskrivelser, manualer og bruksanvisninger. Ved å opphøye informasjon som «offisiell», eller skape manualer som definerer den «riktige» metoden å bruke et verktøy på, er man allerede i ferd med å begrense friheten til den enkelte deltaker i prosjektet, noe som oppleves som negativt av mange OS-utviklere. En rask tur innom sourceforge.net, som er en av de største samlingene av OS-programvare, bekrefter denne tendensen. Her må man ofte lete med lupe for å finne ut hva de ulike programmene er ment å utføre, i tillegg til at navnene i seg selv ofte er svært forkortet[23] og lite informative. Dette mønsteret finner jeg i nokså ulik grad i mine to case, som vi skal se eksempler på i de neste to delkapitlene.

5.1.1 Xiphs ansikt utad

Vi starter med å se nærmere på hvordan Xiph har håndtert offisiell informasjon og kontakt overfor eksterne brukere og andre interesserte. Prosjektbeskrivelsene på Xiph.org er svært brede, og handler mer om intensjonen til utviklerne, enn om hva produktene deres er ment å brukes til. For eksempel så introduseres nye brukere til Vorbis gjennom denne teksten på deres offisielle nettside:

> *«Ogg Vorbis is a fully open, non-proprietary, patent-and-royalty-free, general-purpose compressed audio format» (http://www.vorbis.com)*

Vi ser her at beskrivelsen legger vekt på at Vorbis er et åpent format, mens bruken av det kun betegnes som «general-purpose», som vel egentlig er en annen måte å si «gjør hva du vil med det» på. På e-postlistene til Vorbis oppdaget jeg imidlertid at mange brukere ønsket mer informasjon på nettsidene, og at mange følte terskelen for å installere og bruke Vorbis var unødvendig høy. Etter noen runder med interne diskusjoner rundt dette, ble etter hvert en ny nettside lansert (vorbis.com), der man forsøkte å samle offisiell informasjon om produktet. Oppgaven med å oppdatere denne siden har imidlertid blitt prioritert nokså lavt av de aktive utviklerne i etterkant. Montgomery kommenterer problemet på e-postlistene for Vorbis, der han viser vilje til å gjøre noe med det, men understreker at slike ting tar tid i Xiph:

> *«I didn't realize Vorbis.com had bitrotted to quite this level of uselessness. I'll get it updated because, damn, that's emabarrassing, but it won't quite happen overnight.» (http://lists.xiph.org/pipermail/vorbis/2005-January/025657.html)*

Hvis man ser på hvor mye informasjon som blir produsert gjennom oppdateringer i Subversjon og tekniske diskusjoner på e-postlistene, er den forholdsvise mangelen på offisiell dokumentasjon

[23] Sansynligvis fordi mange av dem blir benyttet gjennom et tekstgrensesnitt.

slående. Faktisk finner man mer oppdatert og utfyllende informasjon om Xiph og Vorbis ved å foreta et søk på Wikipedia, enn det som er tilfelle for organisasjonens egne hjemmesider. Wiki-formatet har forresten også blitt brukt av Xiph for å presentere offisiell informasjon, gjennom opprettelsen av en egen wiki-side for hvert prosjekt. Her har man valgt å definere egne «portvakter», det vil si at enkelte medlemmer av organisasjonen har fått mandat til å passe på hva som havner på disse sidene. Disse sidene har over tid blitt noe utdaterte, og per i dag finner man fortsatt mest informasjon om Xiph ved å gå gjennom de uoffisielle Wikipedia-oppføringene – der alle har lik adgang til å endre det som står skrevet.

Når det gjelder manualer og bruksanvisninger i tilknytning til Vorbis og Theora, vil nok disse virke noe vanskelig tilgjengelig for uinnvidde. Det som gis av informasjon er gjerne knyttet til spesielle problemer og avansert bruk, mer dagligdags bruk er i liten grad beskrevet. På nettsiden Vorbis.com, som er rettet mot nye brukere, er det laget en egen side med overskriften «Get set up». Denne siden gir imidlertid ikke noen trinnvis bruksanvisning, men har i stedet en rekke lenker til ulike løsninger for å kunne bruke Vorbis. Vi kan igjen se en generell motvilje til å definere den «riktige» måten å installere eller benytte programvaren. Fokuset er i stedet lagt på å presentere alle ulike mulighetene som finnes, uten noen form for rangering eller evaluering av disse.

I diskusjonen omkring offisielle informasjonskanaler, vil jeg trekke fram noen episoder som jeg føler gir et ekstra lite innblikk i hvordan Xiph stiller seg i så måte. Hydrogenaudio.org er et uavhengig forum der man diskuterer alle former for multimedia, og der noen av utviklerne i Xiph har vært aktive deltakere. Her foregikk det for noen år siden en diskusjon om hvorvidt OGG Vorbis virkelig var fritt for patenter, som utviklerne hadde hevdet ved flere anledninger. Mange etterspurte en offisiell uttalelse fra Xiph, som aldri kom, muligens på grunn av juridiske årsaker (Xiph var redd for å bli sitert i forbindelse med en eventuell senere rettssak om patenter). I den forbindelse var det en av utviklerne som ordla seg på en interessant måte. Han ville stadfeste at han kun snakket som enkeltperson og at hans meninger ikke måtte forstås som Xiphs offisielle posisjon, og avsluttet med ordene «*if there is such a thing*». Dette vitner med andre ord om en kultur der man ikke har tradisjon for å presentere offisiell informasjon, faktisk til de grader at en intern utvikler bestrider om det i det hele tatt finnes offisiell informasjon fra Xiph. Dette var sannsynligvis en noe flåsete bemerkning, tatt i betraktning at det var en del av en opphetet diskusjon, men det gir likevel en interessant pekepinn på hva som har vært vanlig praksis hos Xiph.

Et annet sitat fra IRC-kanalene til Xiph, bekrefter en noe lunken holdning til offisiell

dokumentasjon. Tema for samtalen var tekniske beskriver av lydalgoritmer, der en av utviklerne karakteriserte denne dokumentasjonen slik:

> *«It's all hacks that worked after people with good ears were able to characterize a problem. All the fancy academic papers just make it look intentional.»*
> *(05/03-2006, #vorbis på freenet)*

Med andre ord legger utvikleren her vekt på det praktiske aspektet («hack's»), og den tause kunnskapen som kreves for å forstå arbeid med lydalgoritmer («people with good ears»), mens de offisielle dokumentene som beskriver funnene i ettertid blir avskrevet som «fancy academic papers». Utvikleren i dette eksempelet tilskriver altså slike dokumenter liten verdi for andre enn de som ønsker å fremstille egne resultater som systematiske og akademiske. Vi ser altså et overhengende praktisk syn på kunnskap blant utviklerne i Xiph, der konkret relevant erfaring vektlegges adskillig mer enn planer, modeller og ideer (jamfør bygningsarbeideren i eksempelet i kap.2.3.1).

Denne nedprioriteringen av prosjektenes offisielle fasade kan vi også finne igjen i måten enkelte utviklere møter eksterne brukere og samarbeidspartnere på. Av og til er ikke åpenheten ensbetydende med en inkluderende samtaleform, noe jeg har observert blant annet hos Xiph.

En episode gikk for seg på et av de månedlige møtene på IRC, der en ekstern samarbeidspartner etterlyste noen finansielle detaljer. Han hadde forsøkt å kontakte Xiph gjennom de offisielle e-postadressene listet på hjemmesiden, men ikke fått svar. I det han nå forsøkte å ta direkte kontakt på møtet, fikk han et kort svar fra en av møtedeltakerne om å sende e-post til de samme adressene. Den eksterne kontakten ville ikke gi seg med dette, men forsøkte flere ganger å få et svar. Diskusjonen ellers i møtet fortsatte ufortrødent, uten at det så ut som noen enset de mer og mer tilspissede kontaktforsøkene. Dette viser at ignorering (som Alan Cox beskriver i kapittel 2.1.5) ikke er et ukjent virkemiddel i OS-sammenheng.

Et annet eksempel observerte jeg i en diskusjon på IRC, der en av de interne utviklerne mente at en nykommer var altfor ordrik, samtidig som han ikke tilførte diskusjonen noe vettugt. Utvikleren avsluttet dermed diskusjonen med ordene: «welcome to my ignorelist. You are worthless, and incredible verbose about it». Denne brutale avbrytelsen av diskusjonen, ble delvis dekt over av en annen utvikler som overtok med en vennligere tone. En tredje utvikler kommenterte tørt at han satt og ventet på at noe slikt skulle skje. Dette viser både at det ikke er uvanlig å bruke ignorering som

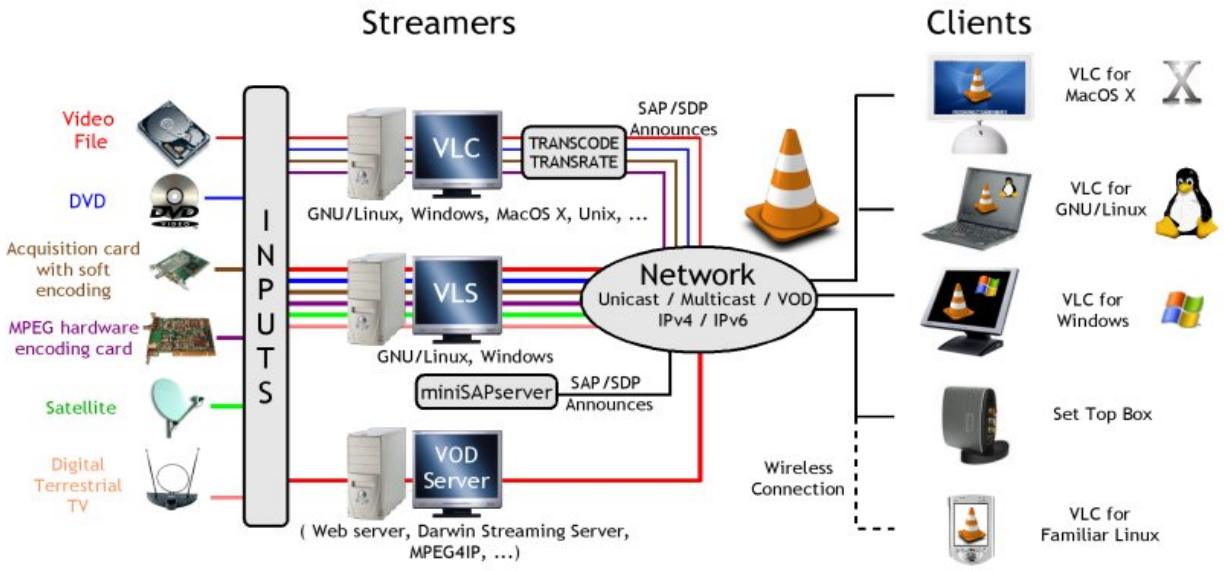
virkemiddel innad i Xiph, samt at enkelte i organisasjonen har nokså lav toleransegrense i forhold til nykommere. Netiketten for Xiphs IRC-kanaler virker altså å være streng i forhold til tomprat og mangel på kompetanse, men tolererer ganske krasse utspill mot nybegynnere.

Vi kan se på denne harde linjen overfor eksterne som en starten på et «joining script», der man er nødt til å beherske både de interne reglene for netikette, samt inneha tilstrekkelig kompetanse for å kunne bli vurdert som en mulig deltaker i fellesskapet (von Krogh, Spaeth & Lakhani 2003).

5.1.2 VideoLANs ansikt utad

Hvis vi nå beveger oss over til VideoLAN, kan vi se lignende holdninger i forhold til offisielle dokumenter og kommunikasjonsformer. Det er likevel interessante forskjeller, med tanke på ekstern og intern kommunikasjon. Hvis vi først ser på offisiell informasjon myntet på eksterne brukere og eventuelle samarbeidspartnere, har VideoLAN tydeligvis klart å mobilisere større ressurser enn hva som er tilfelle for Xiph. Nettsidene videolan.org har ganske omfattende informasjon både av teknisk og organisatorisk art. Det første som møter nye brukere her, er en direktelenke til nedlasting av programvaren, samt en grafisk presentasjon av relevante bruksområder. I tillegg har nettsidene utvidede bruksanvisninger med skjermbilder av de mest brukte funksjonene, samt mer inngående beskrivelser til avanserte brukere. Vi ser imidlertid også her at bruksanvisningene legger vekt på muligheter og alternativer, og at man hele tiden er forsiktig med å være for bombastiske i å utpeke den «riktige» bruksmåten for produktet.

FIGUR 6: Grafisk oversikt over VLCs streamingkapabiliteter, fra videolan.org:



Organisasjonen VideoLAN presenterer på sine nettsider både de sentrale administrative funksjonene, og en liste over alle utviklere, med deres spesialområder. De har også en egen side med offisielle logoer, T-skjorter og lignende markedsorienterte effekter. I forhold til sluttbrukere har VideoLAN opprettet egne e-postlister til formålet, der mange av utviklerne også er medlemmer. Hovedvekten av brukerrelaterte spørsmål ser likevel ut til å foregå på dedikerte diskusjonsfora. Per 14.04.2006 var det 11842 registrerte brukere på disse forumene, som tilsammen hadde postet nesten 60000 innlegg. Tonen på disse forumene varierer sterkt, men er hovedsaklig mer uformell enn det som er tilfelle på e-postlistene. Terskelen for å legge inn innlegg her er forholdsvis lav, spesielt med tanke på at diskusjonsfora er et allment utbredt fenomen, også utenom OS-miljøene, noe som påvirker kommunikasjonen i en mer uformell retning.

Den lave terskelen for deltakelse kombinert med en uformell stil har gjort det nødvendig å moderere enkelte innlegg på VideoLANs forumsider. Dette skjedde blant annet i forbindelse med beskyldninger om virus i en av VLC sine utgivelser. Her ble det lagt inn flere flammende innlegg av en frustrert bruker som mente å ha fått virus av en fil lastet ned fra VideoLANs hjemmesider. VideoLAN benektet dette, og forklarte hvorfor det var svært usannsynlig at dette kunne være tilfelle[24]. Samtidig ble enkelte av disse innleggene fjernet på grunn av ekstrem språkbruk, som kunne bli oppfattet støtende av enkelte. Den frustrerte brukeren ble svært indignert over å få sine innlegg sensurert, men møtte lite støtte fra andre deltakere på forumet, ettersom han i sine innlegg brøt med det meste av nettikette og vanlig høflighet. Her kan man se tendenser til at miljøet selv regulerer omgangsformen på forumene, i tillegg til at utviklere fra VideoLAN er innom og modererer ekstreme innlegg.

På VideoLAN sin IRC-kanal har jeg observert en noe større åpenhet og imøtekommenhet i forhold til nye brukere, enn hva som var tilfelle for Xiph. Det er likevel tydelig at man også her opererer med visse former for nettikette, og at måten man formulerer seg på er essensiell for hva slags respons man får. Spørsmål av typen «Is there anybody here?» eller «Can anybody help me getting my videofile to work?», blir ofte ignorert eller i beste fall besvart med et ønske om å konkretisere spørsmålet. Hvis man derimot spør et konkret spørsmål om en konkret funksjonalitet ved VLC, er sjansene mye større for å få konstruktiv respons, spesielt hvis spørsmålet inneholder interessante tekniske detaljer. Ved å vise at man har tatt seg tid til å bygge opp noe kompetanse på egen hånd, før man stiller et spørsmål, så oppnår man et mye mer velvillig publikum. Dette var tilfelle for både Xiph og VideoLAN, med den forskjellen at terskelen for hva som var verdt å svare på virket

[24]Alle utgivelser av VLC blir kompilert på en linux-plattform, som i svært liten grad er plaget av virus.

generelt høyere hos Xiph. Et annet trekk ved VideoLAN sin IRC-kanal var at tonen var mer uformell, med et større innslag av småprat og personlige referanser. Det ble også vekslet mellom fransk og engelsk, ut i fra hva spørsmålsstilleren behersket best, noe som igjen tyder på ganske løse normer for kommunikasjon i kanalen.

VideoLAN bruker i all hovedsak IRC som arena, både for diskusjoner, beslutninger og generell informasjon. E-postlistene for utviklere består nesten bare av autogenererte meldinger fra Subversion, ved oppdateringer i kodebasen. Det er heller ikke vanlig at utviklerne snakker direkte seg i mellom, med unntak av de mer perifere utviklerne, som av og til benytter privat konversasjon. Ergo fungerer IRC-kanalen her både som en utviklerarena og en brukerarena. En av de mest erfarne utviklerne i prosjektet uttrykte noe skepsis overfor denne praksisen, fordi han mente det gikk ut over informasjonen til nye utviklere. Han ønsket å formalisere mer av kommunikasjonen under utviklingsprosessen, slik at den ikke druknet i strømmen av uformell prat og generell brukerstøtte. Som eksempler på forbedringer trakk denne utvikleren frem økt bruk av e-postlister, for å sikre at utviklingsprosessen blir dokumentert, og opprettelsen av wiki-er, for å gjøre det lettere for alle å se hva de ulike utviklerne driver med for øyeblikket.

5.2 Utveksling av taus kunnskap mellom utviklerne

Vi har så langt sett hvordan mine to case presenterer seg utad, hvilke holdninger og praksis de har i forhold til offisiell informasjon. Begge er preget av en skepsis til å formalisere informasjon, i den forstand at de er redde for å begrense sin egen handlefrihet ved å definere seg selv for smalt. Dette gir seg utslag i en nedprioritering av offisiell informasjon, der man er mer opptatt av å beskrive muligheter enn å definere grenser for eget prosjekt. Denne holdningen minner mye om Turnbull (1993) og hans fokus på arbeidsplassen som laboratorium, med tilstrekkelig rom for eksperimentering. Ny kunnskap som akkumuleres på denne måten, via eksperimentering og åpenhet i arbeidssituasjonen, fester seg ofte som *taus* kunnskap jamfør Orr (1996). Denne kunnskapen spres så mellom utviklerne gjennom ulike måter å kommunisere på internt i organisasjonen. Jeg vil i dette kapittelet vise hvordan mine to case overfører taus kunnskap enten via spontane fortellinger eller via konkrete og detaljerte eksempler.

5.2.1 Fortellingenes rolle

VideoLAN ble som tidligere nevnt startet opp ved Ecole Centrale Paris, og hadde derfor fra starten av en utviklergruppe som var fysisk samlokalisert og som arbeidet nokså tett sammen. Utviklerne

hadde jevnlige møter, samt en fysisk møteplass der de kunne utvikle og teste ulike oppsett i fellesskap. I tillegg utgjorde skolen også en sosial arena i form av kantiner, pauserom og faglige eller utenomfaglige arrangementer. Dermed hadde VideoLAN på et tidlig stadium flere arenaer å møtes tilfeldig på, der spontane samtaler og diskusjoner kunne oppstå. Nettopp i en slik situasjon, med stort rom for uformelle kommunikasjon og spontanitet, er det gode vilkår for at fortellinger utveksles mellom utviklerne. Derfor forventet jeg å finne et større utslag av denne typen for kunnskapsutveksling hos VideoLAN enn hos Xiph, der de fleste prosjektene hadde sitt opphav i enkeltpersoners uavhengige initiativ. Selv om tyngden av utviklere i VideoLAN i dag ikke lenger har sitt utspring fra skolen, kan det se ut som mye av den uformelle og direkte tonen fra det lokale miljøet har blitt overført til IRC-kanalen på freenet, som nevnt i forrige kapittel. Det er også her jeg finner flest eksempler på fortellinger mellom utviklerne.

Diskusjonsforumene på videolan.org ble også undersøkt, og hadde mange historier fra brukerne om problemer de ønsket en løsning på. Disse er likevel ikke interessante i forhold til Orrs definisjon på en fortelling, siden han fremhever *løsningen* som er en viktig del av det som fortelles. I tillegg kommer historiene fra brukere utenfra, og dermed har de ikke den fortettede og effektive formen som man forventer av fortellinger mellom utviklere som kjenner hverandre og er del av samme fellesskap. Mengden innlegg på forumene, samt den lave terskelen for deltakelse, gjør at det i liten grad finnes den type fortellinger jeg leter etter. E-postlistene til VideoLAN er delt mellom brukerlister og en utviklerlister. Brukerlistene har mange likheter med forumene, og virker derfor ikke å være en god arena for fortellinger. Utviklerlistene til VideoLAN er, som nevnt tidligere, nesten ikke i bruk til annet enn automatiske meldinger fra Subversjon. Ergo blir IRC-loggene den primær kilden for fortellinger mellom utviklere i VideoLAN.

Fortellinger krever en gjensidig tillit mellom forteller og mottaker. Med dette mener jeg at historien bør bli forstått nokså intuitivt, uten at fortelleren skal være nødt til å forklare alle de implisitte kulturelle referansene i historien. Derfor observerte jeg fortellinger først og fremst som spontane innlegg i en allerede etablert samtale mellom to eller flere utviklere på IRC, der deltakerne virket trygge på hverandres kompetansenivå, eller med andre ord følte seg noenlunde som likemenn. Nykommere som stilte spørsmål av nokså grunnleggende art, satte sjelden i gang noen utveksling av historier. Derimot var det ved ekstra vanskelige problemer, eller ved spørsmål som krevde betydelig skjønn, at jeg observerte flest spontane fortellinger. Et typisk eksempel er fortellinger som brukes i feilsøking, der utvikleren gjerne har oversett en subtil detalj som skaper vanskeligheter. En slik fortelling dukket opp på IRC i forbindelse med at en utvikler ønsket hjelp til å konfigurere et litt

uvanlig nettverksoppsett. Etter å ha diskutert ulike løsninger frem og tilbake, spør en av utviklerne om bruken av en konkret metode, noe som setter i gang følgende ordveksling mellom to andre utviklere:

> «
> *Utvikler3: we did try to use source NAT, but SNAT happens after routing decision.*
> *Utvikler3: I remember Utvikler7 had a lot of trouble load-balancing two DSL-connections because of this.*
> *Utvikler9: hm I see*
> *Utvikler3: basically, we could not control the output interface on the basis of the natted-source*
> *Utvikler3: and then the provider would kill half of our packets because of its ingress filters*
> *Utvikler9: right indeed*
> *Utvikler3: which had a disastrous effect on TCP connections you can imagine*
> *Utvikler9: so marking and iproute2 is what is needed here*
> »
> *(31/3-2006 #videolan på freenet)*

I første setning her gir Utvikler1 årsaken til hvorfor SNAT ikke er egnet til å løse det aktuelle problemet. Han avslutter imidlertid ikke med dette, men velger å utbrodere med en kort fortelling som illustrerer hvordan han selv kom fram til denne kunnskapen. Denne fortellingen er svært fortettet, ved at den inneholder mange begreper og uttrykk som krever en viss kjennskap til nettverksteknologi, samt erfaring fra lignende situasjoner, for å kunne forstås fullt ut. Vi ser derfor at Utvikler2 hele veien understreker at han forstår fortellingens relevans til det aktuelle problemet, ved hjelp av korte bekreftelser. Utvikler1 benytter også uttrykket «you can imagine» for å formidle en felles forståelse av problemet. Utvikler2 går dermed direkte over i å diskutere andre løsninger på problemet, siden den første løsningen raskt ble vurdert og forkastet ved hjelp av Utvikler1s komprimerte fortelling. Vi ser har at fortellinger kan fungere som et effektiv verktøy i problemløsningen, samtidig som de gir ekstra kunnskaper om emnet til diskusjon, for alle som har nok faglig og kulturell kompetanse til å følge fremstillingen. Lærdommen fra slike fortellinger vil ofte være vanskelig å uttrykke i form av en manual eller en bruksanvisning, men vil i stedet ligge latent i form av taus kunnskap utviklerne kan hente frem fra hukommelsen hvis lignende situasjoner skulle oppstå senere.

Fortellinger trenger ikke alltid henge sammen med problemløsning. Det følgende eksempelet kom etter en diskusjon omkring nettverksprotokoller og videooverføring. En av utviklerne på listen var i ferd med å forklare et gammelt nettverkstriks, og hvordan det oppsto. Halvveis i hans fortelling, bryter en annen utvikler inn med følgende:

«Utvikler8: I have a nice story on that. When a user from a certain wireless network watched a stream from our win2003 server it crashed our server»
(09/04.2006 #videolan på freenet)

Denne utvikleren fortsetter deretter med flere tekniske detaljer som forklarer hvorfor dette skjedde, selv om det bare er delvis relatert til diskusjonen som pågår. Den første utvikleren fortsetter så med sin fortelling, uten å kommentere den andres. Vi ser altså at fortellingene inspirerer til nye fortellinger, og at disse tar naturlig del i samtalen. Den uformelle, spontane formen på samtalene over IRC virker altså som en god katalysator for utvekslingen av fortellinger.

I tilfellet over hjelper fortellingene i liten grad til å løse det aktuelle problemet som diskuteres, men i stedet har de andre like viktige funksjoner. For det første gir fortellingene uttrykk for fortellerens kompetanse og erfaring, slik at denne styrker sin status og tilhørighet i fellesskapet. For det andre kan detaljene i fortellingene inneholde kunnskap som er nyttig for andre i kanalen, eller inspirere til nye fortellinger som kan inneholde ytterligere biter med kunnskap og erfaring. For det tredje kan fortellingene virke styrkende på fellesskapet som helhet, ved at man får formidlet typiske situasjoner og følelser, som kan være vanskelig å få forståelse for hos andre enn utviklere på tilsvarende kompetansenivå. Dette siste poenget ser jeg også i form av små henvisninger eller referanser til typiske, gjenkjennbare situasjoner. Dette kan gjerne komme i en litt humoristisk form, ofte som galgenhumor, der utvikleren rekonstruerer følelser som irritasjon, avmakt, fortvilelse overfor maskinenes uregjerlighet. Her er et eksempel på en slik digresjon:

«
Utvikler3: I just love when automake rebuilds the whole aclocal...make thing because someone changed a comment in configure.ac
Utvikler3: ...at work
Utvikler3: I can't stand it at home
Utvikler6: lol[25]
Utvikler6: i thought u studied (...)
»
(31/3-2006 #videolan på freenet)

Utvikler1 lufter sin frustrasjon her ved hjelp av en tørrvittig framstilling, noe som setter i gang en uformell samtale, der mer personlige tema kan tas opp uanstrengt. Den humoristiske innfallsvinkelen er med på å ufarliggjøre problemer eller irritasjonsmomenter for utvikleren, samtidig som han oppretter bånd til fellesskapet, gjennom gjenkjennelsen av situasjonene - og følelsen av å være «i samme båt». En lignende type kommentar dukker opp i det følgende

[25]Lol = «laughing out loud»

eksempelet, der en person nettopp har spurt om råd i forbindelse med kompilering av kode. Det første svaret han får er dette:

«I allways cross my fingers and hope i don't run into those unreferenced errors»

Dette svaret har liten eller ingen verdi for den som spør, men virker mer myntet på resten av utviklerne i kanalen, som et hint til en kjent og svært uønsket situasjon mange av utviklerne har vært oppe i. Vi ser at disse korte fortellingene innehar en viktig sosial funksjon, som internt lim i fellesskapet. Ergo utgjør slike fortellinger en viktig del av VideoLANs «joining script» for potensielle nye medlemmer i organisasjonen, i den forstand at man gjennom fortellinger kan demonstrere både faglig og sosial kompetanse, som begge er viktige komponenter i det daglige arbeidet ved VideoLAN.

Den uformelle omgangsformen på IRC gir også rom for en del metainformasjon om koden som jeg ikke finner i noen særlig graf hos Xiph. Dette oppstår gjerne etter et nytt innlegg i kodebasen, der andre utviklere setter i gang en samtale omkring den nye kodens kvalitet og intensjon, og hvordan koden forholder seg til andre interne eller eksterne moduler. Dette er med på å styrke samhandling og koordinasjon mellom utviklerne i deres daglige arbeid.

Hvis vi nå går over til Xiph, der jeg har fokusert på prosjektene Vorbis og Theora, finner jeg i liten grad den samme bruken av spontane fortellinger. IRC-kanalene er forholdsvis lite brukt, og e-postlistene ser ikke ut til i samme grad å inspirere til spontane historier, sannsynligvis på grunn av den noe høyere tekniske og normative terskelen for bidra med innlegg. Det finnes enkelte referanser til kjente problemer og interne spøker, men det er langt fra et like iøynefallende fenomen som på IRC-kanalen til VideoLAN. Dette understøtter tanken om at fysisk samlokalisering, med tilgang til uformelle arenaer for kommunikasjon, er viktig for å etablere en tradisjon for utveksling av spontane fortellinger.

5.2.2 Konkrete eksempler som kunnskapsbærere

Hvis vi retter blikket mot den interne kommunikasjonen i Xiph, finner vi her helt andre mekanismer for overføring av taus kunnskap, enn de uformelle fortellingene vi så hos VideoLAN. E-postlistene og de månedlige møtene på IRC, holder begge en forholdsvis seriøs stil. Det finnes så godt som ingen innlegg med en usaklig tone, og svært få av innleggene har noen utpreget sosial funksjon. Over hele linjen diskuteres det løsninger på ulike problemer, enten vedrørende utviklingen direkte,

eller knyttet til avansert bruk av programvaren. Kommunikasjonen har formelle trekk, i den forstand at det er klare uskrevne kjøreregler for hvordan man ter seg på listene. For eksempel så viser den samme personen som opptrådte så bryskt på IRC[26], større respekt og ydmykhet for sine samtalepartnere når han opererer på e-postlistene. Det faktum at man må være medlem av en liste for å kunne poste direkte til den, hjelper nok på å holde seriøsiteten og kvaliteten oppe (innlegg fra ikke-medlemmer blir vurdert av en administrator før de eventuelt legges inn).

I kapittel 2.3.3 ble bruken av eksempler og maler trukket fram som en direkte og effektiv metode for utveksling av kunnskap. Ved å se hvordan andre har løst beslektede problemer, kan en utvikler få ny innsikt og nye innfallsvinkler til sitt aktuelle problem. Av og til er eksemplene konstruert for anledningen, enten som ferdig kode, pseudokode, eller bare en beskrivelse av framgangsmåte og konsept for løsning. Detaljrikdommen i svarene er ofte proporsjonale med spørsmålene, det vil si at jo mer informasjon om problemet, jo mer utfyllende løsningsforslag. Denne korte korrespondansen fra e-postlisten til Vorbis viser dette i praksis: («>» indikerer tidligere innlegg)

```
> Hi,
>
> I've got alot of long 1-2hr files which I'd like to split up into 5-10 minute
> chunks.  I think this should be possible using a small shell script and vcut,
> but my scripting abilities are lacking.  Does anyone know of the existence of
> a script which would do this, or know how I'd go about making one?
>
> So say I had a 60 min file (60mins.ogg) I'd like to issue a command something
> like
>
>> split outfilename 60mins.ogg 600
>
> which would do a series of vcuts
>
> vcut outfile1.ogg tmpfile.ogg 60mins.ogg +600
> vcut outfile2.ogg tmpfile2.ogg tmpfile.ogg +600
> /bin/rm tmpfile.ogg
> vcut outfile3.ogg tmpfile3.ogg tmpfile2.ogg +600
> /bin/rm tmpfile2.ogg
>
> .......
>
>
> until it's reached the end of the file
>

This does it:

-------------------

#!/bin/bash
split=$1;shift
file="$*"
name="${file%.ogg}" # strip extension.
cp "$file" /tmp/file-$$.ogg
stop=0
```

[26]Se kapittel 5.1.1

```
n=0
while [ $stop -eq 0 ] ; do
    echo "cutting to ${name}-$n.ogg\"
    vcut /tmp/file-$$.ogg "${name}-$n.ogg" /tmp/remainder-$$.ogg "$split"
    stop=$?
    mv /tmp/remainder-$$.ogg /tmp/file-$$.ogg
    n=$(( $n + 1 ))
    done

rm /tmp/file-$$.ogg

--------------------

save it as vdice, then do vdice +600 filename.ogg

The key bit is the 'stop=$?' line, which tells the script to stop cutting
when vcut returns an error (because the split point has gone beyond the
end of the file.)
```

(http://lists.xiph.org/pipermail/vorbis/2005-January/025594.html)

Vi ser i dette eksempelet at løsningen er direkte relatert til problemet som skal løses. Det gis et kortfattet svar, som inneholder den relevante biten med kode som trengs. Det eneste av ekstra informasjon som legges ved, er en kort setning om hva som er den viktigste og minst intuitive delen av skriptet. Resten av den kompetansen som kreves for å få dette til å fungere, regnes det med at spørsmålsstilleren besitter, ettersom han viste tilstrekkelige kunnskaper gjennom sin konkrete og informative formulering av spørsmålet.

I mange tilfeller er hverken spørsmål eller svar så konkret som i forrige eksempel. Det vil ofte kunne ta flere meldinger, med ulike innspill, før et konkret forslag til løsning blir presentert. I eksempelet under er det en uvikler som ønsker å hente ut metainformasjon om lydfiler som er lagret digitalt ved hjelp av i OGG Vorbis. Han etterspør om det finnes ferdige verktøy for dette formålet, eller om han selv skal gå i gang med å skrive kode:

```
>> I was thinking of writing something myself to extract the
>> trackname/etc but I thought it best to check with everyone to see
>> if it hasn't already been done.
>
> A standalone app for that purpose? Not to my knowledge.

"vorbiscomment" from the vorbis-tools, possibly combined with some scripting
depending on purpose.

vorbiscomment foo.ogg | sed -n /^title=/s/.*=//p

would for example display a track's title.
```

(http://lists.xiph.org/pipermail/vorbis/2005-March/025729.html)

Etter et negativt svar, kommer det et forslag, der en modul kalt «vorbiscomments» blir foreslått. Den siste utvikleren navngir altså en mulig kandidat for løsning, og beskriver med et konkret eksempel hvordan den kan benyttes til det formålet som etterspørres. Her får ikke den første utvikleren levert hele den ferdige koden, slik som i det forrige eksempelet, men han får et deleksempel, som kan gi ham et utgangspunkt for selv å sette sammen den koden som trengs. Dette harmonerer med måten spørsmålet blir stilt på, siden spørsmålsstilleren gir uttrykk for at han innehar ferdigheter innen programmering. Slike kortfattede eksempler opptrer ofte på e-postlistene til Vorbis og Theora.

Det er også interessant å merke seg at mange av innleggene på disse listene handler nettopp om å finne riktig verktøy til å løse et generelt problem. Dette illustrerer OS-miljøenes fokus på gjenbruk av kode, og at de fleste OS-utviklere så langt det er mulig vil unngå å «gjenoppfinne hjulet», men heller starte sitt arbeid på hverandres skuldre, jamfør Raymonds tese nummer 2; «Good programmers know what to write. Great ones know what to rewrite (and reuse)» (Raymond 1997).

I bruken av konkrete eksempler observerer jeg at mange utviklere velger å trekke fram kode som de selv har benyttet før. Denne koden trenger ikke være helt kompatibel med det som blir etterspurt, men er gjerne en løsning på et lignende og relatert problem, som kan gi inspirasjon til å løse det aktuelle problemet. Følgende eksempel illustrerer dette:

```
>>>Could you perhaps recommend a "best" strategy for changing my ogg/vorbis
>>>256 kbps files into (say) 128 kbps files? I am capable of hacking scripts,
>>>and such, I am more looking for recommendation of the best tool: does sox
>>>produce good results, would you rather recommend ogg123 to wav, then
>>>oggenc, perhaps something entirely different?
>>
>>I'd probably use something like these two lines in a loop:
>>
>>oggdec -o - old.ogg | oggenc -o new.ogg -q 0 -
>>vorbiscomment -R old.ogg | vorbiscomment -Rw new.ogg
>
>
> Thanks a bunch.

I spent a couple days this Christmas break writing a Perl script to reencode my
-q7 oggs to -q3 to work with the iRiver. I added in a vorbisgain step, and I
apply the replaygain setting to the intermediate wav, since the iRiver player
doesn't pay attention to REPLAYGAIN tags.

It works on cygwin, though it wouldn't be hard to get working on Linux.
Yell if you're interested.
A peeler sure would be nice, though.
```

(http://lists.xiph.org/pipermail/vorbis/2005-January/025631.html)

Her spørres det først etter en strategi for å konvertere mange lydfiler fra en kvalitet over til en annen. Et konkret forslag, med eksempelkode, gis først. Deretter svarer en tredje person med henvisning til et skript som han selv har laget, og tilbyr seg å dele dette med spørsmålsstilleren. I akkurat dette tilfellet ble ikke selve koden lagt med i innlegget, sansynligvis på grunn av kodens omfang. Skriptet som han henviser til ble laget for et lignende formål, men med litt andre hensikter og på en annen (men beslektet) plattform. Her er det med andre ord en underliggende forståelse av verdien i å ha et konkret eksempel som utgangspunkt når man begynner å skrive kode, i forhold til å begynne fra grunnen av på egen hånd.

For VideoLAN sitt vedkommende har jeg ikke funnet like utstrakt bruk av konkrete eksempler og kodesnutter i utviklernes interne kommunikasjon. Dette kan henge sammen med at VideoLAN benytter IRC som primær kommunikasjonskanal, der innleggene typisk er svært korte – fra en til to linjer (Paolillo 1999). Det blir derfor lite rom for å utveksle mer omfattende kodeeksempler av den typen som dukker opp jevnlig på e-postlistene til Xiph. Av og til brukes det lenker til lengre tekster, i form av en «pastebin», det vil si en nettside der man kan legge opp større tekstbiter og opprette en lenke til dem etterpå. Jeg så imidlertid ikke at disse ble brukt for å utveksle kodeeksempler, men stort sett for å legge ut logg fra VLC ved alvorlig feil.

5.2.3 Eksternalisering av taus kunnskap gjennom ulike kanaler

Vi har observert at mine to case har noe ulik praksis i forhold til kommunikasjon og kunnskapsoverføring. Når det gjelder ekstern informasjon, har VideoLAN fokusert på å samle offisiell dokumentasjon på sine nettsider. I tillegg har de opprettet egne diskusjonsforum, der både brukerne og utviklerne opererer. Dessverre har antallet brukere gjort det vanskelig å holde en god balanse mellom spørsmål og svar på disse forumene, noe som svekker nytteverdien deres som kunnskapsbærere. Intern kommunikasjon hos VideoLAN foregår nesten utelukkende på IRC, der man også finner brukerstøtte og generell småprat. Informasjonen som utveksles her blir imidlertid knapp og tildels utilgjengelig for utenforstående, samtidig som man i liten grad får en systematisk arkivering av prosessen. Xiph har på sin side gjort hovedvekten av sin interne informasjonsflyt permanent tilgjengelig for allmennheten, via e-postlistenes arkiver. Her er også bruken av e-postlister foretrukket mellom utviklerne, fremfor for eksempel IRC, noe som gir en noe tregere respons, men som sikrer en høyere kvalitet og presisjon i innleggene, gjennom strenge normer for oppførsel og høy terskel for deltakelse (det vil si et mer selektivt «joining script»). Arkivene kan dermed karakteriseres som en voksende og åpent tilgjengelig kunnskapsbank. Ved hjelp av

intelligente søk på en av de store søkemotorene på internett, kan hvem som helst finne biter av informasjon relatert til deres aktuelle problemer. Denne praksisen utgjør et eksempel på hvordan e-post kan brukes i kunnskapsarbeid, jamfør blant andre Bontis med flere (Bontis, Fearon & Histon 2003).

Nå er det ikke gitt at informasjonen på e-postarkivene vil være forståelig for andre enn de involverte i prosjektet. Dette vil avhenge av den reelle bruken av listene, i forhold til innleggenes form og innhold. Hvis flest mulig skal kunne benytte den kunnskapen som ligger latent i e-postarkivene, må den gjøres mer generell og allmenngyldig. Dette skjer for eksempel ved at man i tillegg til selve koden også tar seg tid til å beskrive hvordan koden fungerer, og hva som er hensikten med den. Hvis vi tar for oss det første eksempelet fra forrige kapittel, så ser vi at innlegget i tillegg til å vise den aktuelle koden, også har en liten beskrivelse av den viktigste funksjonaliteten:

```
The key bit is the 'stop=$?' line, which tells the script to stop cutting
when vcut returns an error (because the split point has gone beyond the end of the
file.)
```

(http://lists.xiph.org/pipermail/vorbis/2005-January/025594.html)

Denne korte teksten beskriver en programmeringsteknisk metode, som en ekstern utvikler vil kunne gjenkjenne på generelt grunnlag. Dermed kan en utvikler bruke dette eksempelet som en *mal* for lignende programmeringsoppgaver, selv om disse ikke har noe direkte med Vorbis å gjøre. Den tause kunnskapen som lå i det konkrete eksempelet blir altså her opphøyet til en mer eksplisitt og allmenngyldig form, og øker dermed sin verdi for *både* eksterne og interne utviklere.

Prosessen med å omdanne taus kunnskap til eksplisitt kunnskap kalles eksternalisering (Nonaka og Takeuchi 1998). Når VideoLAN lager offisiell dokumentasjon og bruksanvisninger er dette en måte å eksternalisere kunnskap på, siden man blir nødt til å konkretisere og sette ord på det som er vanlig praksis. Her er det interessant å merke seg at hoveddelen av denne eksternaliseringen har blitt foretatt av bidragsytere som ikke har deltatt direkte i utviklingen av VLC (Basset 2005). Den store aktive brukerbasen til VLC har tiltrukket seg mange som ønsker å bidra, men som ikke har den nødvendige programmeringskompetansen. Disse har i stedet kunnet bidra med sine brukererfaringer, og produsert omfattende dokumentasjon og bruksanvisninger til VideoLANs hjemmesider.

For Xiph sitt vedkommende er denne formen for eksternalisering nedprioritert eller de mangler de

rette ressursene til å gjennomføre det. I stedet skjer denne prosessen først og fremst gjennom den pågående korrespondansen på e-postlistene, der konkrete eksempler brukes som maler av andre utviklere. Måten man organiserer intern og ekstern kommunikasjon på, resulterer altså i ulike former for eksternalisering av kunnskap. Tabell 1 oppsummerer noen av hovedtrekkene vi har etablert så langt vedrørende kunnskapsarbeid i mine to case:

TABELL 1: Kanaler for kunnskapsarbeid

CASE	*VideoLAN*	*Xiph*
ekstern kommunikasjon	- Offisielle nettsider - IRC - Diskusjonsfora - Wiki	- Offisielle nettsider - Månedlige møter på IRC - Wiki
intern kommunikasjon	- IRC	- E-postlister
taus -> taus kunnskap	- Fortellinger	- Konkrete eksempler
taus -> eksplisitt kunnskap (eksternalisering)	- Offisiell dokumentasjon - Bruksanvisninger (how-to) - Wiki	- Strukturerte e-postarkiver - Noe offisiell dokumentasjon - Wiki

Det er interessant å merke seg at begge case har i økende grad begynt å bruke wiki-er, og flere av utviklerne uttrykker ønske om å bruke denne kanalen ytterligere. Dette kan henge sammen med at wiki-er oppleves som en god måte å eksternalisere kunnskap på. En wiki blir ikke så statisk og begrensende som en tradisjonell offisiell nettside, men heller ikke så flyktig som en diskusjon på IRC. I tillegg har en Wiki sansynligvis større potensiale til å nå ut til eksterne enn det en tradisjonell e-postliste har.

For både Xiph og VideoLAN har det vært en stor hjelp i det arbeidet som er lagt ned av standardorganisasjoner som IETF og W3C, spesielt med hensyn til funksjonalitet som innebærer bruk av internett. Den kunnskapen som er eksternalisert i form av beskrivelser av standarder for nettverkskommunikasjon, gjør at begge prosjektene har et sett med kjøreregler for nettverksprogrammering, som det ellers ville tatt mye prøving og feiling å kommet frem til på egen hånd. Når det gjelder selve behandlingen av lyd og bilde, er det fortsatt MPEG-gruppen som har lagt ned mest arbeid i å utarbeide standarder. Ingen av de mer OS-vennlige standardorganisasjonene har klart å etablere noe tilsvarende, så Xiph og VideoLAN må her gjøre mer av grunnarbeidet selv.

5.3 Balansen mellom anarki og kontroll

Både VideoLAN og Xiph lar seg beskrive som elektroniske praksisnettverk, der taus og eksplisitt kunnskap blir spredt innad i organisasjonen, via ulike kanaler for kunnskapsarbeid. Vi har videre identifisert enkelte forskjeller mellom disse to casene, både når det gjelder valg av kommunikasjonskanaler og i graden av struktur og formalitet i kommunikasjonen. Jeg vil i dette kapittelet forsøke å systematisere disse forskjellene ved å benytte Holck & Jørgensens (2005) konsept om anarki og kontroll, og mestringen av balansegangen mellom disse motpolene (se kapittel 2.1.6).

5.3.1 VideoLAN: Arven fra praksisfelleskapet

VideoLAN har vist seg som et svært åpent og inkluderende prosjekt, der terskelen for deltakelse i de mest relevante kommunikasjonskanalene er lav. Dette, kombinert med at VLC har blitt et solid produkt, er viktige årsaker til hvorfor VideoLAN har klart å etablere et bredt EnoP rundt VLC. Hvis vi henter fram Frost & Shoen (2004) sin oppskrift på et fungerende praksisfelleskap, kan vi si at VideoLAN har fokusert på de to første kriteriene: (1) *Organisering og støtte for aktiviteter i fellesskap* og (2) *Opprette relasjoner mellom mennesker og kunnskapen de besitter* (kap.2.1.5). Den fysiske og organisatoriske tilknytningen til Ecoles Centrale Paris har hatt en sentral rolle i forhold til å legge grunnlaget for disse to kriteriene.

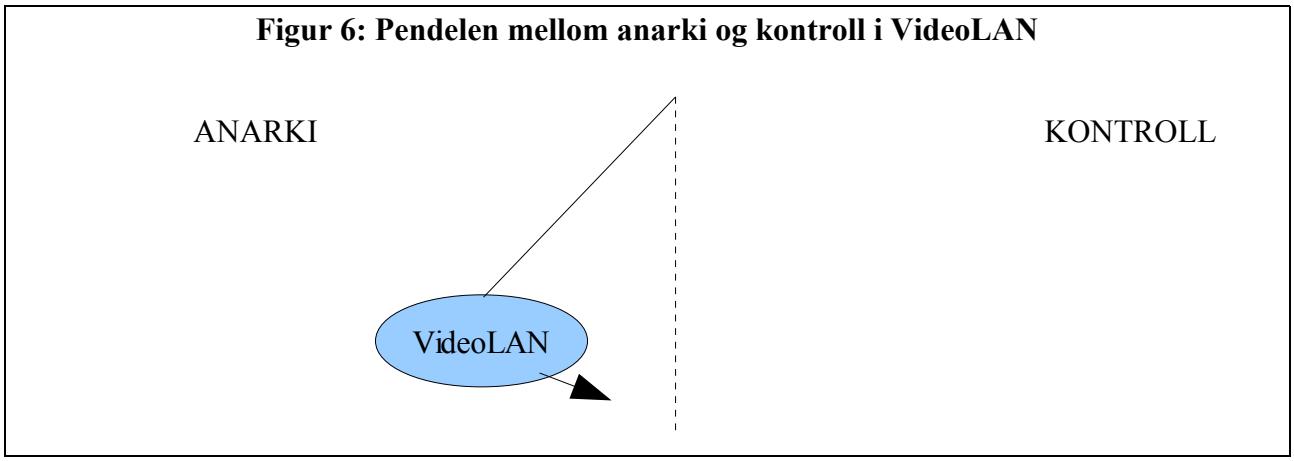
Prosjektet har imidlertid ikke etablert formelle prosedyrer for beslutningstakning eller kodekontroll, noe som enkelte innad i organisasjonen har sett på som et problem. «We allways have trouble releasing», som en av utviklerne uttrykte det. Dette mønsteret kommer også til uttrykk i forbindelse med bruken av IRC som både utvikler- og brukerarena. Her blir informasjonen uformell og «flyktig» i den forstand at den i liten grad egner seg som dokumentasjon av utviklingsprosessen.

Hvis vi tar for utgangspunkt i konseptet om en balansegang mellom anarki og kontroll, vil jeg plassere VideoLAN ganske langt mot anarkiet. Dette kan nok henge sammen med VideoLANs historie som et fysisk forankret prosjekt, noe som har hatt en viktig rolle i å forme måten utviklingen og samhandlingen foregår på. Behovet for koordinering og formalisering av kommunikasjonen har ikke blitt fremtredende, ettersom man i stedet har hatt spontane og uformelle gruppeprosesser gjennom det lokale praksisfelleskapet som ble dannet ved Ecoles Centrale Paris. I tillegg kan det argumenteres med at tryggheten ved å ha en akademisk institusjon i ryggen har gjort kontrollhensynene mindre fremtredende. Prosjektet har ikke hatt noen klar «agenda», men heller

vokst fram som et resultat av mange studenters ulike preferanser og interessefelt, noe som kommer til uttrykk gjennom det store antallet moduler og den utvidede funksjonaliteten i VLC.

Med GPL-lisenseringen og utvidelsen av VideoLAN til å inkludere flere eksterne utviklere, har behovene for koordinering og kontroll i større grad meldt seg. Samtidig har mangelen på prosessdokumentasjon blitt mer fremtredende ettersom prosjektet har blitt eldre og utskiftingen av medlemmer har blitt mer merkbar. Dette gjør seg blant annet utslag i et økt fokus på bruken av prosjekthåndteringsverktøy som TRAC, og eksternalisering av taus kunnskap via Wiki-er og annen offisiell dokumentasjon. Det er derfor ting som tyder på at pendelen nå er i ferd med å svinge mer mot kontroll (illustrert i figur 6), for å følge Holck & Jørgensens (2005) terminologi.

Figur 6: Pendelen mellom anarki og kontroll i VideoLAN



5.3.2 Xiph: Et målrettet ENoP

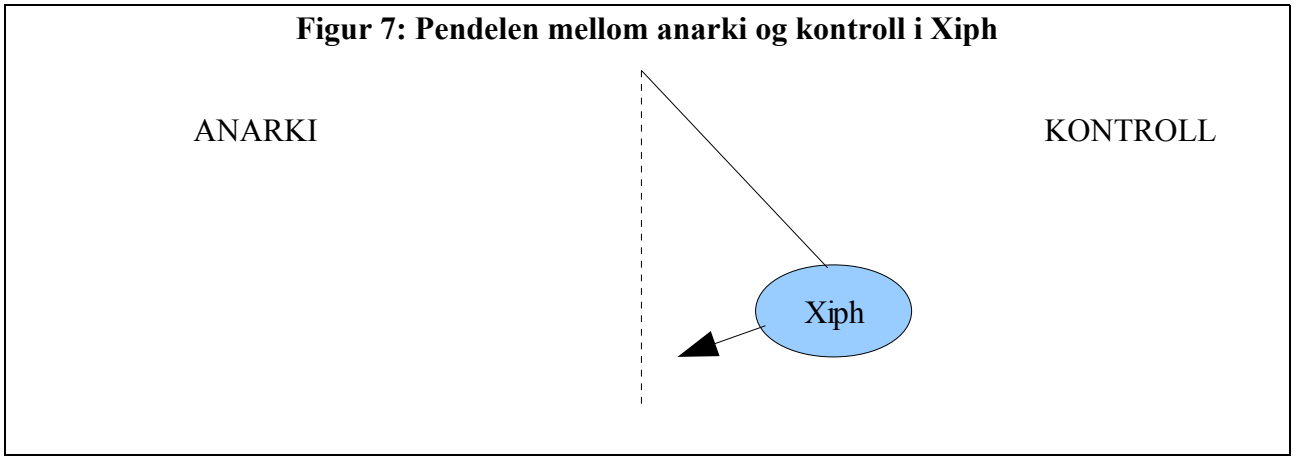
Utgangspunktet for opprettelsen av Xiph, og mer spesifikt prosjektene Vorbis og senere Theora, var klart forskjellig fra VideoLAN. Xiph ble lansert som en reaksjon mot eksisterende kommersielle multimedia-kodeker og deres restriktive lisenseringspolitikk. Allerede fra starten av hadde Xiph en klar og uttalt visjon om å etablere seg som et alternativ for de som ønsket å beholde den samme åpenheten og friheten innenfor multimedia, som det man allerede hadde i det tekstlige domenet på internett. Hvis vi igjen henter fram Frost & Shoen (2004) sine kriterier for et fungerende praksisfelleskap, gjenkjenner vi punkt (3) *å finne et felles fokus,* nokså umiddelbart her. Ved å definere klare visjoner og mål for deres virksomhet har Xiph klart å samle flere ulike prosjekter under samme paraply, som samlet utgjør en helhetlig multimedia-plattform, med liten grad av funksjonell overlapping mellom prosjektene.

Det kan imidlertid virke som Xiph har betalt en pris for sin målrettethet. Fra flere hold har

organisasjonen blitt kritisert for å være lite åpen og inkluderende overfor såvel brukere som potensielle samarbeidspartnere. De to første kriteriene til Frost & Shoen (2004), som VideoLAN har gjort et solid arbeid i forhold til, virker ikke å ha vært like prioritert hos Xiph. Terskelen for å delta i både kommunikasjonskanalene og utviklingsarbeidet har vært betydelig høyere her. Selv enkelte bidrag av teknisk god kvalitet har blitt holdt utenfor i lengre perioder, som for eksempel aoTuV sine optimaliseringer av Vorbis-kodeken.

Dette har gått ut over både rekrutteringen av utviklere og ikke-teknisk personell (til hjelp for eksternalisering i form av dokumentasjon, bruksanvisninger og lignende). I tillegg har relasjonene til potensielle eksterne samarbeidspartnere fått lite pleie. Dette problemet var nok grunnleggeren av Vorbis klar over, ettersom han i 2004 annonserte på et månedlig IRC-møte at han ville trappe ned sin innsats i selve kodingen, til fordel for å arbeide med Xiphs ansikt utad. Kort oppsummert kan vi altså si at Xiph har plassert seg et stykke mot kontroll i Holck & Jørgensens (2005) skala, men at de nå forsøker å bevege seg noe nærmere anarkiet (som illustrert i figur 7).

Figur 7: Pendelen mellom anarki og kontroll i Xiph



5.4 Kunnskapsledelse i ENoP

Både VideoLAN og Xiph kan beskrives som elektroniske praksisnettverk. I forrige kapittel så vi hvordan kunnskapsarbeidet i nettverkene var blitt preget av deres ulike tilnærminger til problemet med å balansere anarki og kontroll. Denne balansegangen er altså en viktig del av kunnskapsarbeidet i et EnoP, som man bør være bevisst på for å kunne lede nettverket i riktig retning. Dette bringer oss over i de utfordringene som ligger i kunnskapsledelse generelt, og for OS-prosjekter spesielt. I dette kapittelet vil jeg se nærmere på disse utfordringene, samt undersøke hvordan mine case har håndtert dem.

5.4.1 Endringer i nettverket

Frost & Shoens fjerde kriterium for et fungerende praksisfelleskap er at man må (4) *Virke med fellesskapets ytre og indre miljø.* Dette belyser en viktig lederutfordring, nemlig å sikre at man har en fleksibel og omstillingsdyktig organisasjon og kultur. For mine to case har denne utfordringen vist seg på ulik vis, men i begge tilfeller har det vært snakk om en utvikling over tid, som har gjort behovet for endring i nettverket tydelig. Vi så i kapittel 5.3 hvordan VideoLAN hadde vokst seg stor på kort tid, men likevel beholdt en arbeidsform som ligger nærmere et praksisfellesskap enn et elektroniske praksisnettverk. Dette har gitt grunnlag for et ønske om å flytte pendelen mot mer kontroll, for å sikre kvalitet og forutsigbarhet i eget produkt. Xiph på sin side har hatt større kontroll på utviklingsprosessen, og et klarere fokus og retning på sitt arbeid. De har på sin side ikke klart å samle en stor nok gruppe frivillige til å sikre tempo i utviklingen, og mangler også ressurser til å utføre noen av de ikke-tekniske oppgavene til organisasjonen. Her oppstår nå et ønske om en dreining mot mer anarki og åpenhet overfor eksterne. Med andre ord står begge to overfor den samme utfordringen; å skape endringer i det etablerte nettverket. Hvordan kan dette gjøres?

5.4.2 Det velmenende diktatur

For å kunne si noe om hvordan man kan lede kunnskapsarbeidet i et OS-prosjekt, er det nødvendig å gå dypere inn i lederrollen innenfor OS-miljøene. Som et resultat av OS-kulturens egen variant av gaveøkonomi, blir autoritet og lederskap først og fremst knyttet til innsats og evne til å bidra til fellesskapet. Erfaring og fartstid i nettverket er også et viktig kriterie for autoritet, der grunnleggeren for et nettverk som oftest har en spesiell status. Så lenge grunnleggeren fortsatt tar aktivt del i nettverkets arbeid, og ikke har gitt stafettpinnen over til noen andre, vil denne som regel ha siste ordet i beslutninger som gjelder prosjekter han selv har igangsatt. Grunnleggeren av Xiph omtaler seg selv som en «velmenende diktator», i den forstand at han arbeider for fellesskapet, ikke seg selv, men ser det som sin rett til en viss grad å definere hva som er til fellesskapets beste.

Denne statusen gir imidlertid ikke noen klar og entydig autoritet i klassisk forstand. Eventuelle eksisterende formelle og uformelle makt- og statusstrukturer blir som regel underkommunisert i OS-prosjekter. Arven fra Richard Stallman og the Free Software Foundation setter fortsatt sitt preg på OS-bevegelsen, ved at man holder det demokratiske likhetsidealet høyt, og i liten grad aksepterer noen form for autoritet utover det skrevne ords egen logikk. Enkelte lederskikkelser høster likevel spesiell oppmerksomhet og høflighet fra sine medutviklere, men uten at dette gir dem mandat til å diktere eller beordre andre.

I Xiph sitt tilfelle kommer denne autoriteten til uttrykk ved at man får ansvar for å vedlikeholde en eller flere moduler, og dermed fungerer som en portvakt overfor større endringer i koden. Dermed blir man en slags diktator over sitt spesifikke domene innenfor organisasjonen. Xiph har også et oppnevnt styre som skal håndtere ekstern offisiell kontakt, og som er satt sammen av disse portvaktene. Disse posisjonene blir ikke utnevnt ved valg, men ved at de mest aktive innenfor et område av organisasjonen over tid, blir gitt statusen som portvakt. Det forventes at portvaktene forblir blant de mest aktive utviklerne på «sitt» prosjekt og deltar hyppig i den faglige debatten via prosjektet kommunikasjonskanaler. Dette ser vi eksempler på hos Xiph, ved at portvaktene generelt står for en stor andel av oppdateringer i koden til sine prosjekter, samtidig som de er blant de mest aktive på de respektive e-postlistene. Hvis en annen utvikler i en lengre periode er mer aktiv enn portvakten, oppstår det etter hvert en forventning i miljøet om et bytte i lederskap, i tråd med OS-kulturens fokus på at koden er et felles gode, ikke en personlig eiendel (jamfør Raymond (1997) sin femte regel i kapittel 2.1.5).

VideoLAN har ikke i like stor grad formalisert portvakt-funksjonen, men har likevel en utstrakt arbeidsdeling, og de mest aktive innenfor hver sin modul gis en naturlig autoritet i forbindelse med endringer som påvirker deres arbeid. I enkelte tilfeller kan det være flere som peker seg ut som «eiere» av en modul, noe som kan resultere i diskusjoner og konflikter. Jeg observerte lite av dette, men ble fortalt at det hadde forekommet ved flere anledninger tidligere. De fysiske møtene ved Ecoles Centrale Paris kunne være svært opphetede, og IRC-kanalen til VideoLAN hadde også vært arena for harde forhandlinger. Alle disse ble imidlertid løst i fellesskap, uten at man fikk noen splittelse i organisasjonen. Bruken av rike medier, som fysiske møter og sanntidskanaler som IRC, kan ha vært en medvirkende årsak til at man lykkes med å finne en felles forståelse i slike tilfeller.

Autoritet er altså knyttet til innsats og faglig dyktighet, og blir ofte underkommunisert i OS-miljøene. Dette fører til at en lederskikkelse i miljøet først og fremst har sin egen praksis og sin egen evne som taler eller skribent, som effektive midler til å påvirke kunnskapsarbeidet innad i organisasjonen. Denne tankegangen finner vi igjen i Frost & Shoen (2004) sitt siste kriterium for et velfungerende praksisfelleskap, nemlig at man må (5) *Leve ut fellesskapets verdier og normer.* Her understrekes viktigheten av at lederskapet går foran med et godt eksempel, i forhold til å overholde fellesskapets egne verdier og normer. Det engelske uttrykket «practice what you preach» (løst oversatt: «praktiser det du forfekter»), fanger opp mye av det samme poenget. Hvis det er merkbar avstand mellom ledelsens praksis og de normene som er ment å gjelde for nettverket som helhet, vil som regel praksisen bli toneangivende for nettverkets deltakere, ikke normene. Dette er noe de

fleste OS-ledere er klar over, noe vi kan se et eksempel på blant annet i et gammelt innlegg fra grunnleggeren av Xiph på en intern e-postliste, der han beklager tidligere bruk av direkte kommunikasjon, og lover at han og de andre kjerneutviklerne skal bli flinkere til å benytte offentlige e-postlister som hovedkanal for kommunikasjon. I stedet for å beordre alle andre at de skal bli flinkere til å bruke e-postlistene, velger han altså her å sette fokus på sin egen praksis og dermed indirekte gi uttrykk for hvordan han mener kommunikasjonen skal foregå. Med en slik strategi kan ledelsen unngå å bli oppfattet som autoritære, samtidig som de utøver en betydelig påvirkning på hvordan arbeidet i organisasjonen går for seg.

5.4.3 Strategiske valg i oppbyggingen av et ENoP

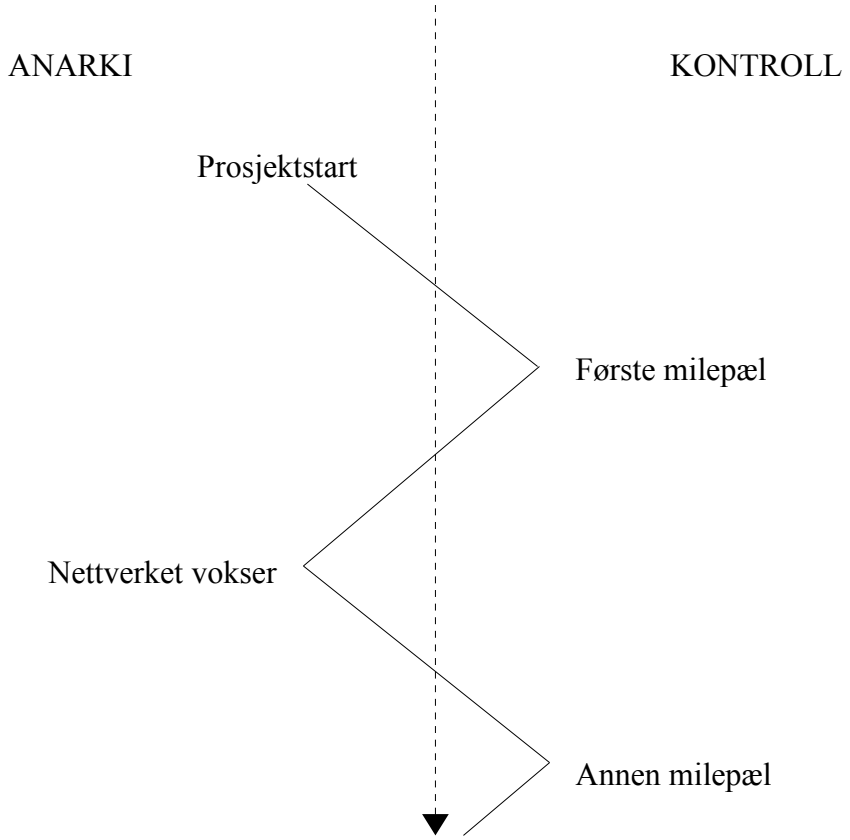
Historien om VideoLAN viser hvordan det lokale praksisfelleskapet kan fungere som en katalysator for godt kunnskapsarbeid. Ved å opprettholde en åpen og inkluderende arbeidsform har prosjektet lykkes med å bygge opp et stort og dynamisk utviklingsmiljø, der en kjerne av utviklere driver prosjektet framover. Rundt denne kjernen virker en større mengde med bidragsytere, der grensen mellom bruker og utvikler er flytende og der bidragene justeres etter kompetansenivå. Anarkiet blir her en viktig faktor for å sikre rekruttering i alle ledd, og for å sikre tilstrekkelig åpenhet i kommunikasjonen. Det kan derfor argumenteres for at et hvert utviklingsprosjekt, enten det er OS eller ikke, vil tjene på å ha en forankring i et lokalt praksisfelleskap, før man eventuelt bygger dette videre til et elektronisk praksisnettverk.

Ved opprettelsen av et elektronisk praksisnettverk, vil omfanget av kommunikasjonen bli av en slik størrelse at man trenger andre mekanismer for kunnskapsoverføring, i tillegg til de som er typiske for det lokale praksisfelleskapet. Det blir også behov for i større grad å balansere anarkiet med kontrollmekanismer, i form av strengere normer for kommunikasjon, og bruk av kommunikasjonskanaler som egner seg for arkivering og systematisering.

I dette scenariet blir det viktig at prosjektets lederskikkelser tar de strategiske valg som er nødvendig for a balansere mellom anarki og kontroll. Ut i fra de observasjoner jeg har gjort i mine case, vil jeg anbefale et fokus på anarki i startfasen av et prosjekt, slik at man sikrer en kommunikasjon basert på åpenhet og tillit. Etter hvert som nettverket modnes og får et klarere definert fokus, kan man begynne å konsolidere arbeidsform og organisasjon i større grad. Her er det viktig å finne det riktige tidspunktet å begynne å snu pendelen mer mot kontroll. Hvis man begynner for sent kan man risikere at viktig prosessinformasjon går tapt, og hvis man begynner for tidlig risikerer man at prosjektet mister sitt momentum. Ved å sette ned konkrete milepæler, å

justere balansen i forhold til disse, minsker risikoen for at pendelen svinger for langt ut til ene siden. Figur X illustrerer denne tankemåten i en fortettet og forenklet form:

Figur X. Balansere anarki og kontroll over tid (Y-aksen angir tid):



5.5 Konklusjon

OS-utvikling lar seg beskrive som elektroniske praksisnettverk, der ulike mekanismer for overføring av taus kunnskap er en viktig del av utviklingsarbeidet – samt bidrar til å holde nettverket samlet. Jeg har observert at dette skjer enten via uformelle og spontane fortellinger, som i tilfellet med bruken av IRC i VideoLAN, eller via utvekslingen av konkrete eksempler over e-postlister, slik vi så hos Xiph. Årsaken til denne forskjellen i casene, mener jeg å spore tilbake til deres opprinnelse, der VideoLAN hadde en forankring i et lokalt praksisfelleskap ved Ecoles Centrale Paris, mens Xiph ble dannet som et rent elektronisk praksisnettverk fra starten.

Måten kunnskapsarbeidet i VideoLAN foregår på gir fordeler i form av et sosialt og attraktivt miljø for nye brukere og utviklere, og dermed et hurtig tempo i utviklingen. Ulempene kommer i form av manglende arkivert prosessinformasjon og et uklart fokus eller mål for utviklingen. Xiph har på sin

side opplevd vanskeligheter med å samle et stort nok miljø til å få utført de mindre teknisk krevende oppgavene sine, men har klart å holde et tydeligere fokus i sitt utviklingsarbeid. Xiph har også bygget opp en mengde prosessinformasjon i form av strukturerte e-postarkiver, som utgjør en potensielt kraftfull database for både taus og eksplisitt kunnskap.

Det bør være et mål å prøve og beholde aspekter ved begge måtene å overføre kunnskap på; både de uformelle spontane fortellingene over IRC, og de konkrete eksemplene på e-postlistene. Dette krever at de mest toneangivende i nettverket er bevisste på styrkene og svakhetene ved de ulike kommunikasjonskanalene som er til rådighet, og går foran med et godt eksempel i bruken av disse.

5.5.2 Eksempler på videre studier

Mange utviklingsprosjekter, innenfor OS og ellers, går veien fra å være et lite praksisfelleskap til et større elektronisk praksisnettverk. Det kan være interessant å undersøke flere slike tilfeller, for å opparbeide seg en dypere forståelse av hvordan en slik organisatorisk endring går for seg. For eksempel kunne man som forsker vært deltakende observatør i en slik overgangsprosess, og der igjennom lære mer om de utfordringene man møter.

En annen interessant mulighet er å utforske nye og mer interaktive måter å bruke multimedia til å etablere uformelle arenaer på. Gjennom ny audiovisuell teknologi kan man forsøke å gjenskape fordelene ved et fysisk forankret praksisfelleskap. Dette krever både en god forståelse av hvordan praksisfelleskapet fungerer, samtidig som man må overkomme enkelte tekniske barrièrer slik at den virtuelle kommunikasjonskanalen oppfattes og brukes som om den var direkte og lokal.

For bedre å utnytte det potensialet en e-postliste har som kunnskapsdatabase, trenger man verktøy som kan gjøre informasjonen som ligger der lettere tilgjengelig for den enkelte (ved hjelp av søking, sortering, intelligente informasjonsagenter og lignende). For eksempel kunne man i tilknytning til en e-postliste, også ha et felles system for katalogisering av spesielt nyttige e-postinnlegg, til ulike formål. Systemet kunne gjerne vært inspirert av wiki-formatet, i den forstand at alle som deltok på listen hadde tilgang til å revidere «katalogen». Det hadde vært interessant å forsøkt og utvikle slike typer applikasjoner, og deretter fått et eksisterende OS-prosjekt av noenlunde størrelse til å teste ut en eventuelle prototype.

6. Bibliografi

Earl Babbie & J. Mouton, *The Practice of Social Research*, Oxford University Press, 2001

Thomas Basset, «Coordination and Social Structures in an Open Source Project: VideoLAN», *Free/Open Source Software Development,* Idea Group Inc*.,* 2005*,* Side 125-151

Nick Bontis, M. Fearon & Marissa Histon, «The e-flow audit: an evaluation of knowledge flow within and outside a high tech firm», *Journal of Knowledge Management,* Vol.7, Nr.1, 2003, Side 6-19

Marcus Bergquist & J. Ljungberg, «The power of gifts: organizing social relationships in open source communities», *Information Systems Journal,* Vol..11, 2001, Side 205-220

Nikolai Bezroukov*,* «Open Source Software Development as a Special Type of Academic Research (Critique of Vulgar Raymondism)», *First Monday,* Vol. 4, Nr. 10, 1999

Frederick Brooks, *The Mythical Man-Month,* Addison-Wesley, 1975

Tony Cornford & S. Smithson, *Project Research in Information Systems: A Student's Guide*, Palgrave, N.Y. 1996

Alan Cox, *Cathedrals, Bazaars and the Town Council, (http://slashdot.org/features/98/10/13/1423253.shtml),* 1998

T. H. Davenport, D. W. De Long & M. C. Beers, «*Successful knowledge management projects*», *Sloan Management Review*, Vol.39, Nr.2, 1998

Alan M. Davis, E. H. Bersoff & E. R. Comer, «A Strategy for Comparing Alternative Software Development Life Cycle Models», *IEEE Transactions on software engineering,* Vol.14, Nr.10, 1998

J. Roberto Evaristo & E. Karahanna, «Is North American IS Research Different from European IS Research», *The DATA BASE for Advances in Information Systems,* Vol.28, Nr.3, 1997

Benjamin Frost & S. Schoen, «Viable Communities within Organizational Context: Creating and Sustaining Viebility in Communities of Practise at Siemens AG», *Knowledge Networks: Innovation Through Communities of Practise,* Idea Group Inc., 2004, Side 133-141

Robert D. Galliers & F. F. Land, «Choosing Appropriate Information Systems Research Methodologies», *Communications of the ACM,* Vol.30, Nr.11, 1987, Side 900-902

Robert L. Glass, V. Ramesh & I. Vessey, «An Analysis of Research in Computing Disciplines», *Communications of the ACM,* Vol.47, Nr.4, 2004

Roelien Goede & C. De Villiers, «The Applicability of Grounded Theory as Research Methodology in studies on the use of Methodologies in IS Practices», *Proceedings of SAICSIT,* 2003, Side 208-217

Jonathan Grudin, «Why Groupware fail: Problems in design and evaluation», *Office: Technology and People,* Vol.4, Nr.3, 1989, Side 245-264

Gisle Hannemyr, *Technology and pleasure: Hacking Considered Constructive* (*http://folk.uio.no/gisle/essay/oks97.html),* 1997

M. Hatling & K. H. Sørensen, «Social Construction of User Participation», *The Spectre of participation: Technology and work in a welfare state,* K.H.Sørensen*,* Scandinavian University Press, Oslo, 1998, Side 171-188

Kunihiko Higa, O. R. Liu Sheng, B. Shin & A. J. Figueredo, «Understanding Relationships Among Teleworkers' E-mail Usage, E-mail Richness Perceptions, and E-mail Productivity Perceptions Under a Software Engineering Environment», *IEEE Transactions on Engineering Management,* Vol.47, Nr.2, 2000, Side 163-173

Eric von Hippel & G. von Krogh, «Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science», *Organizational Science,* Vol.14, Nr.2, 2003

Jesper Holck & N. Jørgensen, «Do not check in on red: Control meets anarchy in two open source projects», *Free/Open Source Software Development,* Idea Group Inc., 2005, Side 1-26

Heinz K. Klein & M. D. Myers, «A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems», *MIS Quarterly,* Vol.23, Nr.1, 1999, Side 67-94

Ned Kock, «Can communication medium limitations foster better group outcomes? An action research study», *Information & Management*, Vol.34, 1998, Side 295-305

Roger Kolbotn, «Communities of Practise in the Royal National Lifeboat Institution», *Knowledge Networks: Innovation Through Communities of Practise,* Idea Group inc*.,* 2004*, Side 70-78*

Georg von Krogh & S. Spaeth & K. Lakhani, «Community, joining, and specialization in open source software innovation: a case study», *Research Policy,* Vol.32, 2003, Side 1217-1241

Maarten de Laat & W. Broer, «CoPs for Cops: Managing and Creating Knowledge through Networked Expertise», *Knowledge Networks: Innovation Through Communities of Practise,* Idea Group Inc., 2004, Side 58-69

David R. Millen & M. A. Fontain, «Improving individual and organizational performance through communities of practise», *Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*, 2003, Side 205-211

Robert L. Miller & J. D. Brewer, *The A-Z of Social Research,* SAGE Pub. London, 2003

Audris Mockus, R. T. Fielding & J. Herbsleb, «A case study of open source software development: the Apache server», *Proceedings of the 22nd international conference on Software engineering*, Vol.4, Nr.11, 2000, Side 263-272

Michael D. Myers, «A disaster for everyone to see: an interpretive analysis of a failed IS project», *Accounting, Management and Information Technologies,* Vol.4, Nr.4, 1994, Side 185-201

I. Nonaka & H. Takeuchi, «A Theory of the Firm's Knowledge-Creation Dynamics», *The Dynamic Firm: The Role of Technology, strategy organisation and regions,* Chandler et.al, Oxford University Press, Oxford, 1998, Side 214-241

W. J. Orlikowski & J. J. Baroudi, «Studying Information Technology in Organizations: Research Approaches and Assumptions», *Information Systems Research,* Vol.2, 1991, Side 1-28

Julian E. Orr, «Talking about Machines: An Ethnography of a Modern Job», *Collection on technology and work,* Stephen R. Barley, Cornell University Press, 1996, Side 124-143

Shaun Pather & D. Remenyi, «Some of the Philosophical Issues Underpinning Research in Information Systems: From Positivism to Critical Realism», *Proceedings of SAICSIT,* 2004, Side 141-146

Bruce Perens, «The Open Source Definition», *Open Sources; Voices from the Open Source Revolution,* Chris DiBona, Sam Ockman & Mark Stone, O'Reilly & Associates, 1999

John C. Paolillo, «The Virtual Speech Community: Social Network and Language Variation on IRC», *Journal of Computer-Mediated Communication,* Vol.4, 1999

Nelson Phillips & C. Hardy, «Discourse Analysis: Investigating Processes of Social Construction», *Qualitative Research Methods Series,* Vol.50, SAGE Pub. London, 2002

Roger S. Pressman, *Software Engineering: A Practitioner's Approach,* McGraw-Hill, 1997

Eric S. Raymond, «A Brief History of Hackerdom»*, Open Sources; Voices from the Open Source Revolution,* Chris DiBona, Sam Ockman & Mark Stone, O'Reilly & Associates, 1999

Eric S. Raymond, *The Cathedral and The Bazaar (http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/index.html),* 1997

Andrew Shenkel, «Investigating the Influence that Media Richness has on Learning in a Community of Practice: A Case Study at Øresund Bridge», *Knowledge Networks: Innovation Through Communities of Practice,* Idea Group Inc., 2004, Side 47-57

Richard Stallmann, «The GNU Operating System and the Free Software Movement», *Open Sources; Voices from the Open Source Revolution,* Chris DiBona, Sam Ockman & Mark Stone, O'Reilly & Associates, 1999

Frederick W. Taylor, *The Principles of Scientific Management*, New York, Harper Bros, 1911

Robin Teigland & M. M. Wasko, «Extending Richness with Reach: Participation and Knowledge Exchange in Electronic Networks of Practise», *Knowledge Networks: Innovation Through Communities of Practice,* Idea Group Inc., 2004, Side 230-242

Marco Torchiano & M. Morisio, «Overlooked aspects of COTS-based development», *IEEE Software*, Vol. 21, Nr. 6, 2004, Side 62-69

David Turnbull, «The Ad Hoc Collective Work of Building Gothic Cathedrals with Templates, String and Geometry», *Science, Technology & Human Values,* Vol.18, Nr.3 1993

Janaka Wijayanayake & K. Higa, «Communication media choice by workers in distributed environment», *Information & Management,* Vol.36, 1999, Side 329-338

Rober K. Yin, «The Case Study Crisis: Some Answers», *Administrative Science Quarterly,* Vol.26, Nr.1, 1981, Side 58-65

Appendix A: Verktøy for uthenting og behandling av data fra Subversion

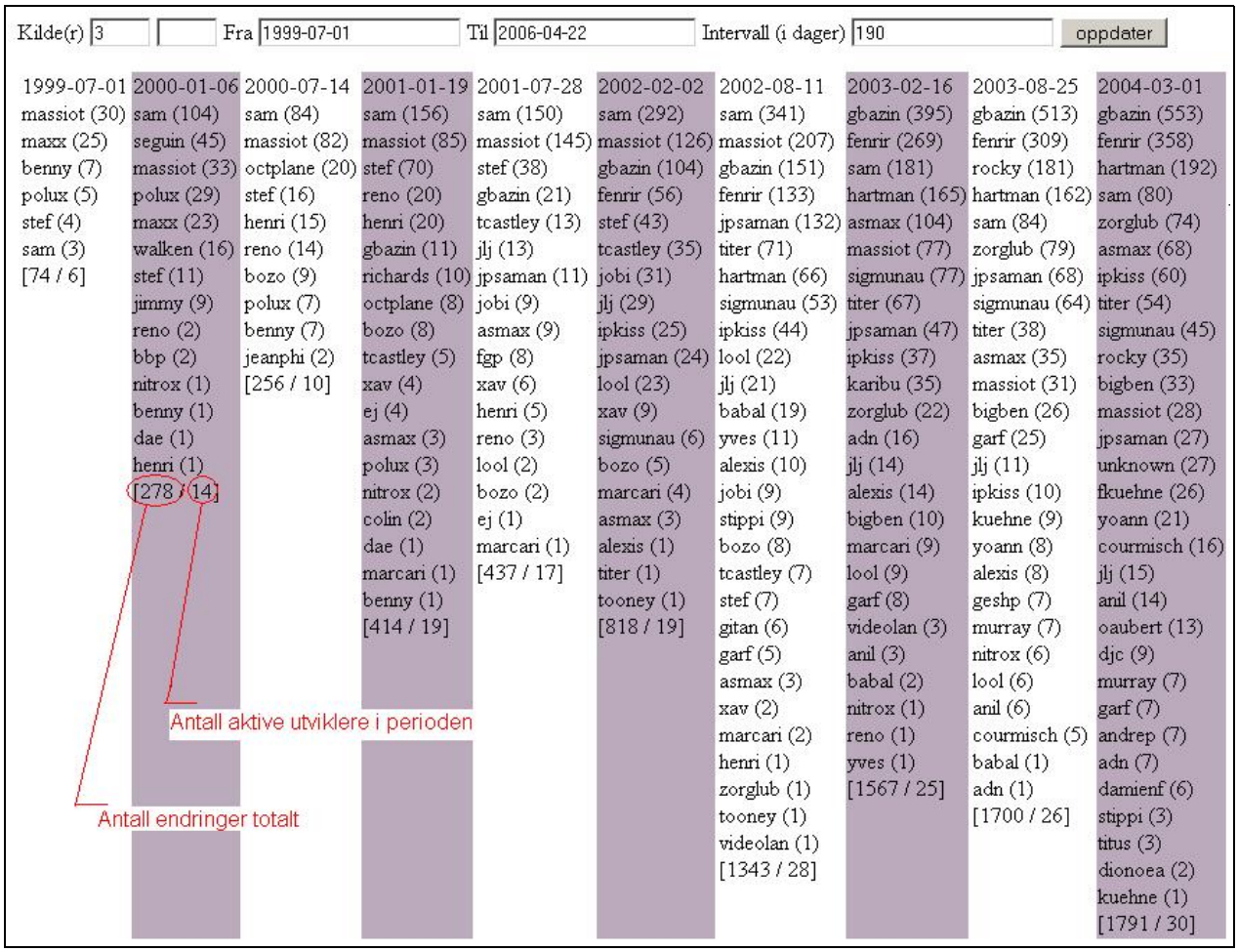
Data til behandling ble hentet fra følgende nettsider:
VideoLAN: *http://trac.videolan.org/vlc/log*
Xiph: *http://trac.xiph.org/log/trunk/vorbis* og *http://trac.xiph.org/log/trunk/theora*

Informasjonen ble så kjørt gjennom et skript skrevet i PHP[27], som la alt inn i en database av formatet MySQL[28]. Dette ble utført på mitt område på universitetets studentserver (stud.ntnu.no).

Databasen besto av en tabell med følgende felter:

Commits_ID: Unik ID for hver endring i Subversion
Kilde_ID: Unik ID som henviste til en tabell over ulike kilder (VideoLAN, Vorbis eller Theora)
Date: Dato for endring i Subversion
Rev: Revisjonsnummer i Subversion
Author: Navn på utvikler som gjorde endringen (brukernavn i Subversion)
Summary: Kort tekst som beskriver hva endringen går ut på, legges inn av utviklerne

Et nytt PHP-skript ble så laget for å behandle disse dataene via en nettleser, som ble grunnlaget for de grafene jeg presenterer i kapittel 4. Her er et skjermbilde fra PHP-skriptet i arbeid:



[27]Se www.php.net
[28]Se www.mysql.com

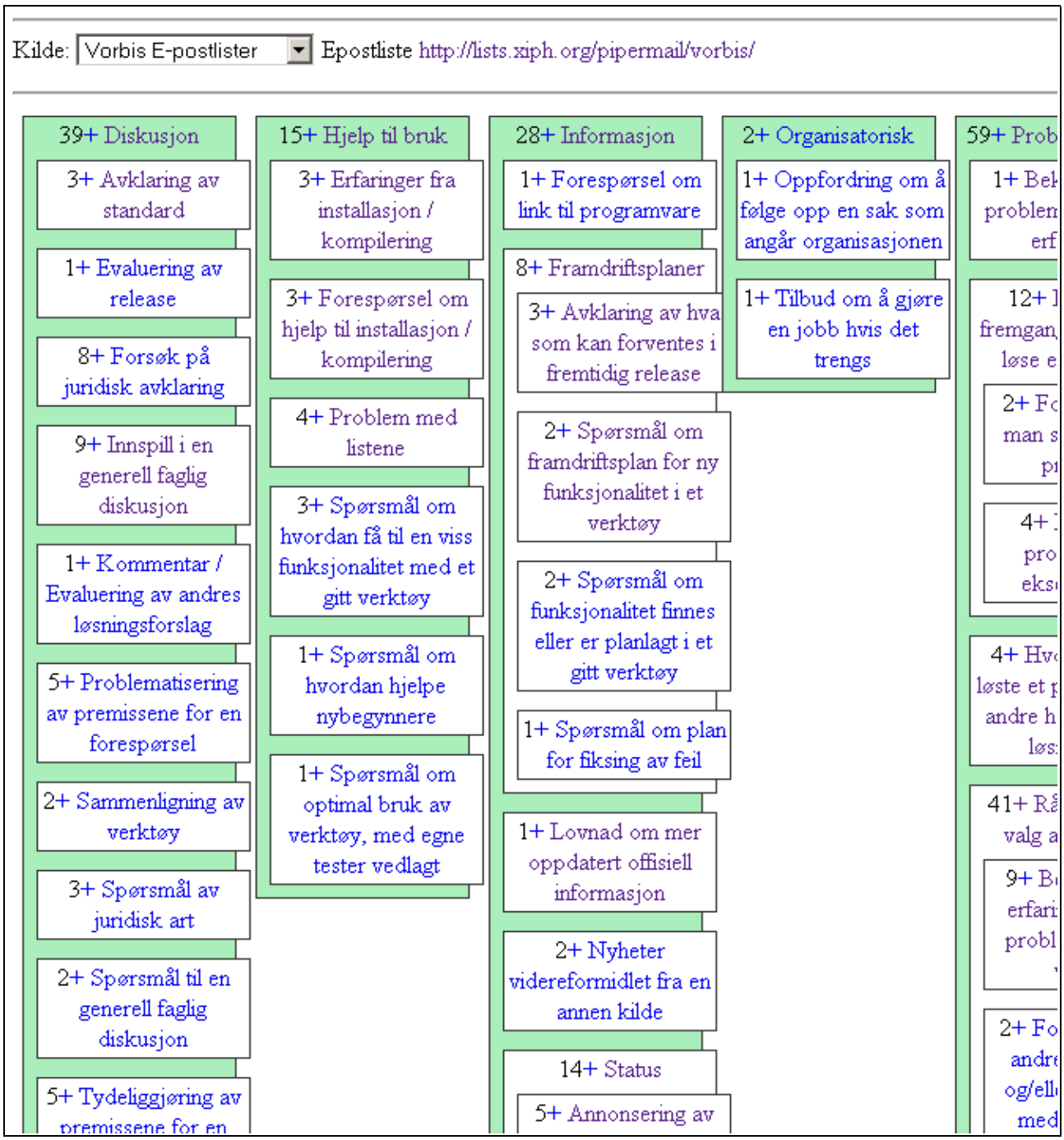
Appendix B: Verktøy for katalogisering av e-postinnlegg

Data til behandling ble hentet fra følgende nettsider:
http://lists.xiph.org/pipermail/vorbis/
http://lists.xiph.org/pipermail/vorbis-dev/
http://lists.xiph.org/pipermail/theora/
http://lists.xiph.org/pipermail/theora-dev/

Innleggene fra januar til og med mars 2005 ble lest gjennom ett og ett, og lagt inn i databasen. For hvert innlegg ble det enten opprettet en ny kategori, eller det ble tilknyttet en eksisterende.

Verktøyet ble skrevet ved hjelp av PHP, HTML og MySQL.

Her er to skjermbilder fra verktøyet i bruk:
Bilde 1 – Hierarkisk oversikt over kategorier med antall inlegg (for valgt kilde):



Bilde 2 - Visning av kategori og enkeltinnlegg: