

Open Source Community – Kreativitet, Vitensdeling og Samhold i Systemutvikling

av Hilde M. Hølmebakk

**Spesiale ved Informationsvidenskab, Institut for Informations og
Medievidenskab, Aarhus Universitet, November 2002**

Takk til!

Takk til guttene mine, Per, Halvor og Sigbjørn, fordi dere har vist meg tillit og fordi dere har fulgt meg hele veien. Takk til min veileder Ellen Christiansen - for hennes engasjement, generøsitet, og faglige og menneskelige kvalifikasjoner. Ikke minst vil jeg takke Mads Toftum, som alltid har vist meg imøtekommenhet og svart på alle mine email og spørsmål. Takk til Rob Wood for hjelp med engelsk abstract. Takk til alle mine herlige venner og familie, for støtte og oppmuntring underveis. Ikke minst takker jeg dere alle for deres tålmodighet! Lange kvelder med monologer om schismogenese og hackere og nerder og systemutviklingstradisjoner - Så mange har *villet med meg*, og vært generøse på mine vegne for at jeg skulle nå mitt langsiktige mål. Det er jeg dere alle dypt takknemmelig for.

til Per

Innholdsfortegnelse

Open Source Community – Creativity, Knowledge-Sharing and Solidarity in System Development

Hovedinnledning 1

Del 1

Hjernen er alene

Innledning	7
Først litt om systemutvikling	8
Kreativitet, metaspråk og metakommunikasjon	14
Kommunikasjonsteori	17
Batesons 'schismogenese' og 'ethos' begrep	19
Kunstens epistemologi.....	22
Metafor og metonymi	28
Primær Prosesser	32
Kunstens korrigerende natur	36
Oppsummering	37

Del 2

Vitensdeling i systemutvikling

Innledning	40
Forskjellige former for vitensdeling	42
Vitensdeling knyttet til spr. om intellektuell kapital og økonomi:	42
Vitensdeling knyttet til spørsmål om kompleksitet	42
Vitensdeling knyttet til spørsmål om organisering og metode	43
Vitensdeling knyttet til spørsmål om læring.....	43
Det medierende perspektiv i synet på læring og læreprosesser	43
Hacker kulturens opprinnelse.....	45
Utviklingen av Unix	48
Unix blir proprietær kode.....	50
Computerindustriens ekspansjon og diskusjonene om opphavsrett	51
Richard Stallman og Free Software Foundation	53
Linux prosjektet	55
Apache prosjektet	59
Dannelsen av Open Source Community	60
Oppsummering	63

Del 3

Verdisystemer innen Open Source kulturen; merittbasert ridderkultur og pragmatisk fellesskap i cyberspace.

Innledning	66
Sammenfatning og presentasjon av Raymonds fremstilling	66

Innholdsfortegnelse

Hva er felles for miljøet? Kampen mot Microsofts monopol	68
Halloween dokumentene.....	71
Observerbar adferd (schismogenese)	73
Relasjoner – De som er innenfor og de som er utenfor.	
Hvordan kommer man inn?	73
Hvordan forholder man seg til konflikter innad og mot utenforstående.	76
Konflikter mot utenforstående.....	76
Konflikter innad i miljøet.....	80
Konfliktløsning	81
Formelle teknikker for sosial påvirkning	83
Hvem er ledere?.....	83
Hva er tabu?	84
Underliggende verdier (ethos)	86
Hva slags forhold har man til økonomi?.....	86
Hva slags former for belønning finnes i kulturen?	87
Verdisystemer innen Open Source kulturen - oppsummering.....	89
Litteratur & Kilder	
Litteratur	95
Lenker	96

Open Source Community – Creativity, Knowledge-Sharing and Solidarity in System Development

The Open Source Community is an Internet culture with its focus on software development. People co-operate on diverse projects in an international network, to develop computer programs for free public use. Open Source Community is a conglomerate of different small and big projects and organizations, who cover different needs in software development. This paper is in three sections, corresponding to the three spheres of examination: the individual, the group, and the social sphere. I will now present my theses in accordance with this threefold division.

Section One: The Individual Sphere

Within the culture of the OSC there is a common activity – development of program codes. This it has in common with many software developers. But what does it mean to code? As I have read, explored and gathered material on this theme, and come in contact with system developers, certain phrases and expressions have come up, that tell us something about developers views of themselves. I think of expressions such as “code poets”, artists etc. Then I wanted to examine closer, what do people imply by that?

Thesis: Programming as an individual is a lonely process. By regarding the programmer as an author, who is totally involved in the creation of a text, I would like to use Inger Lytje, Peter Naur and Roman Jacobson to further my understanding of software development as text. I would also like to refer to Bateson's description of the artistic process or play and compare that with the coding process. In this way I will show some of the creative aspects and innovations which are characteristic of system development.

Section Two: The Group Sphere

This paper has system development as its focus. The second section is an attempt to present the general environment of OSC. It is also an attempt, by analysis, to show that this form of system development demonstrates diverse forms of knowledge-sharing. To share knowledge means that you show others what you have made, it entails openness. In such a culture, there are many different aspects of knowledge-sharing. One of them is related to copyright and economic issues. Another is related to questions about complexity. The third is connected to the organisation and methodology in development, and the fourth is concerned with questions relating to learning. The thesis for part two follows:

Thesis: By analysing the development of the OSC, I would like to show that the full extent and quality of knowledge-sharing depends on openness in economic and intellectual capital, complexity, organisation & methodology, and learning. Vygotsky's mediating perspective shows that the human ability to use tools, to learn, and therefore to develop, is culturally conditioned. I want to show that these different kinds of knowledge-sharing are creating an environment of learning processes, and therefore learning itself.

Section Three: The Social Sphere

In this section I will describe the OSC from an anthropological point of view. I will use Bateson's concept of 'schismogenesis' and 'ethos' as an explanatory model. In addition to personally collected material, I will use Eric S. Raymond's anthropological essays as a basis for my description.

Thesis: Thanks to the Internet programmers today can view themselves as participants in a global society. They can share values in spite of differences in economy, geography and language. By describing the observable behaviour as well as its underlying values, I aim to describe, the various value systems within this culture.

These three theses have one thing in common: they are all concerned with different types of communication within a net culture. If one writes a program or considers it as text production, so one is dealing with communication. Different forms of knowledge-sharing are likewise linked to communication. An anthropological approach to a

culture is also a description of different forms of communication. I have chosen a theory and a set of ideas, that are appropriate to the phenomena that I have examined. Therefore I have relied heavily on Gregory Bateson's *Steps to an Ecology of Mind*. Bateson perceives the mental, or mind, as communication. His starting point is always the natural system, comprised of living organisms in their environment, and he uses the theory of communication and the language of cybernetics in his work. By focusing on form, Bateson always returns to what communication is all about; the interactions of living creatures with each other and with the world around them.

The first section - *The Brain is Alone* - is meant to be read as the theory section. Here I try to draw in different theories to underpin my thesis concerning creativity. The second section - *Knowledge Sharing in System Development* - analyses collected material in light of the thesis of knowledge-sharing. The third section has an anthropological angle, and is therefore a description of the Open Source culture. In my approach to this work, I have taken some basic assumptions. These are as follows:

- What people say they do is one thing.
- What people think they do is another
- What people actually do is something else

The first assumption implies that in such a culture there exists a discourse. This I have tried to get a knowledge of through interviews, by reading up on the subject, and not least by following different discussions on the Internet. There is a difference between what we say and what we think. By describing and analysing the behaviour and praxis, one can uncover some of the underlying values of a culture; its 'ethos'. That is the aim of the third section, and the title *Value Systems in Open Source Community – A Chivalrous Meritocracy and Pragmatic Solidarity in Cyberspace* reflects some patterns I have discovered in the Open Source culture. My intention with this paper is not to end up with any definite “scientific” conclusions or answers about Open Source Community. My humanistic background has given me an understanding of technology as a human expression. My propositions should be interpreted as exactly what they are; an outsider's attempt to examine some aspects of this culture, from the outsider's point of view.

Open Source Community – kreativitet, vitensdeling og samhold i systemutvikling

Hovedinnledning

Open Source Community (OSC) er betegnelsen på en nettkultur. Denne kulturen har systemutvikling som omdreiningspunkt, hvor aktiviteten består i at mennesker arbeider sammen på forskjellige prosjekter i et internasjonalt nettverk, for å utvikle programkode som er tilgjengelige for offentligheten.

Når man skal skrive en oppgave om dette miljøet, finnes det mange mulige tilganger. Materialet er stort; hva slags avgrensning skal man foreta seg? Hva slags problemstilling(er) skal lede skriveprosessen? Skal man gå inn i tekniske spissfindigheter for å forstå hva dette dreier seg om? Spørsmålene har vært mange. Min hovedintensjon har vært å forsøke å få en helhetlig innsikt i et miljø, og en kultur. Jeg er utenforstående, og nettopp i kraft av ikke å være deltager i miljøet har mitt ønske vært å betrakte dette fenomen med den utenforståendes blick. I den tiden jeg har gått på universitetet har jeg skrevet mange oppgaver. Gang på gang har jeg synes det var vanskelig å skulle beskrive min egen metode og min egen prosess ut i fra de vitenskapsteoretiske idealer, hvor man starter med en klar problemformulering, tester denne mot innsamlet empiri, for så å avkrefte eller bekrefte hypotesen. (Hypotetisk-deduktiv-metode) Mitt mål har heller ikke vært å skrive en etnografisk oppgave, (en ren beskrivelse) ved å bruke en rendyrket kvalitativ metode. Allikevel minner vel fremgangsmåten på mange måter om den kvalitative forsker; i innsamling av materialet (intervju) og i rollen som observatør.

For mange år siden arbeidet jeg på et omreisende teater i Norge. Jeg la merke til en scenograf som arbeidet der. Når han var i villrede om sin dekorasjon og var i tvil om løsningen, så kom han ofte bærende med et bjerketre og plantet det på scenen i mangel av noe bedre. Ofte har jeg tenkt på at jeg minner om ham, fordi jeg i mine tidligere innledninger så ofte har slengt det samme bjerketre frem: *eksplorativ metode!* Vel, denne gangen har treet utviklet seg til en hel skog! Sannheten er at jeg har arbeidet

meg frem til mine problemstillinger, eller teser, underveis. Mitt utgangspunkt fra starten av, har vært at jeg ønsket å dele oppgaven opp på tre nivåer: et individplan, et gruppeplan og et samfunnsplan. Jeg vil nå redegjøre for og presentere mine problemformuleringer ut fra denne inndelingen.

Del 1. Individ plan

Open Source Community er et miljø som har én felles nevner – utvikling av programkode. Det å utvikle programmer har miljøet til felles med mange andre softwareutviklere. *Men hva vil det si å kode?* Når jeg har undersøkt og lest og forsøkt å samle materiale om miljøet generelt, og rundt systemutviklere, så har det stadig vekk dukket opp begreper og fraser som sier noe om programmørers selvoppfattelse. Jeg tenker på begreper som 'code poets', kunstnere, kunstmaler ol. Dette fikk jeg lyst til å undersøke nærmere; hva legger folk i dette?

Tese: Programmering på individplan er en ensom prosess. Ved å betrakte programmøren som forfatter, som ved hjelp av sin total-involverende bevissthetsakt skaper en tekst, vil jeg ved hjelp av Lytje, Naur og Jakobson føre tekstbegrepet innenfor programmering videre. Selve kodningsprosessen, vil jeg ved hjelp av Bateson, sammenligne med den kunstneriske prosess - eller lek - som han beskriver. På denne måten vil jeg synliggjøre den kreativitet og innovasjon som kjennetegner systemutvikling.

Del 2. Gruppe plan

Dette specialet har systemutvikling som omdreiningspunkt. Del 2. er et forsøk på å presentere miljøet, samtidig som jeg forsøker å analysere det som først og fremst kjennetegner denne form for systemutvikling: forskjellige former for vitensdeling.

Tese: Ved å analysere utviklingen av Open Source miljøene, vil jeg vise at omfanget og kvaliteten i vitensdeling innen systemutvikling, avhenger av åpenhet i økonomi og intellektuell kapital, kompleksitet, organisering og metode, samt læring. Vygotskys medierende perspektiv, viser at menneskets evne til å ta i bruk redskaper, å lære og dermed å utvikle seg, er kulturelt betinget. Jeg vil vise at de ovenfor nevnte forskjellige former for vitensdeling, medvirker til å skape et miljø hvor det foregår læreprosesser, og dermed læring.

Del 3. Samfunns plan

I dette avsnittet vil jeg komme med en *beskrivelse* av OSC, ut fra en antropologisk tankegang. Jeg vil bruke Batesons 'schismogenese' og 'ethos' begrep som forklaringsmodell. I tillegg til eget innsamlet materiale, vil jeg bruke Raymonds antropologiske fremstilling som kilde for å underbygge min beskrivelse.

Tese: Takket være Internet, kan programmører idag oppfatte seg som deltagere i et verdenssamfunn. De kan holde sammen om forskjellige verdier, på tvers av forskjeller i økonomi, geografi, og språk skiller. Ved å beskrive den observerbare adferd samt de underliggende verdier, vil jeg avdekke hvilke verdisystemer som finnes, og om det eksisterer flere verdisystemer i Open Source kulturen.

Disse tre tesene har én ting felles: de handler alle om forskjellige former for kommunikasjon i en nettkultur. Hvis man skriver et program, eller regner dette for tekstproduksjon, så omhandler dette kommunikasjon. Forskjellige former for vitensdeling er likeledes knyttet til kommunikasjon. Ikke minst er en antropologisk tilgang til en kultur en beskrivelse av forskjellige former for kommunikasjon. All kommunikasjon i miljøet foregår ved hjelp av språket. Jeg har forsøkt å finne frem til en teori og et begrepsapparat som er hensiktsmessig i forhold til det fenomen jeg har valgt å undersøke, og som jeg har hatt ønsket å formidle noe om, og har derfor lagt hovedvekten på Gregory Batesons (1904-1980) *Steps to an Ecology of Mind*, 2000. Jeg har også hatt stor glede av Bent Ølgaards *Kommunikation og økomentale systemer ifølge Gregory Bateson*, 2001. Batesons produksjon regnes ikke for å være stor, men den er vanskelig tilgjengelig fordi den spenner så vidt. Bateson var utdannet naturhistoriker og antropolog og var opptatt av disse emner, men han skriver og uttaler seg om emner som psykologi, psykiatri, filosofi, og økologi. Batesons interesser i de forskjellige enkeltvitenskaper sirkler rundt ett problem: det mentale. Ølgaard har en helhetsoppfattelse av Batesons livsverk, og fremstiller og forklarer særlig med henblikk på hans mind begrep og hans epistemologiske problem. Ølgaard kaller Batesons teorier for en metateori, forstått på den måten at hans livsverk går ut på å finne frem til en overordnet forståelse av det mentale aspekt i verden. Bateson oppfatter det mentale eller mind, som kommunikasjon. Hans utgangspunkt er alltid det naturlige system, som består av levende organismer i sin omverden, og han anvender kommunikasjonsteorien og kybernetikkens språkbruk i sitt arbeid. Bateson (og kommunikasjonsteorien generelt)

er ikke opptatt av kommunikasjonens innhold, men av det formale, på samme måte som innenfor lingvistikken. Det jeg synes har vært fasinende i min tilegnelse av Bateson, er at han ved å fokusere på det formale, hele tiden kommer tilbake til hva kommunikasjon dreier seg om: levende veseners relasjoner til andre levende vesener og til deres omverden. Ølgaard forsøker å belyse, samt komme med en tolkning av det han betegner som Batesons epistemologiske problem.¹ Epistemologi er et filosofisk problem. Innenfor filosofien skiller man mellom ontologiske problemer, som dreier seg om hva tingene er, og epistemologiske problemer som knytter seg til hvordan vi tenker om og vet noe om tingene. Dette vil si dss. hvordan våre tankestrukturer er. Bateson kalte seg selv for naturalist, en iakttager av mennesker og dyrs adferd. I stedet for å skille ontologi og epistemologi foreslo han å bruke betegnelsen 'epistemologi' som samlebetegnelse for hele den gren av vitenskapen som undersøker hvordan enkelte organismer eller aggregater av organismer vét tenker og avgjør. (Ølgaard, s.200)

Ovenfor beskrev jeg hvordan jeg i løpet av prosessen har arbeidet meg frem til de forskjellige problemstillinger for oppgaven. Den første delen skal leses som et teoriavsnitt. Her forsøker jeg å trekke inn forskjellige teorier som underbygger min problemstilling knyttet til kreativitet. Den andre delen analyserer innsamlet materiale, på bakgrunn av problemstillingen knyttet til vitensdeling. Den tredje delen har en antropologisk problemstilling. Jeg har ikke til hensikt å redegjøre for hele Batesons begrepsapparat. Min intensjon har vært å anvende noen av Batesons begreper, fordi han med sin kommunikasjonsteori tilbyr en forklaringsmodell som jeg kan anvende for å formidle noe om de mønstrene jeg har oppdaget. I min tilgang til dette miljøet, har jeg arbeidet ut fra noen overordnede påstander. Disse er som følger:

- En ting er hva folk sier de gjør.
- En annen er hva folk tror de gjør.
- Noe annet er hva folk faktisk gjør.

Den første påstanden vil si dss. at det i en kultur som denne eksisterer en diskurs. Den har jeg forsøkt å få innsikt i ved intervju, ved å lese litteratur om miljøet, og ikke minst ved å følge med på forskjellige diskusjoner på Internet. Det gjelder for alle mennesker at vi danner oss tanker om hvordan vi konstruerer virkeligheten. Det er en forskjell på

1. I 2. utgave har Ølgaard skrevet et nytt avslutningskapittelet som omhandler epistemologi.

hva vi sier og hva vi tenker. Ved å beskrive og analysere den observerbare adferd, (hva man faktisk gjør) kan man finne frem til noen av de underliggende verdiene for en kultur; dens ethos. Det er disse underliggende verdier, som danner grunnlag for å fortelle noe om en kulturs verdisystemer. Dette er formålet med del tre. Min intensjon har ikke vært å analysere meg frem til noen slags 'fasit' om dette miljøet. Mitt utgangspunkt som humanist, er at jeg har en forståelse av teknologi som et menneskelig uttrykk, og mine problemformuleringer skal heller tolkes som nettopp det de er – en fremstilling av noen forskjellige nivåer av denne kulturen, sett med den utenforståendes øyne.

Som det fremgår av litteraturlisten bak, har innsamling av materiale til oppgaven foregått ved hjelp av Internet. Det ligger jo i sakens natur, at jeg ved å undersøke en nettkultur benytter meg av Internet. Vinteren 2002 holdt Aarhus Linux Group et foredrag med en anerkjent foredragsholder fra Open Source miljøet, Peter Salus. Dette foredraget, i likhet med andre, ble streamet og var derfor en god kilde i mitt videre arbeid. Det var i den anledning jeg innledet samtaler med Mads Toftum som var en av arrangørene og medlem av Aarhus Linux Group. Senere har jeg foretatt et lengre intervju med ham som har vært til stor hjelp. Av litteratur jeg har støttet meg mye til, i både annen og tredje del, må nevnes Eric S. Raymond *The Cathedral & the Bazaar*.

Musings on Linux and Open Source by an Accidental Revolutionary, 2001. Raymond har vært pådriver i miljøet, og med bakgrunn som antropolog, beskriver han utviklingen i det han navngir som hacker kulturen. I oppgaven har jeg valgt å bruke betegnelsen "hacker" om deltagerne i miljøet. Man kan undre seg over hvorfor deltagerne velger å bruke denne betegnelsen om seg selv. For utenforstående gir ordet konnotasjoner til datakriminalitet. Raymond velger bevisst å benytte denne termen, og det har jeg også valgt å gjøre. Del 2 vil inneholde en nærmere redegjørelse for hva de innforståtte legger i begrepet, og del 3 vil gi en presentasjon av Raymonds fremstilling.

Artikkelsamlingen *Open Sources: Voices from the Open Source Revolution*, 1999, må også nevnes. Denne inneholder en samling essays fra fremtredende aktører i miljøet, og har vært til stor hjelp.

Étt spørsmål, har gjort seg gjeldende helt fra begynnelsen av dette arbeidet: hva er det som *driver* folk til å bidra og delta i dette miljøet?

Hvis man overordnet betrakter denne oppgaven i en systemutviklingshistorie, så kan man se for seg den tidlige programmøren - matematikeren som stod i hvit frakk - oppslukt av sine algoritmer. Dengang foregikk programmering nesten direkte på hardwaren i en slags *naturlig tilstand* i stabile omgivelser. Programmering dengang var en tid hvor form og innhold var ett. Kall det gjerne lekens begynnelse.

Etterhvert gikk systemutviklingen inn i en form for *bevisst tilstand*. Form og innhold ble adskilt i en analytisk prosess. Nye generasjoner av programmeringsspråk ble utviklet og gjentagelsen og metoden gjorde at man oppdaget mønstre. Det å utvikle systemer innebar at man involverte mange forskjellige aktører, og derfor var det nødvendig å utvikle et begrepsapparat og en metode som gjorde denne prosessen forståelig. Første del av denne avhandling kan også leses som spørsmålet: hva vil det si å kode? Annen del, forteller noe om at noen programmører har oppdaget muligheten for å utfolde seg. Slik jeg ser det, vil det å leke med koden, si det samme som å eksperimentere i rommet mellom form og innhold. Utfoldelsen gir mulighet for å sette sammen ting på nye måter. Open Source Community gir uttrykk for en *ny ubevisst tilstand* i systemutviklingens historie. En av drivkreftene for dette miljøet er kan hende lengselen tilbake til den opprinnelige leken. Ved hjelp av kjente mønstre kan man arbeide med nye form & innholdskonstellasjoner. Man kan kanskje oppdage nye mønstre og nye former for lek, i mellomrommet mellom form & innhold.

Innledning

For noen år tilbake deltok jeg på et kurs som kaltes EDB-for humanister ved Universitetet i Oslo. Kurset var oppdelt; hvor den ene del het *Informasjonsteknologien kulturelle aspekter* og den andre var *Innføring i programmering/Pascal*. Dette kurset startet opp i begynnelsen av 80 – tallet, og var rettet mot humanister som f.eks. lingvister, navneforskere, historikere ol., som selv måtte programmere de programmer de behøvde for å kunne bearbeide store kildematerialer, folketellinger ol. Programmeringskurset var hardt. Vi begynte stille og rolig, men etterhvert ble vanskelighetsgraden så stor, at frustrasjonen blandt studentene ble mer og mer tydelig. Vi måtte hele tiden levere inn obligatoriske prøver for å få kurset godkjent. Det ble mange sene netter, hvor jeg satt oppe til den lyse morgen, drakk kaffe og slet med funksjoner og prosedyrer. Jeg ble fasinert av det å skrive programmer. Tenk at man kunne ta en hvilken som helst menneskelig praksis – så lenge den lot seg formalisere – for så å kunne prosessere den ved hjelp av noen små tegn, og få ut en masse annen betydning. Jeg ble aldri noen programmør, men jeg lærte noe viktig på dette kurset. Jeg fikk innblikk i hva det vil si å programmere. Noe av det jeg senere har tenkt mye på, er selve den prosessen som foregår når man skriver kode. Man sitter f.eks. med et program, som ikke fungerer. Man vet ikke hvor feilen er, men må forsøke å gå dypere og dypere ned i algoritmen, holde tråden, selv om denne tankerekken blir mer og mer kompleks. Man må holde på en variabel – gå nedover i strukturen sammen med den – gjennom de funksjoner og prosedyrere man har skapt – og så ringer telefonen! Man blir avbrutt. Midt i et langt resonnement. Og så prøver man igjen. Og igjen. For programmet virker ikke. Hvis man så lykkes i å holde på denne lange tankerekke, finner ut at: “jammen! Det er jo HER et lite > som er galt kodet” – og så virker det. Programmet kjører! Jubel og glede! Det er kanskje en grunn til at mange skriver kode om natten? En annen ting jeg har tenkt over, er hvordan man i Pascal og i programmering forøvrig, nesten skriver en tekst, eller et handlingsforløp: **if** det og det **then** det, **else** dette og hint. Denne forståelsen, at programmering ligner tekstproduksjon, ligger som et grunntema i den

videre gjennomgang. Det å skrive eller skape en tekst, er en kommunikativ handling. Dette kommer jeg tilbake til.

Først litt om systemutvikling

For en tid tilbake skrev jeg en oppgave: *Hvordan foregår vitensdeling hos Mjølner? Et forsøk på å oppdage mønstre...*, hvor et ledd av oppgaven var intervjuer med forskjellige systemutviklere i denne bedriften. Allerede i det første intervjuet, ble vi korrigert av den intervjuede. Vi titulerte ham “programmør”, og han forklarte at dette ikke var i samsvar med hans egen selvforståelse. En programmør gjør det han blir bedt om, i motsetning til en softwareutvikler – hvor utfordringen ligger i å finne ut *hvordan* en oppgave skal løses.¹ En annen av de intervjuede beskrev den estetiske dimensjonen i et godt utført stykke arbeid: “man har håndverkere eller malere...og man har kunstmalere...” Selv ville han betegne seg som en brukskunstner.

Systemutvikling er mange ting. Dette er prosesser som foregår i mange faser, og med forskjellige aktører. Å lage et system, er ikke det samme som å tenke litt på problemet, for så å sette seg ned og begynne å kode. I vår tid, handler systemutvikling om å tenke på problemet, for så å designe systemet - uten å kode - i en formalistisk notasjon. Enten det være seg metodiske fremgangsmåter som f.eks. UML (Unified Modelling Language), eller om man bruker det naturlige språket. Man begynner med gule lapper og white board; kartlegger hvilke prosesser eller menneskelige praksiser som skal formaliseres. Denne form for prosesser bærer preg av en intellektuell eller akademisk disiplin. Så begynner selve kodingen. Denne fremgangsmåte betegnes ofte som softwareingeniør-tradisjonens livssyklusmodeller. Den mest kjente er navngitt som 'waterfall modellen', fordi utviklingen går gjennom forskjellige trinn: *Kravsspesifisering, Design, Implementering, Test, Vedlikehold*. Innen denne tradisjonen ser man programmering kun som én av mange faser i utviklingen. Innen for softwareindustrien finnes en mangeårig tradisjon for hvordan utviklingsprosjekter organiseres, og arbeidet fordeles mellom forskjellige aktører, som designer, systemanalytiker, programmør, brukerkonsulent osv. Inger Lytje *Software som tekst. En teori om systemutvikling*, 2000, mener dette kan ses som en levning fra dengang programmering var det sentrale ledd

1. Selv velger jeg å ikke skille mellom disse uttrykkene, mao. definerer jeg det å være programmør som en som skriver kode.

i en systemutviklingsprosess, hvor man så systemutvikling som programmering pluss litt mer. Den Skandinaviske tradisjon satte fokus på analyse og design, som sentrale momenter, men tradisjonen har ikke problematisert eller interessert seg for programmering og for relasjonen mellom design og programmering, selv om den relasjonen er uhyre kritisk. (Lytje, s.19) Troen på at bedre programmeringsspråk, verktøyer og metoder vil løse programmeringsproblemet, synes å være meget sterk. Den tro deler Lytje ikke. Hun tror heller at vi skal interessere oss for programmering som en menneskelig aktivitet og for relasjonen mellom design og programmering.

Inger Lytje mener at det er mye som tyder på at den antagelse om det å programmere som softwaretradisjonen bygger på er feilaktig.¹ Først og fremst viser det seg at software stadig er fulle av feil. Nye versjoner og programmer lanseres og markedsføres, og selv om de på overflaten virker brukervennlige og smarte, så er den underliggende software defekt. Feil i software er frustrerende, men det kan også være fatalt og livstruende.²

Software-engineering tradisjonen, hvor det å programmere blir utskilt til en fase i prosjektet, tenderer også mot å sammenligne denne form for arbeidsprosess som tilhørende industrisamfunnet, og hvor programmøren blir underlagt den form for kontroll som kjennetegner denne form for produksjon; time-management. Det var kanskje på denne bakgrunn den intervjuede i sitatet ovenfor ikke ville kalles programmør, men systemutvikler? Lytje mener at livssyklusmodellen, og den stramme tidsstyring som preger softwarebransjen, på mange måter har smittet over på samfunnets psyke. Det har bredt seg en slags forestilling om at ting blir for gamle, før vi egentlig har hatt tid til å tilegne oss dem. Vår tid er full av uttrykk om at vi ikke må gå i stå, at vi må utvikle oss, at man skal være i stand til omstilling osv.

“Grunden til, at den følelse oppstår, er, at utvikling ensidigt forstås som utvikling af tekniske systemer, mens ibrugtagning og tilvænning til nye systemer slet ikke regnes med eller i beste fald undervurderes.” (Lytje, s.21)

-
1. Svært mange softwareutviklings prosjekter klarer ikke å holde tidsplan og budsjetter. Dessuten har historiene vært mange om regelrette fiaskoer, hvor innføring av systemer ikke har underbygget brukernes aktiviteter og forventninger, noe som har krevd store økonomiske og menneskelige ressurser.
 2. Levenson, og Turner skriver i artikkelen *An Investigation of the Therac - 25 accidents*, om hvordan en software bug var årsaken til at et stråleapparat ga massive overdoser på pasientene. “Computers, Ethics & Social Values”.

Lytje legger vekt på at man isteden kunne velge en annen synsvinkel. Erfaring viser, at systemer som først er innført, blir i bruk over lang tid. Systemene som tas i bruk, blir en integrert del av vår bevissthet og av våre daglige rutiner. Et evolusjons perspektiv på software, innebærer at man ser på systemene nærmest som levende organismer som skal pleies og vedlikeholdes gjennom hele sitt liv – til de dør. Inger Lytje ønsker å fremme en forståelse for software som en form for tekst: “Ud fra en overordnet betragtning vil jeg opfatte 'systemudvikling' som kæder af skrive- og læseprocesser, gennem hvilke softwareprodukter designes, konstrueres og evalueres. Det betyder, at jeg vil opfatte systemudvikling som en skriftlig form for kommunikation”. (Lytje, s.15)

Hvis man ser software som tekst, mener Lytje at lese og læreprosessene er del av en 'utvikling', og dette medfører en annerledes innfallsport. Skrive/lese/lære og kommunikasjonsprosesser vil alltid flette seg inn i hverandre i et mer organisk utviklingsforløp – som aldri stanser. På samme måte som tekst har også software evig liv.

Forfatterinstansen ser hun som designer-gruppen, (dvs. eksperter og systemutviklere, gjerne et team, og ikke den isolerte, ensomme forfatter!) og brukeren som leser. Lytje mener at informasjonsteknologi påvirker alle menneskelige prosesser; også den måten vi tenker om disse prosesser på og hvordan de er forankret i en kulturell kontekst. Hennes forsøk på å forstå software som tekst, er motivert ut fra et grunnprinsipp om demokrati. Det å produsere og lese tekster er grunnleggende for demokratiet. Lytje bygger på Derrida og hans språkfilosofi, og for språk som konstruksjon og dekonstruksjon, fordi dette innebærer et nytt syn på språk og viten eller mellom form og mening.

“Forskellen mellem at sige, at en form *har* mening, og en form *er* mening, er forskellen på at betragte systemudviklingsprocessen som “modellering af fænomener og begreber fra den virkelige verden” og at anlægge et socialkonstruktivistisk syn på systemudvikling, dvs. se systemudviklingsprocessen som en videnskabende proces i sig selv. Det er forskellen på at se en tekst som afbildning af en sandhed der findes på forhånd, og at se tekst som en produktiv størrelse, gennem hvilken sandhed konstrueres. [...] Altså, form kan ikke adskilles fra mening; formen er meningen.” (Lytje, s.60)

Innenfor humanistiske sammenhenger, ser man ofte med beklagelse på formaliseringsprosessene i softwareutvikling. Hvis man i utgangspunktet ser på teksten (eller systemet) som en avbildning av sannhet (formen *har* mening), så bygger denne oppfattelse på at formalisering står for entydighet, og dermed en fattiggjørelse av den avbildede virkelighet. Dette fordi man skiller mellom *uttrykk* og mening. Lytje ser formalisering annerledes, og hun følger Derridas tankegang om at en tekst ikke avbilder en dypere mening, men at den er et dynamisk system av forskjellige tegn. Formen (eller formelen) skal ikke ses som en fattig representasjon av sannhet, men som et spåklig uttrykk som skiller det som uttrykket handler om fra det det ikke handler om.

“Formen er altså i like så høy grad formens negation, og det er, siger Derrida, betingelsen for formens produktivitet.” (Lytje, s.61)

Denne tankegangen skaper avstand mellom designeren og brukeren, men i motsetning til 'participatory design', gir dette muligheter for designeren i å bruke et formalspråk og formaliseringsprosessen blir et middel til å rette sin oppmerksomhet mot noe bestemt. Når Lytje velger å se software som tekst, vil det si det samme som at designeren ikke har egenrett på en bestemt måte å tolke verket på. På samme måte som en litterær forfatter kan designeren kun vende tilbake til verket som leser, når han/hun først har gitt slipp på det.

“Inden for brugerorienterede systemudvikling bliver det ofte understreget, at man som udvikler må tage hensyn til den kontekst, der på forhånd er brugerens, og at man ikke må overskride brugerens tankegang. Dette ville dog svare til, at man ikke måtte skrive bøger, der på nogen måde kunne rokke ved læserens forestillingsverden, hvilket er absurd. Man skal være åben over for, at et softwaresystem kan skrive nye kontekster og åbne nye forestillingsverdener for brugeren.” (Lytje, s.62)

Lytje mener at ett typisk problem i forholdet bruker/leser og system, er at brukere er for dårlige lesere. Det har aldri vært noe særlig prestisje i å *tilegne* seg programmer eller systemer. Det har heller ikke ligget prestisje i å vedlikeholde dem. Prestisjen har ligget i å skape dem. For Lytje handler et godt design om å kunne lage rammer for forandring etterhvert som brukeren blir mer og mer involvert. Det å ta i bruk nye systemer endrer folks praksis, og gir nye muligheter. Designeren er ekspertene, men det forhindrer ikke at brukeren med sin ekspertise skal ta ansvar og bidra i utviklingen.

Inger Lytje bygger på Peter Naurs definisjon av systemutvikling som teoriskaping.

“Software som tekst er nært beslægtet med opfattelsen af software som teoribygning, der primært ser softwareudvikling som vidensintensive lære - og kommunikationsprocesser gennem hvilke et softwareprodukt tager form.” (Lytje, s. 21)

Peter Naur tar i artikkelen *Programming as Theory Building* (1986), et oppgjør med et utpreget formalistisk syn på programmering.¹ I motsetning til dette synet, ser Naur viten som en psykologisk realitet, som er knyttet til menneskets evne til å velge, å skille og å fokusere og å se systematiske sammenhenger osv. Han mener at det er viktig å forsøke å forstå hva programmering innebærer. Hvis vår forståelse av programmering er upassende, så vil man i utviklingsprosesser kunne misforstå de vanskelighetene som dukker opp underveis, og dermed vanskelig kunne overvinne de problemene som oppstår. Dette fører ofte til frustrasjon og konflikter, et velkjent fenomen i mange systemutviklingsprosjekter. Naur ser programmering som teori skaping: Utvikleren eller utviklergruppen, må bruke sin kunnskap og intuisjon, for å danne seg et helhetssyn, mao. danne seg en teori om hvilke prosesser et program skal kunne utføre.

“In terms of Ryle`s notion of theory, what has to be built by the programmer is a theory of how certain affairs of the world will be handled by, or supported by, a computer program.” (Naur, *Programming as Theory Building*, s.255)

I artikkelen *Intuition in Software Development* (1985), beskriver Naur hvordan man ofte i diskusjoner innen utviklermiljøer, så intuisjon som en litt lavere form for menneskelig bevissthet. Dette utpregede rasjonelle syn på hva programmering er, innebar at intuisjon og den form for viten som Naur beskriver, burde elimineres i softwareutvikling. Mennesker bruker sansene, når vi sorterer og skaper mening ut av de inntrykkene vi hele tiden blir påvirket av, og vi er som mennesker hele tiden i stand til å korrigere og modifisere våre intuitive evner gjennom livet. Den intuitive forståelsen vi har av omverden kan allikevel ikke skilles fra den forståelsen vi har av språk og de teorier vi tilegner oss, eller den bevissthet og det syn på verden vi danner oss. Lytje viser til Naur og påpeker at det stadig vekk er en utbredt fordom at når man taler om repre-

1. Naur bruker betegnelsen programmering om både design og implementering av programløsninger.

sentasjon av viten i relasjon til computeren, så må man nødvendigvis operere med viten som logikk. (Lytje, s.146) Det er ikke tilfelle. Lytje tror denne fordømmen faktisk er mest utbredt blandt humanister, som ofte har en forestilling om, og sine skrekk scenarioer om computere. Teknologer som arbeider med datamaskiner er ofte langt mindre fordomsfulle.

Lytje deler Naurs vitensteoretiske syn og sier om formaliseringsprosessen: “ Det er altså ikke nødvendigvis sådan, at man først har en viden, som derpå formaliseres og implementeres, men at formalisering og implementering bidrager til utvikling af ny viden.” (Lytje, s.156)

Det er ikke slik at man første tilegner seg den nødvendige viten, for så å gjøre denne viten tilgjengelig for systemutvikling, for så å notere den i et formelt språk og derpå programmere den i softwaresystemet. I selve formaliseringsfasen, den fasen hvor viten noteres i et formelt språk, kan det skje at man oppnår nye innsikter, fordi det ut fra et brukersynspunkt kan være mange teorier som danner utgangspunkt for formaliseringsprosessen. En aktør som skal beskrive sitt bruksunivers, i f.eks. et ekspertsystem, vil ha *sin* kulturelle kontekst og sine holdninger til sine aktiviteter.

Naur (som bygger på Ryles notion of theory) mener at kunnskap eller det at et menneske tilegner seg en teori, betyr at et menneske kan gjøre visse ting, og i tillegg kan støtte denne gjøren med forklaringer, rettferdiggjøring, og svare på spørsmål som har tilknytning til aktiviteten. Ryle beskriver intelligent adferd, ikke bare som en oppramsing av visse kunnskaper eller fakta, men evnen til å gjøre visse ting, som f.eks. å skape eller avkode vitser (humor), å snakke grammatisk riktig, å fiske ol. Det som kjennetegner intelligent adferd er at man er i stand til å utføre dette bra ut fra visse kriterier – men det som først og fremst kjennetegner intelligent adferd, er evnen til å anvende kriteriene, og på den måten være i stand til å korrigere feil, det å kunne lære fra andres eksempler osv.

“It may be noted that this notion of intelligence does not rely on any notion that the intelligent behaviour depends on the person`s following or adhering to rules, prescriptions, or methods. On the contrary, the very act of adhering to rules can be done more or less intelligently;” (Naur, *Intuition in Software Development*, s. 255)

Naur mener at alle programmer må modifiseres og vedlikeholdes, de må korrigeres og forandres på samme måte som den virkeligheten de representerer. Det er i den forstand han understreker hvor viktig det er at programmøren har dannet seg et helhets-syn, en teori, om programmet. Det som behøves i en modifisering (av et program) er først og fremst en konfrontasjon mellom eksisterende løsninger og de nye kravene man ønsker skal imøtekommes i en modifikasjon.

Det å designe, eller tegne arkitekturen av et system er én ting. Den prosessen handler om å kartlegge hva et system skal utføre ut fra en helhetstankegang, og om hvordan man kan innbygge fleksibilitet og mulighet for nye implementeringer i fremtiden. Men selve det å skulle utføre kodingen – å finne løsningene, eller hva Naur definerer som teoriskaping - kan beskrives som en annen form for prosess. Dette tolker jeg som hva Lytje kaller relasjonen mellom design og programmering. Begge disse prosesser ser hun som del av forfatterinstansen. På bakgrunn av Lytjes fremstilling av software som tekst, og på bakgrunn i Naurs oppfattelse av hvordan intuisjonen er del av denne kreative prosess, vil jeg i de neste avsnitt føre tekstbegrepet innenfor programmering videre.

Kreativitet, metaspråk og metakommunikasjon

Det å skrive programmer, innebærer at man skal kunne finne nye løsninger på problemer. Det er en kreativ prosess. Når jeg har undersøkt og lest og forsøkt å samle materiale innen programmør-kulturen, har det stadig vekket dukket opp begreper og fraser som sier noe om programmørers selvpoppfattelse. Man benytter ord som kunstmaler, kunstnere ol. Innen denne kulturen eksisterer f.eks. betegnelsen *Code poet*. Denne beskriver en systemutvikler som skriver vakker, elegant kode. Det er ikke hvem som helst som kan ta i bruk denne ærefulle betegnelsen, som man i tilfelle blir tildelt av de andre. På bakgrunn av mange programmørers sammenligning mellom det å skrive kode og kunstnerisk aktivitet, samt Lytjes forståelse av software som tekst, vil jeg forsøke å trenge dypere inn i hva denne kreative prosessen består av. Roman Jakobsens kan fortelle noe om hva som gjør språk poetisk; han uttaler seg om kreativitet innenfor språkets rammer. Bateson kan fortelle noe om selve den kreative prosessen som kjennetegner all kunstnerisk aktivitet. Men først er det nødvendig med litt begrepsavklaring.

Roman Jacobsen har innen retningene formalisme og strukturalisme undersøkt *litterariteten* som en egen type språkbruk, hvor arbeidet har gått ut på å undersøke grunnbegreper i retorikk i forhold til moderne lingvistiske teorier. Roman Jakobson har i sin dobbeltsrolle som lingvist og litteraturforsker, arbeidet med to aspekter av språket: metaforen og metonymien. Den ene av språkets aspekter kan betegnes som horisontal – konstituensdelen – eller det metonymie aspekt. Dette er kombinasjonsaksen, hvor setninger ord, fonemer – viser til konteksten. Språkets andre aspekt kan betegnes som vertikal - alternasjonen – eller det metaforiske aspekt. Dette er seleksjonsaksen, det at vi kan velge forskjellige ord, og bruke vårt forråd av uttrykk. Jakobson viser til at vi trenger begge disse måtene å bruke språket på for at det fungerer. Jakobson mener at språket har seks forskjellige funksjoner, hvor den ene av dem er den poetiske. Ved først å avdekke de to aspektene av språket, viser han at når man projiserer likhetsprinsippet fra seleksjonsaksen over på kombinasjonsaksen - så er det den poetiske funksjon som dominerer. Dette kommer jeg tilbake til.

Innenfor symbolsk logikk snakker man om et objektspråk og et metaspråk. “Som Carnap siger: “for at tale *om* et hvilket som helst *objektspråk*, så har vi brug for et *metaspråk*.” (Jakobson, 1995, s.55) Vi bruker metaspråk i dagliglivet, ofte uten å være bevisst: “Hva mener du? Forstår du hva jeg sier?” osv. Det som kjennetegner metaspråk er at det retter seg mot selve koden.

Man har også innført begrepet metaspråk innen systemutvikling. Når man snakker om metaspråk i denne sammenheng, så handler det om data om dataene. Dataene inneholder data om seg selv, slik at f.eks. et bank system får et input fra en kunde og metadata forteller om hvordan disse data skal prosesseres. F.eks. vil metadata fortelle om systemet selv skal svare en kunde, eller om disse data kanskje tilhører en annen prosess – f.eks. en bank ansatt. XML er et språk, som gir mulighet for å lagre informasjon i et gjennomført metadata språk. Man skiller det semantiske innhold, fra utseende, fra prosessering. Koden kommenterer koden.

Både Jakobsons og systemutviklingens forståelse av metaspråk, faller inn under det begrepet Bateson kaller metalingvistikk, hvor språket kommenterer seg selv. Bateson mener at språket har flere abstraksjonsnivåer, og han skiller mellom metalingvistikk og metakommunikasjon (Bateson, s.178). Han viser til Rusell og Whiteheads teori om logiske typer. Grovt forenklet kan man si at denne teori beskriver det matematiske

forholdet mellom en klasse og dens elementer, og budskapet er at det finnes en diskontinuitet mellom en klasse og dens elementer. En klasse kan ikke være et element i seg selv, fordi det uttrykket som brukes om klassen, er på et annet abstraksjonsnivå eller av en annen logisk type enn de uttrykk som brukes om elementene. (Ølgaard, s.56)

Rusell påpekte at hvis man blander abstraksjonsnivåene sammen, så oppstår kategori-feiltagelser, eller paradokser. Bateson var ikke særlig interessert i metalingvistikk. Det som opptok ham, var metakommunikasjon hvor diskursen er rettet mot relasjonen mellom de som kommuniserer. Det at pattedyr kan kommunisere et utsagn som f.eks: "dette som vi foretar oss nå, er lek!" viser at vi klassifiserer det vi foretar oss, uttrykket er et meta uttrykk. Denne forståelse av språkets forskjellige abstraksjonsnivåer ligger til grunn for Batesons kommunikasjonsteori. Han beskriver hvordan vår metakommunikasjon implisitt er tilstede i all vår interaksjon med omverden og i relasjoner til andre.

I de neste avsnitt vil jeg med bakgrunn i Batesons og Ølgaards fremstillinger gjøre rede for kommunikasjonsteorien, herunder 'schismogenese' og 'ethos' begrepet. Jeg velger å gi en samlet fremstilling av Bateson i denne første delen, selv om jeg først i tredje del vil anvende f.eks. 'scismogenese' og 'ethos' begrepene.

Gjennomgangen av Gregory Batesons fremstilling av kunstens epistemologi, skal leses som en beskrivelse av den kreative prosess – eller leken. Bateson beskriver den kodingen som kunstnere bedriver, og redegjør også ut fra en systemisk tankegang, for de ubevisste mentale prosesser som kjennetegner all menneskelig, men også kunstnerisk aktivitet. Noe av det han legger vekt på er *skills* – selve håndverket. En kunstmaler kan bli sikker på hånden, og vite på hvilken måte vinkelen på penselen skal ligge for å oppnå en viss effekt. Eller hun kan utvikle og lære hvilke farger som uttrykker følelser, at blått er en kald farve, rød er en varm ol. Dette blir etterhvert en intuitiv del av prosessen og ikke en bevisst tanke. (Jmf. Naur.) Sammenlignet kan man påstå at hvis man gjennom gjentagelse og gjentagelse, blir en god programmerer, så vil etterhvert denne kunnen bli mer og mer ubevisst – en intuitiv del av bevisstheten. Man kan male en stol på 1000 forskjellige måter, og man kan skrive et program på 1000 forskjellige måter. Kreativitet og nyskapelse ligger som en mulighet i ethvert menneske, men forutsetningen for denne kreative prosess er gjentagelsen, eller håndlaget. I følge Bateson innebærer kunstnerisk aktivitet at man implisitt (eller intuitivt) transformerer infor-

masjon gjennom selve utvelgelsen av koden. I det følgende vil Batesons definisjon av metakommunikasjon bli utdypet, den form for kommunikasjon som hele tiden kommenterer hvilke lag eller nivåer våre budskaper har mening på.

Kommunikasjonsteori

Gregory Bateson forsøker i artikkelen *Style, Grace and Information in Primitive Art* (Bateson, s.128) å kartlegge en teori som har tilknytning til kultur og ikke-verbal kunst. Bateson forsøker i denne artikkelen å lage et konseptuelt rammeverk for å kunne analysere kunstobjekter – en kunstens epistemologi. Først og fremst mener han at et kunstverk har i seg interne strukturer, eller mønster, men at de samtidig er del av et større hele; den kulturen de er en del av. Dernest gjennomgår han hvordan all kommunikasjon a) handler om menneskelige relasjoner og b) at språket er lagdelt i hierarkiske strukturer, både når det gjelder verbal og ikonisk kommunikasjon. Bateson slår fast, at man ikke kan beskrive en kunstens epistemologi, uten å beskrive de forskjellige nivåer i mind. Bateson bygger på forskjellige vitenskapelige tilganger i synet på disse nivåene – eller lagene – som man også kan karakterisere som “det bevisste” vs. “det ubevisste”. Mennesket er opptatt av *hva* (det bevisste) det oppfatter, men behøver ikke å vite *hvordan* det oppfatter (det ubevisste). Bateson mener at kunsten blir et slags interface mellom det bevisste og det ubevisste.

Bent Ølgaard legger vekt på at Bateson anvender kommunikasjonsteoriens og kybernetikkens språkbruk, og at kommunikasjonsteoriens verden, er en måte å tenke, forstå, beskrive og tale om verden på, som adskiller seg fra det Bateson betegnet som *den newtonske verden*. (Ølgaard, s.44) Den newtonske verden beskriver den materielle verden, masse, utstrekning, hastighet og energi. I kommunikasjonens verden beskrives budskaper, informasjon, idèer, og aggregater av informasjon, budskaper og idèer. “A “bit” of information is definable as a difference which makes a difference. Such a difference, as it travels and undergoes successive transformation in a circuit, is an elementary idea.” (Bateson, s.315) I den klassiske fysikk betrakter man virkninger som er forårsaket av støt, slag, krefter og energi, mens kommunikasjonsteoriens virkninger forårsakes av forskjeller, informasjon eller budskaper. Ølgaard mener at det kan være problematisk å bruke begrepet *virkning* på fysikken vs. kommunikasjonens verden, fordi det gir assosiasjoner til begreper som kraft og energi. Allikevel kan man ikke

unngå begrepet, så lenge man er seg bevisst at det innebærer noe annet enn den newtonske tenkemåte.

Enhver kommunikasjonskanal består av: en *avsender* – som *koder* et budskap – via en *kanal* – hvor budskapet så må *avkodes* – av *mottager*. “I ethvert kommunikasjonssystem sker der stadig en omkodning eller transformering af de budskaber, der er i kredsløb, og forskellige slags kodning giver forskellige kommunikationsmåder, hvoraf de vigtigste er den ikoniske, den analoge og den digitale kommunikations- eller kodningsmåde.” (Ølgaard, s.49)

Den *ikoniske kommunikasjon* kalles også billedkommunikasjon, som f.eks. et kart over landskapet, en hund som reiser bust, eller f.eks. en smiley i en e-mail. Den *analoge kommunikasjon* viser til at det finnes en eller annen overenstemmelse mellom tegnet og det betegne. Det talte språk inneholder mye analog kommunikasjon, også kalt paralingvistiske trekk, som f.eks. pauser, trykk, prosodi ol. – som jo ofte er del av den ikke-verbale adferden. Analog kommunikasjon er implisitt, på den måten at vi ikke kommuniserer den med ord, men non-verbalt. Hvis et hyl uttrykker smerte, vil et kraftigere hyl bety mer smerte. Ofte er denne form for kommunikasjon lett å avkode. Dyr kommuniserer først og fremst analogt, både med mennesker og med hverandre. Jeg er av den oppfatning at mange paralingvistiske trekk, som kjennetegner det talte språk, er å finne i nettkulturer. Jeg tenker på bruken av tegn i dialoger på nettet, som ofte blir blanding av talt og skriftlig språk; WOW! uttrykker noe mer enn wow. Den *digitale kommunikasjon* står i motsetning til den analoge. Tegnet er her helt arbitrært, og har ingen likhet med det som avbildes. Her gjelder det ikke som i den analoge, at jo mer av et tegn, jo mer av budskapet. Ølgaard illustrere dette ved å vise til ordet 'ni'. Det er mindre enn ordet 'fire' men 9 er som vi vet større enn 4. Denne kommunikasjonsform innbefatter f.eks. det verbale språk, notasjonssystemet, tall osv. Ølgaard mener at retorikkens stilfigurere som metonymer og metaforer er underinndelinger av den digitale kommunikasjonsmåte, men at metaforer og lignelser også er en blanding av ikonisk og digital kommunikasjon. (Ølgaard, s.51) En siste kommunikasjonsmåte for transformering av budskap, kaller Bateson for den *Ostensive* eller påpekende kommunikasjon. Hvis jeg f.eks. peker på en katt og sier til et barn; “se sånn ser en katt ut”, eller hvis man skal lære et programmeringsspråk, så må man ha eksempler å orientere seg etter.

Som jeg var inne på i hovedinnledningen foregår all kommunikasjon i Open Source miljøene ved hjelp av språket i alle dens avskygninger. Man skaper tekster/programmer, man bruker newsgrupper, IRK kanaler og mail-lister som kommunikasjonskanaler, og man holder til, lærer og “møtes” via forskjellige portaler på Internet. På den måten kan man påstå at kommunikasjonen foregår ved hjelp av alle de ovenfor nevnte kodningsmåter.

Batesons 'schismogenese' og 'ethos' begrep.

Ofte er det slik, at det enkleste er det som er vanskeligst å forklare. Dette avsnittet; hvor jeg skal redegjøre for Batesons 'schismogenese' og 'ethos' begrep, har gitt meg mye hodebry. Disse begrepene ble mitt første “møte” med Bateson; for meg utgjorde denne måten å betrakte menneskelige adferd og relasjoner på et gjennombrudd – plutselig oppdaget jeg en slags struktur, noen mønstre, som man kunne eskalere opp og ned i forhold til hva man betraktet, enten det var interaksjonen mellom enkeltmennesker, eller mellom pattedyr, eller kulturer eller nasjonalstater. Denne oppgaven er ikke ment å være en parafrase over hele Batesons teori og begrepsapparat. Jeg har ønsket å trekke inn de konseptene jeg har ment har vært hensiktsmessig i forhold til det fenomen jeg har ønsket å undersøke, for så å anvende dem ut fra den forståelse jeg har tilegnet meg av dem. Allikevel er det vanskelig å kunne redegjøre for disse begrepene uten å trekke inn andre vitale deler av Batesons arbeid, fordi hele hans teoriapparat henger sammen og bygger på hverandre. Men uten å gå inn på en lengre fremstilling av hans læreteorier, eller kontekst begreper, eller double-bind teorier, vil jeg forsøke å redegjøre for 'ethos' og 'schismogenese'.

Gregory Bateson ønsket å finne ut av hvordan, særlig mennesker, la seg til en bestemt handlingsstil. Hvorfor utvikler man en bestemt personlighet, karakter eller adferd, som gjør at vi mennesker, eller grupper av mennesker lærer bestemte måter å anskue verden på, f.eks. bestemte kulturelle mønstre. (Ølgaard, s. 94) I artikkelen *Bali: The Value system of a Steady State* (Bateson, s.107), beskriver Bateson sine antropologiske undersøkelser av et lite samfunn på Bali, hvor han (i samarbeid med Margaret Mead) analyserer menneskelig adferd. Det var i denne artikkelen han lanserte sin egen betydning av begrepene 'ethos' og 'schismogenese'. Som jeg var inne på innledningsvis, ønsket Bateson å skape en slags meta-teori, en epistemologi eller et konseptuelt rammeverk for sosialvitenskapen, etter samme vitenskapsteoretiske idealer som den

newtonske verden. I artikkelen beskriver han hvordan han formulerte 'ethos' begrepet under sine antropologiske undersøkelser av Iatmul folket:

“The most general conclusion I could draw was of this order: that my own mental processes had certain characteristics; that the sayings, actions, and organization of the Iatmul had certain characteristics; and that the abstraction, 'ethos' performed some role-catalytic, perhaps – in easing the relation between these two specificities, my mind and the data which I myself had collected.” (Bateson, s.108)

I kraft av det å være forsker og utenforstående til en annen kultur, så kan man tolke dette dithen at Bateson forsøkte å abstrahere, å finne noen strukturer som var felles for ham/eller hans kultur og den fremmede kulturen han ønsket å undersøke. Man kan sammenligne dette med inndelingen av forskjellige former for epistemologi som Ølgaard foreslår på bakgrunn av Batesons livsverk. (Ølgaard, s.236) Først er det *den ubevisste* epistemologi. Med dette forstås den kunnskap som et individ får ved å ferdes i sin omverden, det man må vite for å kunne tenke, avgjøre, ferdes og klare seg i sine omgivelser. Dernest er det den *lokale* epistemologi, som gjelder for en samling av individer. En gruppe eller en kultur, må ha noen bestemte måter å tenke på, å vite eller å avgjøre på. Den *vitenskap* som forsøker å forstå, eller forsøker å tilegne seg kunnskap om hvordan andre levende vesener tilegner seg kunnskap, ved hjelp av forskerens målrettede empiriske undersøkelser, kaller Ølgaard den *eksperimentelle* eller *empiriske* epistemologi. “Denne empiriske epistemologi er deskriptiv, dvs. den ønsker at beskrive, hvordan idéer, tanker, avgjørelser, viden rent faktisk og typisk virker i den ydre verden, f.eks. hos skizofrene, hos en innfødt stamme på Ny Guinea eller lignende.” (ibid.) Ølgaard skisserer opp to andre former for epistemologi. Med formuleringen *epistemologi som videnskapsteori*, mener han: “Nogle prinsipper for, hvordan man driver empirisk epistemologi på videnskabelig måte inden for disse felter.” (ibid.) Videre kaller han *Epistemologi med stort E*, for det han beskriver som en normativ, filosofisk disiplin, hvor man forsøker å formulere regelmessigheter eller regler av samme generelle gyldighet og av samme abstrakte karakter som termodynamikkens andre lov eller matematikkens algebra. På mange måter rommer Batesons 'schismogenese' og 'ethos' begrep - og hans beskrivelser av hvordan han kom frem til disse - alle de former for epistemologi som Ølgaard forsøker å synliggjøre. Sitatet ovenfor viser at Bateson er seg sin egen rolle som forsker bevisst, dessuten har han bevissthet om at han vil

undersøke en annen lokal epistemologi. Han forsøker likeledes å finne noen forklaringsmodeller samt noen abstrakte begreper som dekker all menneskelig adferd, og dermed bidrar han til de to sistnevnte former for epistemologi.

Med 'ethos' begrepet forsøker Bateson å formidle eller fange opp, den grunnverdi som ligger i en kultur. En slags taus kunnskap/mekanisme som ligger i bunnen av et folks kultur. Hvilke underliggende verdier er det som gjør at en kultur utvikler seg slik den gjør? Bateson mener at de "valg" en kultur velger, grunnes i dens 'ethos'. Bateson og Mead hadde også lagt merke til at menneskelig kommunikasjon ofte har en tendens til å utvikle seg i en cumulatív adferd. Han utdyper 'schismogenese' begrepet ved å anvende systemteorien, samt feed-back begrepet, som han har hentet fra kybernetikken.

Systemteorien ble grunnlagt av Bertalanffy i tiden rundt annen verdenskrig - (på omtrent samme tid som kybernetikken utviklet feed-back begrepet som Bateson også tok i bruk). Karakteristisk for systemteorien er at alt – fra atomer til universet – blir betraktet som et system, dvs. en organisert helhet. "Mens altså naturvidenskaber og biologiske videnskaber traditionelt har brugt den fremgangsmåde at nedbryde et system i enkeltdele, vil systemteorien undersøge systemet som en helhed, uden at bryte det ned." (Ølgaard, s.31)

Kybernetikken, med Wiener og Bigelow, hadde satt ord på en av de viktigste betingelser for voluntær adferd, forstått på den måten at hvis feed-back ikke er mulig, så er voluntær adferd umulig.

"For at samle en blyant op må nervesystemet hele tiden have rapport om, hvor stor forskellen mellem fingrene og blyanten er. Et kontrolcenter af én eller anden slags må modtage besked om, hvor fingrene befinder sig i forhold til blyanten. Denne besked (altså kommunikation) kan være visuel, hvis vi kan se blyanten og fingrene, eller kinæstetisk, hvis vi kan føle blyanten, og hvis den kinæstetiske og visuelle sansing mangler, kan vi ikke ramme blyanten." (Ølgaard, s.32)

De som arbeidet med teorien fikk bekreftet at nervesystemet inneholder feed-back mekanismer, hvor det ikke bare er slik at sentralnervesystemet mottar signaler fra sansene og gir dem videre til musklene, men at noen av de mest fundamentale aktiviteter i nervesystemet er *sirkulære prosesser*. De utgår og vender tilbake til sentralnerve-

systemet via sanseorganene. Bateson utviklet 'schismogenese' begrepet ut fra hva som skjer i feed-back loopen i interaksjonen mellom individ/klasser av individer, og omverden, ut fra en systemisk tankegang om menneskelig adferd.

Bateson opererer med to hovedformer av 'schismogenese': symmetrisk vs. komplementær. Den *symmetriske* kjennetegnes ved at A og B har lignende adferd. A kommuniserer noe som stimulerer og forsterker B sin adferd, som kommuniserer og stimulerer mer av A sin adferd osv. osv. inntil et slags klimaks. Konkurransen, krangel, slagsmål, våpenkappløp, ol. er eksempler på denne type relasjon. *Komplementær*, betegner relasjoner hvor adferden er forskjellig, men allikevel passer sammen, og hvor relasjonen kan utvikle seg til en ensidig forvridning av partene, f.eks. A er dominerende og B tilpasser seg/trekker seg. Tilskuer – artist, dominans – underkastelse, omsorg – avhengighet er typisk for denne type relasjoner. Hvis adferdselementene er forbundet slik at mere av A stimulerer mer av B's tilpassede adferd, så kalles den komplementær. Bateson oppdaget at begge av disse mønstrene kunne eskalere og bli skadelige for partene, slik f.eks. et våpenkappløp eller sado-masochisme kan utvikle seg, men at elementer fra den ene relasjon kunne dempe den andre, og føre til en slags likevekt. (Ølgaard, s.65) Ølgaard illustrerer dette med et hundeslagsmål. Selve slosskampen er en symmetrisk relasjon, bitt fra den ene fører til mer bitt fra den andre. Men så kan plutselig den ene hunden legge seg på rygg og blotte strupen. Dette er et komplementært mønster som vanligvis fører til at slosskampen stopper.

Som jeg tidligere var inne på, er min intensjon (i del tre) å beskrive forskjellige former for observerbar adferd i OS miljøene, og forsøke å oppdage relasjoner som enten er symmetriske eller komplementære, for på den måten å forsøke å tilegne meg en forståelse for hvilke underliggende strukturer, eller 'ethos', som ligger som en drivkraft for denne kulturen og som dermed utgjør dette miljøets verdisystem(er).

Kunstens epistemologi

Bateson introduserer i artikkelen *Style, Grace and Information in Primitive Art*, (s.128) med å referere til Aldous Huxley, som engang skal ha sagt at menneskehetens problem er dens søken etter *grace*. I likhet med Walt Whitman, så argumenterte Huxley for at dyr har en umiddelbar måte å kommunisere på, som har en naivitet og enkelhet som mennesket har mistet. Menneskets adferd er korrumpert av svik, inkludert selv-

bedrag, og selv-bevissthet. Huxley mente at mennesket har mistet den "grace" som dyrene (og Gud) fortsatt er i besittelse av. Batesons argument i denne artikkelen bygger videre på denne definisjonen og han fremsetter derfor:

"I argue that art is a part of man's quest for grace; sometimes his ecstasy in partial success, sometimes his rage and agony at failure.

I argue also that there are many species of grace within the major genus; and also that there are many kinds of failure and frustration and departure from grace. No doubt each culture has its characteristic species of grace toward which its artist strive, and its own species of failure." (ibid.)

Bateson argumenterer for at søken etter grace, enten den lykkes eller mislykkes, er representert i alle kulturer gjennom kunstverk. Videre argumenterer Bateson for at problemer knyttet til grace, har å gjøre med integrasjon mellom de forskjellige delene av mind – og i særdeleshet de forskjellige lag i mind som gjør det vi vanligvis betegner som "det bevisste" og "det ubevisste". Et sentralt spørsmål for Bateson er: i hvilken form er informasjon om psykisk integrasjon oppbevart eller kodet i et kunstverk? Karakteristisk for mye av det Bateson har skrevet, er at han er opptatt av den ikke-verbale kommunikasjonen, og slik også her. Når han går løs på kunstanalyse, er det ikke for å analysere "historien" bak verket, eller hva det som sådan representerer, men hva som implisitt ligger i selve formen. (Jmf. Derrida og Lytje.) Men representasjonalisme som sådan finner han interessant. "It is the very rules of transformation that are of interest to me-not the message, but the code." (Bateson, s.130) Hvorfor er denne koden valgt? Hva slags mening ligger i å velge denne koden? Bateson har en definisjon av "mening" sett på den måten at hvis man har en samling av hendelser eller objekter, enten det være seg et bilde, en frosk, en kultur eller en samling fonemer – så kan denne samling inneholde redundans eller mønstre. (Redundans > i betydningen overflod/overflødig, særlig at det i språklig el. annen overføring av informasjon brukes en uttryksform som gir flere opplysninger om innholdet enn det minimum som trengs for å oppfatte det.) Hvis vi kan oppfatte denne samling av f.eks. fonemer og sette et "slash mark" etter den, så vil vi være istand til gjette hva slags mening som vil være på den andre siden av slash marken. Mao. det som finnes på den ene siden av slash marken, har informasjon eller mening om hva som er på den andre siden. En ingeniør vil kalle dette redundans, en kybernetiker vil mene at informasjonene på den

ene side vil tilbakeholde eller redusere mulighetene for feiltolkning på den andre siden. “It is, I believe, of prime importance to have a conceptual system which will force us to see the “message” (e.g, the art object) as *both* internally patterned *and* itself a part of a larger patterned universe- the culture or some part of it.” (Bateson, s.132) Dette forsøker Bateson å illustrere ved hjelp av et diagram:

[Characteristic of art object/Characteristics of rest of culture]

Rammeparantesene avgrensner relevansen og forbindelsen utgjøres av slashen. Hva slags relasjon eller korrespondanse er det som krysser denne forbindelsen? Bateson illustrerer dette med et eksempel. Hvis jeg sier til deg: “det regner” og du antar at hvis du så ut av vinduet, så ville du få øye på regndråper.

[Characteristic of “It’s raining”/Perception of raindrops]

Dette er ikke så enkelt som det i første omgang kan virke. Forutsetningen er at vi snakker det samme språk, samt at jeg stoler på din sannferdighet. Faktum er, at de fleste mennesker ikke kan la vær å kikke ut av vinduet; vi dupliserer informasjonen, “We like to prove that our guesses are right, and that our friends are honest. [...] *we like to test or verify the correctness of our view of our relationships to others.*” (Ibid.) Hvis vi tror et annet menneske er sannferdig – og det viser seg at dette mennesket ikke er sannferdig – så får vi det dårlig. På samme måten som hvis vi mistenker et annet menneske for ikke å være sannferdig, når det i virkeligheten er det – så får vi det også dårlig. Dette legger Bateson stor vekt på, fordi det minner oss om alle kommunikasjons systemer. Enten er det samsvar eller ikke-samsvar mellom partene i et hele, og dette gir informasjon om et ennå større hele. Dette kan illustreres slik:

[(“It’s raining”/raindrops)/you-me relationship]

Her vil redundansen finnes mellom slashen i de runde parentesene, mens det samtidig gir informasjon om redundansen i informasjonen som er avgrenset av de hårde parenteser. Utsagnet “det regner” er i seg selv konvensjonelt kodet og strukturert. På samme måte som regnet er strukturert. Ved å se på en regndråpe som faller til jorden, kan jeg forutsi hvordan de andre vil falle. Men forbindelsene på tvers av det verbale utsagnet korresponderer ikke uten videre med forbindelsene på tvers av regndråpene, eller regndråpenes struktur. Hvis jeg istedenfor et verbalt utsagn, hadde gitt deg et bilde av regnet, så ville det vært større samsvar mellom strukturen på bildet og regn-

været. Denne forskjellen illustrerer den vilkårlige og digitale koding som er karakteristisk for den verbale del av språket, i motsetning til den ikoniske kodingen som en billedlig fremstilling innebærer. Den måten vi kommuniserer på, innebærer mao. en hierarkisk struktur; på samme tid digital eller verbal på ett nivå og ikonisk på et annet.

Bateson kommer inn på dette med nivåer eller “levels”. For det første har han slått fast at kombinasjonen av meldingen “det regner” i tillegg til persepsjonen av regndråper, i seg selv kan konstituere en beskjed om forbindelsen i en personlig relasjon. Videre; at når vi skifter fokus fra mindre til større deler av “meldinger”, så kan vi oppdage at de større enhetene har i seg ikonisk koding selv om de utgjøres av verbale mindre enheter. Den verbale beskrivelsen er ofte ikonisk i en større struktur. En vitenskapsmann som skal beskrive en meitemark kan f.eks. starte ved hodet og jobbe seg lenger og lenger nedover – og på den måten lage en beskrivelse som ved sin sekvensiering og forlengelse er ikonisk eller billedlig. Kunstens epistemologi kan derfor ikke beskrives uten en nærmere beskrivelse av nivåer ifølge Bateson: “The word “know” is not merely ambiguous in covering both *connaître* (to know through the senses, to recognize or perceive) and *savoir* (to know in the mind), but varies – actively shifts- in meaning for basic systemic reasons.”(Bateson, s. 134.) Det vi vet gjennom sansene, kan bli viten i mind. Hvis jeg sier; “Jeg kan veien til Cambridge”, så kan det bety flere ting. Det kan bety at jeg har studert kartet og kan gi deg instruksjer, eller det kan bety at jeg kan rekonstruere detaljer langs ruten. Det kan bety at jeg kan gjenkjenne detaljer langs ruten, selv om jeg kun husker klart noen få. Det kan bety at jeg handler etter vanen fordi jeg kjenner veien, og svinger de rette stedene uten å engang måtte tenke på hvor jeg skal kjøre osv. Dette i seg selv er redundans eller mønstre som i seg selv er komplekse.

[("I know..."/my mind)//the road]

Vanskeligheten består i å finne ut av mønstrene eller strukturene innenfor de runde parentesene, eller hvilke deler av mind er redundant med denne meldingen om å vite.

En [annen] form for “viten” kan ses mer som *tilpasning* enn informasjon. En hai har en vakker form som er tilpasset bevegelse i vannet, men det betyr jo ikke at haiens arveanlegg inneholder informasjon om hydrodynamikk. Antagelig vil arveanlegget inneholde informasjon eller instruksjon som utfyller hydrodynamikk; det som hydrodynamikk trenger, har blitt utviklet i haiens arveanlegg. På samme måte som

med en trekkfugl, så behøver den ikke å vite veien til Cambridge eller andre destinasjoner, slik det er beskrevet tidligere, men fuglen kan være i besittelse av utfyllende informasjon som gjør den i stand til å fly i riktig retning. "The heart has its reasons which the reason does not at all perceive." (Bateson, s.134) Det er disse komplekse nivåene – eller lagene – mellom det bevisste og det ubevisste som skaper vanskeligheter når vi forsøker å diskutere kunst, ritualer eller mytologi. En vitenskapelig tilgang til kunst må ifølge Bateson inneholde en utredning om andre vitenskapelige tilganger i forhold til dette med forskjellige nivåer eller lag i mind. Han nevner fire av slagsen.

1) Samuel Butler insisterte på at, jo mer en organisme "vet" noe, jo mindre bevisst blir den om denne viten. Underforstått; det foregår en prosess hvor viten/kunnskap, (eller vane – enten som handling, persepsjon eller tanke) synker ned i dypere og dypere lag av bevisstheten. Dette fenomen (som også er sentralt for Zen Buddhisme) er relevant for all kunst og all ferdighet.

2) Adalbert Ames' demonstrasjon av at den tredimensjonale visuelle måten vi ser på, fungerer på bakgrunn av matematiske prosesser som involverer premisser for perspektiv ol. som vi er totalt ubevisste om. Vi har ingen kontroll over disse prosessene.

3) Den Freudianske (Fenichels) teori om at drømmer er metaforer som er kodet i samsvar med primær prosessene. (I Freudiansk terminologi er det vanlig å si at det ubevisste er strukturert i termer av primær prosesser, mens bevisstheten (særlig verbale tanker) er uttrykt i sekundær prosesser.) Bateson betrakter stil: netthet, dristighet i kontrast osv. som metaforisk. Denne koden er derfor knyttet til de deler av mind hvor primær prosessene holder til.

4) Det Freudianske synet på det ubevisste, innebærer et syn på det ubevisste som kjelleren eller skapet, hvor de vonde og skremmende opplevelser er lagret gjennom en overdragelsesprosess preget av undertrykkelse og fortrenghing. Klassisk Freudiansk teori, går ut fra at drømmer er et sekundært produkt, kreert ved "drømme arbeid". Materiale som var uakseptabelt for den bevisste tanke, ble således oversatt til et metaforisk idiom, bestående av primær prosesser. Dette for å forhindre den drømmende i å våkne. Dette kan være riktig for de deler av informasjonen som er holdt nede/undertrykt i det ubevisste. Men Bateson påpeker, at mange av våre tanker er utilgjengelige for bevisstheten, inkludert mesteparten av pattedyrs interaksjon. Derfor vil det ifølge

ham være riktig å tenke på disse enheter som noe som først og fremst – primært-eksisterer i den del av idiom som består av primær prosesser, og som også med store vanskeligheter kan oversettes til “rasjonelle” termer. Mao. snur Bateson tidlig Freudiansk teori på hodet. Dengang betraktet man det bevisste som det normale, og det ubevisste var det mystiske som trengte å forklares. Isteden betrakter Bateson det bevisste som det mystiske. Det ubevisste er derimot den metodisk beregnende ubevissthet - den hele tiden aktive, nødvendige altomfattende primære prosess. Dette mener han man må forholde seg til, hvis man skal fremsette en teori om kunst eller poesi.

“Allegory, at best a distasteful sort of art, is an inversion of the normal creativ process. Typically an abstract relation, e.g., between thruth and justice, is first conceived in rational terms. The relationship is then metaphorized and dolled up to look like a product of primary process”. (Bateson, s. 136)

Bateson slår fast, at den bevisste organismen (av pragmatiske årsaker) ikke trenger å vite *hvordan* den mottar – men bare trenger å vite *hva* den mottar. På samme måte som et TV apparat. Vi behøver ikke å vite alt om de forskjellige komponenters funksjoner. Det vi er interessert i er skjermbildet og hva vi får ut av innholdet der.

“In truth, our life is such that its unconscious componenets are continuously present in all their multiple forms. It follows that in our relationships we continuously exchange messages about these unconscious materials, and it becomes important also to exchange metamessages by which we tell each other what order and species of unconsciousness (or consciousness) attaches to our messages.” (Bateson, s. 137)

På den ene siden er det bevisstheten som for Bateson er det mystiske. Et bevisst og frivillig utsagn, kan være fult av løgn. Jeg kan si “jeg elsker deg” uten å mene det. Men vår erfaring og tolkning av de fysiske signaler, vil vanligvis gi oss en pekepinn om sannhetsgehalten i det verbale utsagnet. På den måten er de ubevisste komponenter i våre liv, hele tiden representert i alle mulige former.

Bateson sammenligner dette med *skill*; dette med håndlag indikerer eksistensen av hvor stor rolle de ubevisste komponenter spiller i selve utførelsen av kunstverket. Bateson mener at kunst blir en måte å kommunisere om de forskjellige slag eller arter av det ubevisste. Kunsten blir et slags interface mellom det bevisste og det ubevisste.

“Art becomes, in this sense, an exercise in communicating about species of unconsciousness. Or, if you prefer it, a sort of play behavior whose function is, amongst other things, to practice and make more perfect communication of this kind.” (ibid.)

Dette med dyktighet i utførelsen – er et dilemma for kunstneren. På den ene siden kreves godt håndtverk, og man blir dyktig ved gjentagelse og gjentagelse. Men som tidligere ble nevnt, jo dyktigere man blir, jo mer vanemessig blir utførelsen, og dermed synker bevisstheten om hvordan man utfører, lenger og lenger ned i bevisstheten.

Hvis en kunstner forsøker å kommunisere rundt de ubevisste komponentene i utførelsen, så er dette nesten en umulig oppgave. Bateson illustrerer dette med et bilde av en kunstner som befinner seg på en bevegelig trapp, hvis posisjon han forsøker å kommunisere, men hvor bevegelsene i seg selv er en funksjon av hans forsøk på å kommunisere. Som Bateson legger til: “Clearly this task is impossible, but, as has been remarked, some people do it very prettily.” (Bateson, s.138)

Metafor og metonymi

Tidligere viste jeg til hvordan Roman Jakobson har arbeidet med to aspekter av språket: metaforen og metonymien. Artikkelen *Two Aspects of Language and two types of Aphasia* bygger på Saussures språkteori. Enhver lingvistisk enhet kan ifølge Saussures inngå i to forskjellige forbindelser. Assosiativ eller *paradigmatisk* forbindelse med andre enheter, eller *syntagmatisk* forbindelse. Den syntagmatiske (horisontale) forbindelse (syntagme) er en forbindelse av lyder eller bokstaver til et ord, ordgrupper eller setning. I ordet pil, inngår p i syntagmatisk forbindelse med i og l. Motsatt er en paradigmatisk (vertikal) forbindelse assosiativ. Når vi skriver eller snakker, så velger vi hele tiden fra vårt språklige reservoar, som kan være ekvivalente utfra visse aspekter eller forskjellige ut fra minst ett aspekt. (Jakobson, 1995, s.51) Istedenfor p kunne vi ha valgt b foran –il. Disse bokstavene/lydene inngår i paradigmatisk forbindelse med hverandre. (De rimer.) Jakobson bruker andre betegnelser, og sier at de enheter som utgjør et syntagme, en kontekst, har en kontiguitets-status i forhold til hverandre, (kontiguitet=naboskap, berøring), mens de enheter som kan byttes med hverandre, er forbundet med hverandre ved forskjellige grader av similaritet. Disse kan veksle mellom synonymi og antonymi. Han bruker betegnelsen *kombinasjon* om syntagme og *seleksjon* om paradigmatisk forbindelse. Jakobson snakker om språkets to aspekter som

kombinasjonsplan og seleksjonsplan, og han foretar undersøkelser blandt afasi pasienter ut fra denne lingvistiske teori. Det han påpeker er at det er to typer afasi. For den ene pasientgruppen som er ramt på seleksjonsplanet – spiller konteksten en avgjørende faktor:

“Når en presenterer fragmenter av ord eller setninger for en afasiker av denne typen, fullfører han dem lett. Hele hans tale består av reaksjoner. [...] En pasient av denne typen klarer ikke å finne synonymmer til et gitt ord, og heller ikke heteronymer. [...] Han har mistet sitt metaspråk, sier Jakobson. Pasienter av denne typen kan forstå ordene i deres bokstavelige betydning, men: klarer [ikke] å oppfatte metaforiske betydninger av de samme ordene. Derimot benytter de seg i stor utsrekning av stilfiguren metonymi” (Eggen,1976, s.3)

Disse pasienter kan si gaffel istedenfor kniv, bord for lampe, død for svart ol. Sammenhengen blir avgjørende fordi seleksjonskapasiteten rammes. Den andre form for afasi, kombinasjonsforstyrrelser, kjennetegnes ved at pasientene har problemer med å konstruere setninger. Pasientene snakker i “telegramstil” fordi ord med grammatiske funksjoner forsvinner (konjunksjoner, artikler, pronomener ol.), språkets hierarkiske orden blir vanskelig å fastholde. Men denne gruppen har beholdt sin evne til seleksjon, og typisk benytter denne gruppen metaforlignende uttrykk. De kan si langsyn istedenfor mikroskop, ild for gasslys ol. Roman Jakobson mener at begge disse aspektene av språket (prosessene) er tilstede i normalspråk, under innflytelse av legning eller kulturelle mønstre, og at den ene formen kan være mer dominerende enn den andre. I afasi pasientenes tilfeller snakker vi om en blokkering av den ene eller andre formen.

I den retoriske tradisjon taler man om likhet (simile og metafor) eller nærhet (metonymi). Metonymien og synekdoten er karakterisert ved at tingene “ikke nevnes ved sitt rette navn”, men ved et begrep eller en forestilling som tilhører samme assosiasjonsfelt. (*Moskva* kunngjør, Brede *seil* over Nordsjø går.) I metonymien skjer det en forskyvning fra ett element til et annet innen samme livsområde, samme “kontekst”. Metaforen derimot, er et bilde som er karakterisert ved et sammenfall eller en sammensmeltning av to forestillingsfærer eller to forestillinger. (Du var vinden, eller klisjeer: vinden hyler). Similen er en poetisk sammenligning med “som” eller “lik” (Du er rød som en rose). Jakobson henviser – som Bateson – til Freuds drømmeteorier, og

sier “drømmearbeid” har nære forbindelser med de to prosesser. “Identifikasjon (en person i drømmen representerer en annen i virkeligheten) og seksualsymbolikk er metaforiske prosesser; fortetning og forskyvning (en hatt kan representere eieren) er metonymiske prosesser” (Eggen, s.6)

Jens F. Jensen (som bl.a. bygger på Umberto Eco) beskriver i *Reklame – Kultur* det samme fenomen ut fra en semiotisk tankegang. Hvis jeg har et paradigme f.eks. A vs. B vs. C ... av forskjellige tegn eller termer, og jeg har noen relasjoner fra disse termene til deres respektive semantiske markører (x er en av A`s semantiske markør, y er B`s osv.) Hvis jeg så velger å benevne A ved hjelp av x så er dette i tilfelle metonymi.

“Kaldes (edb-maskinen) en /informationsbehandler/ eller en /talknuser/, hvor en central semantisk egenskap ved maskinen – at kunne processere store mengder af information eller at kunne beregne og dermed nedbryde store tal-komplekser – sættes i stedet for termen selv, så er der tale om en metonymisk udskiftning.” (Jensen, 1993, s.110)

Hvis vi igjen tar paradigmet A, B, C som utgjør et paradigme av forskjellige termer, og hvor de vertikale forbindelsene utgjør relasjoner fra termene til deres respektive semantiske markører – så er i dette tilfelle x også markør for en annen term: D:

“A refererer således ved en kontiguitets-relation til x, og x refererer ved en kontiguitets-relation til D. [...] Med et eksempel løftet fra Eco (cf 1979:79) Hvis en lang hvid hals er en egenskap (x), der både karakteriserer en smuk kvinde (A) og en svane (D), så kan kvinden metaforisk erstattes af svanen (som eksempelvis i udsagnet: “hun var smuk som en svane” etc.)” (Jensen, 1993, s.112) D kan substituere A fordi de på en eller annen måte ligner hverandre, og det er denne substitusjonen Eco forstår som metafor.

I artikklen *Lingvistik og poesi* ønsker Jakobson å undersøke litterariteten som en egen type språkbruk. Ikke bare for å definere hva som gjør en språklig meddelelse til et kunstverk, men for å kunne peke på hva som fører til at det er den poetiske funksjon som dominerer. Han deler inn ethvert språklig uttrykk i *avsender – mottager*, men også de andre faktorer som *kontekst, kontakt, meddelelse, kode*. Disse faktorer bestemmer hver sin språklige funksjon. Avsender har ofte en emotiv funksjon, mottager en konativ. Konteksten har oftest en referensiell funksjon, kontakten en fatisk funksjon og koden

har ofte en metaspråklig funksjon. Når man har behov for å finne ut av om man taler samme språk, så rettes talen mot koden. (Jmf. innledende avsnitt og definisjon av meta-lingvistikk.) Men når språk innstiller på selve meddelsen selv - fokuserer på meddelsen for dens egen skyld - da mener Jakobson at det er den poetiske funksjon som dominerer. Jakobson mener at konkurransen mellom de to prosesser, metaforiske eller metonymiske, er manifest i enhver symbolsk prosess, enten den er sosial eller foregår innen det enkelte individ. Hva er det så som gjør en tekst poetisk ifølge Jakobson?

“Selektion foretages på basis av ækvivalens, lighed og ulighed, synonymi, og antinomi, mens kombinationen, opbygningen af ordfølgen er baseret på berøring, sammenhæng. *Den poetiske funktion projicerer ækvivalensprinsippet fra selektionsaksen over på kombinationsaksen. Ækvivalens gøres til kædens konstituerende princip.*” (Jakobson, 1987, s.47)

Poesi og metaspråk er diametrale motsetninger i følge Jakobson, for i metaspråket brukes kjeden til å danne uttrykk for likhet, (ugift mann = ungarl) mens i poesien brukes uttrykk for likhet til å danne en kjede. Rim & rytme kan være meningsbærende i et dikt. Trykk kan følge trykk ol. Form og mening kan ikke adskilles; mao. ligger det et implisitt budskap i selve utvelgelsen av koden.

Det er denne informasjonen, eller rettere sagt transformasjonen, Bateson interesserer seg for når han undersøker kunstverk. Å tolke et dikt innebærer å analysere dets interne strukturer eller mønstre. Diktet peker også utover seg selv; det forteller også noe om den kulturen det er skapt i. Den poetiske funksjon av spåket innebærer en lek med spåket; nye måter å sette språket sammen på. Uttrykket (søken etter grace), kan lykkes eller mislykkes. Også Lytje peker på det samme når hun viser til at formen (eller formlen) ikke kan adskilles fra mening. Form og mening er ett. Derrida viser til det samme når han peker på at formen i like høy grad er formens negasjon. Formen skiller det som uttrykket handler om fra det det ikke handler om.

Jakobsons diskurs er rettet mot språkbruk. Ved å peke på språkets to aspekter, kombinasjons og seleksjonsaksen, er han i stand til å vise når litterariteten eller språkets poetiske funksjon er den dominerende. Vi har aldri et budskap i en “ren” form. Det skjer stadig en omkodning eller en transformering av de budskaper som er i kretsløp.

På mange måter er det dette all programmering dreier seg om. Selve formalisering-prosessen innebærer en abstraksjon, hvor man bruker et formalspråk (programmeringsspråk, digital kommunikasjon eller hva man ønsker å kalle det) som representerer noe annet. Når teksten eller programmet først er skrevet, så blir transformasjonen foretatt av leseren. Jensens beskrivelse av metaforens betydning er etter min mening en god beskrivelse av hvilke muligheter for kreativitet som ligger som en mulighet for den dyktige, innovative programmøren:

“Semiotiske systemer er nemlig ikke begrenset til kun at udsige det, som allerede er artikulert og konventionaliseret i koden. Men er også i stand til at producere udsagn, som endnu ingen har sagt eller tænkt, eller at betegne det, som endnu ikke er blevet assimileret af kulturen, ved at konstruere nye ikke tidligere virkeliggjorte kombinatoriske muligheder og semantiske koblinger. Som betydningsgenererende mekanisme udgør metaforen her netop:” a new semantic coupling not preceded by any stipulation of the code (but which generates a new stipulation of the code)” (Eco, 1979, 69) [...] Metaforen har således en viktig æstetisk-kreativ funktion, idet den kan generere – i kulturen endnu ikke realiserede – innovative kombinationer og dermed såvel producere nye æstetiske erfaringer som udvide og forfine kodens muligheder. Ligsom metaforen har en afgørende kognitiv funktion, idet den – hvis den lykkes kulturelt – kan producere nye mentale modeller, ny viden og dermed også nye forestillinger om verden.” (Jensen m.fl.; 1993, s. 115)

Primær Prosesser

Det som kjennetegner primær prosessene, eller det ubevisste, er at de mangler tempus, modus, klare negativer og andre lingvistiske markører. Bevisstheten (sekundær prosessene) snakker om ting og personer, og kobler predikater til de ting og personer som nevnes. I primær prosessene blir hverken ting eller personer identifisert, men fokus legges på relasjonene som oppnås mellom dem, typisk kjennetegnet ved psykoanalytikerens som tolker drømmer og hvordan den frie assosiasjon danner mønster. Bateson påpeker, at dette i grunnen er en annen måte å si at diskursen i primær prosesser er metaforisk.¹

1. Bateson skiller i sin teori ikke mellom metafor og metonymi.

Bateson mener at primær prosesser først og fremst handler om relasjoner mellom selvet og andre personer, eller mellom selvet og omgivelsene. Det er dette som er grunnlaget for hva vi navngir som følelser: hat, kjærlighet, frykt, angst, fortrolighet, fiendtlighet osv.

“It is unfortunate that these abstractions referring to *patterns* of relationships have received names, which are usually handled in ways that assume that the “feelings” are mainly characterized by quantity rather than by precise pattern. This is one of the nonsensical contributions of psychology to a distorted epistemology.” (Bateson, s.140)

Drømme-arbeid, kan sammenlignes med den kommunikasjon som foregår mellom organismer, hvor all form for kommunikasjon foregår ikonisk. Vi snakker om en slags “primær prosess kommunikasjon”. Dette er spennende å observere blandt hunder. Hvis to hunder møtes, så har de ikke utviklet et verbal språk hvor de sier: “vi vil *ikke* sloss”. Hundene mangler *simple negative*. Derfor blir de nødt til å kommunisere det motsatte av hva de mener, for så å kunne negere det de i utg. punktet ikke ønsket. De møtes – reiser bust og snerrer til hverandre. Dette er ikonisk kommunikasjon i følge Bateson. Så må de undersøke hva denne snerringen betyr; skal vi sloss? De begynner derfor å bjeffe; for å finne ut av at den andre ikke har ønsker om å drepe – og deretter kan de være venner. Dyrenes diskurs handler om relasjoner – enten mellom selvet og andre eller selvet og omgivelsen. “In neither case is it necessary to identify the relata”. (Bateson, s.141) Dyr A forteller dyr B om sitt forhold til B og det forteller dyr C om sitt forhold til C. Dyr A forteller ikke C om sitt forhold til B. Denne form for ikonisk kommunikasjon mener Bateson er mer arkaisk enn den verbale (fra et evolusjonsperspektiv), og han viser derfor til det Samuel Butler påviste; at det vi vet best er det vi er minst bevisste om. Mao. at vanen er den bevisste tankens økonomi. Det ubevisste inneholder ikke bare de vonde erfaringer som bevisstheten fortrenger. Dette er allikevel ikke så enkelt som det tilsynelatende virker. På den ene side kan vi økonomisere med de nivåer av bevisstheten som fortsetter å være sanne uansett om konteksten forandrer seg, men vi må stadigvekk ha tilgjengelighet og kontroll over den adferden som må korrigeres i forskjellige situasjoner. Vi må ha et pragmatisk forhold til hva som skal beholdes på overflaten, eller bibeholdes i det vi navngir som det bevisste. På samme måte som når vi lærer å kjøre bil. I starten - og heldigvis med kjørelærer ved siden av – tenker man på hvordan man gearer, og hvilken fot man setter ned hvor. Etterhvert

som disse ferdigheter blir bedre pga. vanen, så økonomiserer vi dermed vår bevissthet, og blir bedre og bedre i stand til å tolke informasjon fra omgivelsene.

Det at vanen og ferdigheter synker ned i dypere ubevisste lag av bevisstheten, kan sammenlignes med hva Ragnar Fjelland omtaler i artikkelen *Uartikulert kunnskap og dens betydning for informasjonsteknologien*, hvor han tar for seg hvordan man opp gjennom historien har nedvurdert det han betegner som uartikulerte kunnskap. Vår nyere tids filosofi regnes for å ha sin begynnelse med Renè Descartes. Han betraktet kroppen som en del av den materielle verden, og ikke som del av det erkjennende subjekt. Moderne erkjennelsesteori har sett på kroppens del av erkjennelsesprosessen som noe passivt, og det som har med kroppen som handlende subjekt å gjøre, har blitt oversett. Man har sett vekk fra den kunnskap som ikke er artikulert. "Dette har nok medvirket til en generell nedvurdering av den kunnskap som f.eks. representeres ved håndverk, men det har også medført at man ikke har sett den fulle betydning av uartikulert kunnskap i dagliglivet og heller ikke i vitenskapene". (Fjelland, 1988, s.107) Den engelske fysiker og filosof Michael Polanyi, (*Personal Knowledge*, 1958), satte ord på det han betegnet som "tacit knowledge" ("taus kunnskap"). Også John Searle, *Speech Acts*, skiller mellom "knowing how" og "knowing that". I en informasjonsteknologisk sammenheng, er kravet til kunnskap at den må være artikulert. Terry Winograd og Fernando Flores (*Understanding Computers and Cognition*, 1986,) satte søkelyset på hvor komplisert det er å forsøke å artikulere mye av den kunnskap vi bruker i dagliglivet. Dette er andre ord for hva Bateson gjør rede for i sin artikkel.

Fjelland viser til hvordan filosofien og vitenskapene (Platon) har skilt mellom kunnskap (episteme) og ferdigheter (doxa). Videre skriver han, at man vanligvis har hatt et syn om at minimumskravet til virkelig kunnskap er at den er artikulerbar; mao. at den kunnskapen som ikke er artikulerbar, befinner seg på et så lavt kognitivt nivå, at man ikke kan snakke om kunnskap i det hele tatt. "Det uartikulerte har blitt forbundet med ferdigheter, stort sett manuelle ferdigheter. Når man stiger oppover i hierarkiet av kognitive aktiviteter, til vitenskapene, og fremfor alt til de teoretiske vitenskapene, dreier det seg om artikulert kunnskap". (Fjelland, 1988, s.114) Fjelland problematiserer dette synet ved å vise til Hubert & Stuart Dreyfus. I verket *Mind over Machine* presenterer de en modell for kompetanse utvikling.

- 1) Nybegynner
- 2) Avansert nybegynner
- 3) Kompetanse
- 4) Profesjonalitet
- 5) Ekspertise

Uten å gå i detaljer så vil utviklingen gå fra (1.) eksplisitte situasjonsavhengige regler, hvor man mer og mer (2.) blir i stand til å modifisere ut fra situasjonen. Kompetanse (3.) betegnes ved at utøveren kan lage en hierarkisk prosedyre for beslutninger. Fremgangsmåten er analytisk og kalkulerende, og innebærer å overveie alternativer før man foretar bevisste valg. Mao. er denne kunnskapen artikulert. Profesjonalitet (4.) er i likhet med nivå to et mellomstadium. Det Dreyfus& Dreyfus viser til er at en ekspert (5.) utfører en aktivitet (f.eks. en racerkjører) som om det var en hverdagslig aktivitet. Bilen blir en forlengelse av kroppen.

“Fordi eksperten har erfaring fra mange forskjellige situasjoner, vil han kunne gjenkjenne en situasjon og handle intuitivt, uten å analysere og overveie alternativer. En sjakk spiller på ekspertnivå er i stand til å gjenkjenne omtrent 50.000 forskjellige situasjoner. Dette er situasjoner han kan gjenkjenne med et blick, uten noen som helst form for analyse eller anvendelse av regler. [...] Kompetent utøvelse er analytisk, kalkulerende og artikulert, mens ekspertutøvelse er holistisk, intuitiv og uartikulert. Dette innebærer selvfølgelig ikke at all den kunnskap en ekspert sitter inne med, er uartikulert. Når eksperten møter uventede og problematiske situasjoner, vil han analysere og overveie alternativer.” (Fjelland, 1988, s.118)

Dette er i samsvar med hva Bateson legger frem om at vanen er den bevisste tankens økonomi. Men som vi tidligere var inne på, må vi ha et pragmatisk forhold til hva som skal beholdes på overflaten, eller bibeholdes i det vi navngir som det bevisste (her artikulerte).

Kunstens korrigerende natur

“If, as we must believe, the total mind is an integrated network (of propositions, images, processes, neural pathology, or what have you-according to what scientific language you prefer to use), and if the content of consciousness is only a sampling of different parts and localities in this network; then inevitably, the conscious view of the network as a whole is a monstrous denial of the integration of that hole.[...] What the unaided consciousness (unaided by art, dreams and the like) can never appreciate is the *systemic* nature of mind.” (Bateson, s.145)

Bateson illustrerer sin systemiske tankegang med en sammenligning. Den levende menneskekroppen er et komplekst, kybernetisk integrert system. Medisinere og vitenskapsmenn har studert kroppen i årevis. Hva de vet idag, kan sammenlignes med hva bevisstheten vet om mind. En lege ønsker å kurere, og forskningen blir derfor fokusert (på samme måte som oppmerksomhet er bevissthetens måte å fokusere på), på disse mindre årsakskjeder som forårsaker sykdom. Når man så oppdager en effektiv kur, rettes oppmerksomheten et annet sted. Uten at man dermed forstår mer om de systemiske aspektene av en gitt sykdom. Men selv om man kan kurere og forhindre mange sykdommer – så fører ikke dette til en overordnet *visdom*. Økologi og populasjonen mellom artene er forrykket; parasitter har blitt immune mot antibiotika. (Dagspressen er full av den slags eksempler på resultater av legevitenenskapen. F.eks. hvordan scanning brukt for å utrede sykdom, kunne gi deg svar på at man var frisk, men at selve scanningen utsatte deg for stråler som kan føre til uhelbredelig kreft.) Batesons poeng er ikke å henge ut legestanden, men han ønsker å slå fast det uunngåelige:

“That mere purposive rationality unaided by such phenomena as art, religion, dream, and the like, is necessarily pathogenic and destructive of life; and that its virulence springs specifically from the circumstance that life depends upon interlocking circuits of contingency, while consciousness can see only such short arcs of such circuits as human purpose may direct. In a world, the unaided consciousness must always involve man in the sort of stupidity of which evolution was guilty when she urged upon the dinosaurs the common-sense values of an armament race. She inevitably realized her mistake a million years later and wiped them out.” (Bateson, s.146)

Batesons noe pessimistiske syn, indikerer at menneskets bevissthet vil si det samme som dets evne til å se kun deler av helheten – eller en liten del av systemet – og at denne manglende evne nødvendigvis fører i retning av hat og dumskap. Individet blir hele tiden overrasket og sint, når den harde politikk får tilbakevirkende kraft. Mao agerer vi, uten å se helhet, og dette får tilbakevirkende kraft på oss selv, som gjør oss frustrerte, slik at vi agerer osv. osv. Bateson mener at det er en slik verden vi lever i; strukturer av kretsløp. Den eneste måten kjærlighet kan overleve på - er at det eksisterer en form for gjenkjennelse av at slike kretsløp eksisterer. Dette kaller han *visdom*. Menneskene behøver kunst, religion og drømmer; vi behøver det kreative. Derfor er Bateson, fra et antropologisk synspunkt, ikke interessert i kunst utfra spørsmål om hva dette kunstverk kan fortelle om kunstneren. Bateson kunstsyn, er et spørsmål om kunsten har en posistiv funksjon som bibeholder av det han kaller visdom.

“In correction of a too purposive view of life and making the view more systemic, then the question to be asked of the given work of art becomes: What sort of correction in the direction of wisdom would be achieved by creating or viewing this work of art? The question becomes dynamic rather than static.” (ibid.)

Bateson slår fast at nesten uten unntagelse, vil den adferd som vi kaller kunstnerisk, eller kunst produkter ha to karakteristikk; de krever eller fremstiller håndlag (skill) – og de inneholder redundans eller mønstre.

Oppsummering

I dette teori avsnittet har jeg forsøkt å trenge inn i hva selve det å kode innebærer, eller hva det vil si å programmere. Inger Lytje ser systemutvikling som produksjon av tekster, og P. Naur legger i sine artikler vekt på at en programmør må bruke sin intuisjon, eller med Batesons ord “det ubevisste” – når han/hun programmerer. Fjelland kommer inn på hvordan den moderne erkjennelsesfilosofi har sett på kroppen som del av den materielle verden, og derfor har nedvurdert det han kaller “uartikulert kunnskap”. Når Bateson beskriver kunstens epistemologi, så har han ikke et dualistisk syn på sjel vs. legeme, men han makter derimot å beskrive hvordan kunsten blir et slags interface mellom det bevisste og det ubevisste. Kunsten minner oss om at disse lagene eller hierarkiene av bevissthet eksisterer. Bateson viser også til hvordan gjentagelsen og håndverket gjør at disse former for bevissthet trenger ned i dypere og dypere lag av

bevisstheten. Etter min mening kroppsliggjør han nærmest det man kunne kalle den kunstneriske prosess, og han makter å synliggjøre hva kreativiteten består av. Det som kjennetegner mennesker er at vi i form av enten ikonisk, analog eller digital koding, hele tiden interagerer med omverden ved hjelp av vår metakommunikasjon.

Når jeg har lagt såpass stor vekt på dette med metafor og metonymi, er det fordi det innen all kreativ aktivitet – og i denne sammenhengen programmering – innebærer at man skaper nye uttrykk, og setter sammen kode i nye sammenhenger. I min hovedinnledning beskrev jeg forskjellige faser i systemutvikling, som faser hvor form & innhold ble adskilt i en analytisk prosess. Rent metodisk, tenker jeg da på en adskillelse hvor nye generasjoner av programmeringsspråk utviklet nye syntakser. Én ting er adskillelsen: programmer og input/output, hvor formen klart viser til programmets utforming og innholdet til dataene. En annen måte å tolke form & innholdsbegrepet på, er knyttet mer til selve den prosessen som foregår når man har en syntaks/et språk, og skriver et program ved hjelp av dette dynamiske system av tegn. Da vil noen av de som skriver programmer ha en utfoldelsestrang som gjør at de leker med koden/på samme måte som lek med språket kan føre til at den poetiske funksjonen i språket dominerer. Da vil det å skrive programmer - si det samme som å eksperimentere i rommet mellom form og innhold. Form & innhold kan ikke lenger adskilles, fordi det implisitt ligger en mening i selve utvelgelsen av koden. Det er denne innovative prosess som av noen kan utføres så elegant, eller kanskje effektivt, at enkelte utøvere kan tituleres som code poets. Dette uttrykket er i seg selv en metafor. Uttrykket er positivt ladet og overfører noe av gleden og stoltheten ved det å programmere eller å være en hacker. En poet vil kunne utgi sine dikt, og bli mer eller mindre berømt for sitt kunstverk. En hacker vil ha svært vanskelig for å kunne kommunisere sitt uttrykk/kode til andre utenforstående, enten det være seg sin mor, bror, kjæreste, eller andre deler av den organisasjonen man er del av. Ikke innviede forstår rett og slett ikke helt hva det vil si å programmere, fordi dette er en svært abstrakt aktivitet som utføres i et utpreget arbitrært språk som andre ikke kan tolke eller lese.

I innsamling av materiale til denne oppgaven, intervjuet jeg Mads Toftum, som er medlem av Århus Linux Group. Han jobber som systemansvarlig til daglig, og som han ga uttrykk for: “det er utrolig svært å kommunisere til de andre i firmaet når man bedriver hjernearbeid. Det kan se ut som om jeg sitter og gjør ingenting, mens jeg

kanskje har vært søvnløs i tre netter, fordi jeg forsøker å finne en måte å løse et problem på.”¹

Programmering kan være ensomt. Derfor antar jeg at en av drivkreftene, til at folk deltar i Open Source miljøet, er et fellesskaps behov. Man blir del av en kultur, hvor man blir bekreftet i sitt arbeid, hvor andre kan sette pris på det man bidrar med. Alle har vi behov for bekreftelse, og alle har vi behov for at andre setter pris på det arbeidet man har utført.

Jeg har i første del, forsøkt å føre tekstbegrepet innen programmering videre, å beskrive den kunstnerisk prosess, for dermed å kunne skape en større forståelse for den kreative og innovative prosess som kjennetegner programmering. Denne prosess er også i samsvar med den måten programmører omtaler seg selv, eller i hvertfall dem man innen miljøet ser opp til; slik som uttrykket *code poets* illustrerer. Som jeg var inne på tidligere i forbindelse med softwareengineering tradisjonen, er denne tradisjonen, eller mao. programmørers arbeidsforhold/omgivelser underlagt de rammer som kjennetegner den industrielle produksjon og time-management. Den tidligere industrialisering, og dens oppsplitning av arbeidsprosessen, medførte et syn på arbeideren som lett utskiftbar; en arbeider kunne lett skiftes ut med en annen. Virksomhetens verdi lå så og si i selve fabriksbygningen og maskinene når arbeideren gikk hjem om aftenen. Når vi idag snakker om de spesialiserte, vitenstunge industrier, som f.eks. utvikling av software, så er det nettopp her bildet har forandret seg. Medarbeiderne er virksomhetenes viktigste ressurs. Mao. går selve verdien av virksomheten hjem hver eneste dag! En kontorbygning og datamaskiner kan leases. Verdien ligger ikke her, men nettopp i måten man organiserer og kanalisere kunnskapen i organisasjonen på. Medarbeiderne er i kraft av sin ekspertise og kunnskap, kilden til nye produkter og økt konkurransevne. Derfor heter det ikke *arbeider* lenger – det heter *medarbeider*. Man hører stadig innen IT-bransjen om behovet for innovasjon og kreative medarbeidere. Men det ligger en motsetning her, i den måten softwareutviklings prosjekter og programmørers arbeidsforhold blir organisert på, og hva deres arbeid rent faktisk går ut på.

1. Jeg viser til hovedinnledning, hvor jeg gjør rede for innsamling av materiale for denne oppgaven. Når jeg siterer M. Toftum vil jeg i det videre ikke oppgi referanse.

Del 2

Vitensdeling i systemutvikling

"The idea of open source has been pursued, realized, and cherished over those thirty years by a vigorous tribe of partisans native to the Internet. These are the people who proudly call themselves "hackers" – not as the term is now abused by journalists to mean a computer criminal, but in its true and original sense of an enthusiast, an artist, a tinkerer, a problem solver, an expert". (Raymond, 2001, innledning)

Innledning

Ved å analysere utviklingen av Open Source miljøene, vil jeg i dette avsnittet vise at omfanget og kvaliteten i vitensdeling innen systemutvikling, avhenger av åpenhet i økonomi og intellektuell kapital, kompleksitet, organisering og metode, samt læring. Vygotskys medierende perspektiv, viser at menneskets evne til å ta i bruk redskaper, å lære og dermed utvikle seg, er kulturelt betinget. Jeg vil vise at de ovenfor nevnte forskjellige former for vitensdeling, medvirker til å skape et miljø hvor det foregår læreprosesser, og dermed læring.

Open Source Community avspeiler en tankegang rundt organiseringen av hvordan systemutviklere, eller hackere, arbeider sammen på forskjellige prosjekter i et internasjonalt nettverk, for å utvikle programkode som er tilgjengelig for offentligheten. I dag er det særlig to hovedgrupperinger som bidrar til produksjon av software innen Open Source Community. Den største gruppen består av ulønnede idealistiske programmører som bidrar til fellesskapet. Det kan være ut fra et behov om å delta i et kulturelt felleskap, eller det kan være ut fra et behov for bedre funksjonalitet på programvare. Flere store computer-firmaer har etterhvert vist interesse for OSC, og sponset samarbeide om prosjekter innen Open Source, fordi de ser nødvendigheten av at deres egne produkter er kompatible med Open Source software. På denne måten er de også med på å skape noen felles retningslinjer for den teknologiske utforming. Ved senere å foretære systemer til offentligheten kan miljøet videreutvikle, vedlikeholde og eventuelt finne feil (bugs) i programvaren. Ved å starte prosjekter innen OSC, kan mange samarbeide og spare penger i utvikling, samt kvalitetssikre produktene.

Det som idag utgjør OSC, er et konglomerat av forskjellige små og store prosjekter og organisasjoner, som dekker forskjellige behov innen softwareutvikling. Når man ser tilbake på hvordan dette miljøet startet, var det behovet for et operativsystem som

satte det hele i gang. Historien om Linux, må sees i sammenheng med historien om Unix, fordi dette er fortellingene om hvordan et operativ system blir skapt. En annen viktig aktør i Open Source miljøet, er The Apache Foundation. Dette er en paraplyorganisasjon, som legger forholdene til rette for mangfoldige Open Source prosjekter. Denne stiftelsen var i utgangspunktet kjent som the Apache group, som utviklet Apache serveren. Grovt sagt kan man si at de utvikler nettverks relaterte verktøyer. Apache serveren har vunnet innpass overalt i verden, og har idag en stor markedsandel for servere.

Vitensdeling vil si det samme som å vise frem til andre hva man har uttrykt eller laget; mao. åpenhet. Denne form for vitensdeling kjennetegner de akademiske miljøer, som OSC har sitt utspring i. Forskjellen mellom proprietær og åpen kode, ligger i brukers mulighet for å kunne lese kildekoden. Kildekoden skrives i et programmeringsspråk, f.eks. C++, Java, Perl osv. Andre programmører kan se hvordan programmet er skrevet, og dermed hvordan programmet fungerer. På denne måten kan brukere også oppdage feil, eller såkalte bugs, og gi tilbakemelding til softwareutviklerne.

Lukkethet/proprietær kode, innebærer at andre ikke kan se koden, og det sier seg selv at man forhindrer den vitensdeling som Open Source muliggjør. OS tankegangen fremmer et syn om at man istedenfor å tenke på systemutvikling som stykkproduksjon – hvor hver enhet blir solgt i form av en lisens - heller bør dele viten. Istedenfor å definere softwareutvikling som en industriell stykkproduksjon, velger de å se denne industrien mer som en service-næring. Dette innebærer at inntjeningen ligger i service, support og konsulentvirksomhet for de åpne/gratis produktene. OSC mener at åpenhet i source koden gir brukerne eller kundene, en mye større grad av valgfrihet. Dessuten mener de at åpenheten gir en mye større mulighet for feilfinning/vedlikehold og dermed bedre produkter.

En av drivkreftene for OSC var behovet for verktøyer/teknologi, som man måtte betale penger man ikke hadde for. Microsoft er den virksomheten som er mest kjent for sin lukkethet, men også for sine lisensavgifter.¹

1. Til tross for at de for nylig har innført sin egen versjon av en open lisens, som ikke regens for å være en del av Open Source Community.

De gir stadig ut nye versjoner av den software forskjellige brukere har kjøpt, som f.eks. Microsoft Office, med dertil hørende lisenser. Dette medfører at eldre versjoner av f.eks. Word ikke er kompatible med de nyeste versjoner. Man “faller av lasset” hvis man ikke fornyer sin software og sine lisenser. Dette kan bli en dyr affære for mange organisasjoner. Dessuten vil en organisasjon som har satset på Microsoft, hvilket jo svært mange har, de besitter jo store deler av pc-markedet, bli “fanget” – fordi Microsofts software er inkompatibelt med andre plattformer. Det er på denne bakgrunn myndighetene i mange land, forsøker å utrede hvor vidt Open Source bør erstatte f.eks. Microsofts proprietære kode.¹

Forskjellige former for vitensdeling

Innledningsvis vil jeg forsøke å skissere opp de forskjellige former for vitensdeling, som vil fremgå av den historiske gjennomgang. Del 2 som omfatter vitensdeling, flytter fokus fra individplan og over til gruppeplan. Det er likeledes hensiktsmessig å gjøre rede for det medierende perspektiv, fordi dette kan klargjøre hva en læreprosess innebærer for et individ i samspill med omverden. Tidligere har jeg beskrevet den ostensive eller påpekende kommunikasjon som forutsetning for innlæring av ny kode. Jeg har valgt å trekke inn Vygotsky og det medierende perspektiv, fordi denne forståelsen gir en utfyllende beskrivelse av læring og læreprosesser i et gruppeperspektiv.

Vitensdeling – knyttet til spørsmål om intellektuell kapital og økonomi:

Selv om man er åpen, vil man gjerne ha kreditt for det man har gjort; mao. handler dette om spørsmål knyttet til intellektuell kapital og copyright. Det finnes mange diskusjoner innen Open Source miljøet, som er knyttet til opphavsrett, lisenser og økonomi. Dette kommer jeg tilbake til.

Vitensdeling knyttet til spørsmål om kompleksitet

Softwareutviklingens historie, er også historien om forskjellige generasjoner av programmeringsspråk. Dette medfører at det finnes forskjellige metoder og strategier for systemutvikling. Noen systemer er på flere millioner linjer, mens andre f.eks. Unix kommandoer, er bygd opp ved hjelp av moduler. Det nye ved Unix var nettopp at den

1. Teknologirådet har i oktober 2002 utgitt en rapport om Open Source i Danmark, hvor de mener at staten kan spare milliarder ved å gå over til åpne standarder.

var modul basert. Dvs. at mulighetene for vitensdeling og samarbeid er knyttet til selve den måten systemet er utarbeidet på.

Vitensdeling knyttet til spørsmål om organisering og metode

Den måten Linux prosjektet ble gjennomført på, ved hjelp av Internet, innebar en innovasjon i systemutviklingsmetode som skiller seg fra annen softwareutvikling. Her satt én person, og sendte ut sine versjoner/forslag, (delte sin viten) og fikk så en mengde feed-back og nye forslag fra utviklere i hele verden (andre delte sin viten), som igjen gjorde at han sendte ut ny en versjon osv. osv. På den måten ble hundrevis av utviklere engasjert, og en omfattende vitensdeling organisert. Dette er av mange betegnet som en innovativ metode i systemutvikling.

Vitensdeling knyttet til spørsmål om læring

Alle som skal lære å programmere, lærer av å etterligne andres kode. Som alle andre menneskelige praksiser, utvikler en programmør sine ferdigheter gjennom en kombinasjon av undervisning og praksis over tid. Man prøver og feiler og prøver igjen. Open Source Community fremmer vitensdeling i form av læring, fordi de uerfarne sender inn spørsmål, eller egne bidrag til OS prosjekter. Disse kan være enten gode eller dårlige, og deltakerne får respons på sine forespørsler/bidrag. Det er de som styrer/eier prosjektene som bestemmer hvilke bidrag som tas med.

Det medierende perspektiv i synet på læring og læreprosesser

Når man deler viten i en eller annen forstand eller innen systemutvikling, så foregår det en utveksling av viten mellom mennesker, som kan utvikle seg til læreprosesser som igjen forutsetter læring. Innenfor virksomhetsteori eller activity theory, bruker man begrepet *medierende perspektiv* som setter fokus på disse prosessene. Det teoretiske rammeverk for et medierende perspektiv, ble utformet av russeren L.S. Vygotsky (1896-1934). Han undersøkte hvordan menneskene utviklet høyere psykologiske prosesser, og hvordan det er *språket* som skiller mennesker fra dyr, og dermed vår evne til å ta i bruk redskaper. Menneskehjernen har utviklet seg til det den er idag, lenge før vi oppfant skrive og regnekunst. Det er store individuelle forskjeller på hvordan mennesker kan skrive eller regne, men først og fremst er disse aktiviteter kulturelt betinget. Det er ikke noen naturlig "kognitiv" egenskap. Vygotsky var den første moderne

psykolog som påviste mekanismene for hvordan kulturen blir del av menneskets utvikling av høyere kognitive egenskaper. (Forord, *Mind in Society*, s.6)

“The specifically human capacity for language enables children to provide for auxiliary tools in the solution of difficult tasks, to overcome impulsive action, to plan a solution to a problem prior to its execution, and to master their own behavior. Signs and words serve children first and foremost as a means of social contact with other people. The cognitive and communicative functions of language then become the basis of a new and superior form of activity in children, distinguishing them from animals” (Vygotsky, s.29)

Vygotsky skiller mellom tegn (sign) og verktøy (tool), i menneskets medierende aktiviteter.

“A most essential difference between sign and tool, and the basis for the real divergence of the two lines, is the different ways that they orient human behaviour. The tool’s function is to serve as the conductor of human influence on the object of activity; it is externally oriented; it must lead to changes in the object. It is a means by which human external activity is aimed at mastering, and triumphing over, nature” (Vygotsky, s.55)

Det er denne formåen som er drivkraften i menneskets streben etter å beherske og triumfere over naturen ifølge Vygotsky. Motsatt tegnet, som ikke forandrer noe psykologisk i objektet - tegnet er rettet innad. Det er en intern psykologisk aktivitet som går ut på å meste seg selv; “The sign is *internally* oriented.” (ibid.) Vygotsky skiller på den måten mellom tekniske og psykologiske verktøy. Kaptillinin & Kuutti sier det slik:

“Culture provides tools which are used by human beings in their external activities oriented towards things or other people. This way external activities become externally mediated and change their structure accordingly.” (Kaptillinin & Kuutti, s.155)

Vygotsky beskriver prosessen med å peke. Et lite barn griper etter et eller annet objekt som ikke er inne rekkevidde. Han feker etter et eller annet. Moren kommer til unnsetning. Hun tolker barnet, at han forsøker å peke mot noe, som et ønske om å få tak i objektet, og gir ham det. Dette forandrer situasjonen. Fra en ubevisst feking i luften – lærer barnet å peke. Han ønsker en reaksjon, ikke fra objektet – men fra et annet men-

neske. “From an object-oriented movement it becomes a movement aimed at another person, a means of establishing relations.” (Vygotsky, s.56)

Det er hva Vygotsky kaller “inter-psykologisk” som er det første trinn i utvikling av nye høyere mentale funksjoner. Denne starter først som en distribusjon mellom mennesker. Neste trinn i utviklingen er den “intra-psykologiske”, en indre psykologisk tilpasning. Et verktøy eller artefakt har i følge Vygotsky både en redskapsmessig samt en språklig funksjon. Den redskapsmessige er utadrettet, slik f.eks. et dataprogram er en slags kasse med verktøy som man kan lave forskjellige ting med. Man kan bearbeide materien – i dette tilfelle, snakker vi om det materielle arbeidet foran computeren med arbitrære tegn utformet i et programmeringsspråk. Samtidig er den språklige funksjon (the sign) rettet mot menneskets evne til å tilegne seg tegnverdien av det som erfaring gir, mao.selve læreprosessen.

“Through the internalisation of mediated external activities internal activities become mediated, too (that is, natural mental functions are transformed into higher order mental functions).”(Kaptilin & Kuutti, s.155)

Den eksterne redskapsmessige funksjon i et artefakt (hands-on-erfaring) går forut for den forståelse man tilegner seg ved hjelp av de tegnfunksjoner som bruken av et artefakt skaper – det er dette som kalles læring.

Hacker kulturens opprinnelse

I følge Eric Raymond, kan det som betegnes som hacker-kulturen, først knyttes til aktivitetene ved MIT (Massachusetts Institute of Technology) allerede fra 1961. MIT hadde utviklet den første PDP maskinen, som ble utgangspunktet for programmering og utvikling i årene som fulgte. MIT's Artificial Intelligence Laboratory var verdens ledende forsknings senter innen AI (Artificial Intelligence) opp til starten av 1980 årene.

Det er viktig å ha klart for seg at begrepet *hacker*, har forskjellig betydning i vår egen tid. I media blir oftest denne betegnelsen brukt i negative sammenhenger. Man beskriver datakriminalitet, f.eks. det å bryte inn i systemer – og setter det i sammenheng med hackere. Dette til stor irritasjon for hele den gruppen av mennesker som bruker denne betegnelsen om seg selv. For mange – som definerer seg som hackere –

er dette et begrep som beskriver at man er del av et kulturelt felleskap. Raymond viser til hvordan hacker kulturen utviklet en slags ordbok, over forskjellige begreper. The Jargon File ¹ gir en fyldig beskrivelse av hva hacker begrepet står for, og den sier også noe om softwareutvikleres selvoppfattelse. Det er interessant hvordan dette globale miljøet av systemutviklere, eller nerder vil andre si, oppfinner nye ord og begreper. For å unngå å bli satt i klasse med datakriminelle har miljøet oppfunnet ordet *cracker*, som betegnelse på folk som riktignok er dyktige computereksperter, men som bruker denne kunnskap og viten på en negativ måte.

SAMPLE DEFINITION:

:hacker: n. [originally, someone who makes furniture with an axe] 1. A person who enjoys exploring the details of programmable systems and how to stretch their capabilities, as opposed to most users, who prefer to learn only the minimum necessary. 2. One who programs enthusiastically (even obsessively) or who enjoys programming rather than just theorizing about programming. 3. A person capable of appreciating {hack value}. 4. A person who is good at programming quickly. 5. An expert at a particular program, or one who frequently does work using it or on it; as in 'a UNIX hacker'. (Definitions 1 through 5 are correlated, and people who fit them congregate.) 6. An expert or enthusiast of any kind. One might be an astronomy hacker, for example. 7. One who enjoys the intellectual challenge of creatively overcoming or circumventing limitations. 8. [deprecated] A malicious meddler who tries to discover sensitive information by poking around. Hence 'password hacker', 'network hacker'. The correct term is {cracker}.

The term 'hacker' also tends to connote membership in the global community defined by the net (see {network, the} and {Internet address}). It also implies that the person described is seen to subscribe to some version of the hacker ethic (see {hacker ethic, the}).

It is better to be described as a hacker by others than to describe oneself that way. Hackers consider themselves something of an elite (a meritocracy based on ability), though one to which new members are gladly welcome. There is thus a certain ego

1. <http://www.tuxedo.org/jargon>

satisfaction to be had in identifying yourself as a hacker (but if you claim to be one and are not, you'll quickly be labeled {bogus}). See also {wannabee}.]¹

Som nevnt, var det aktiviteter ved MIT som regnes for å være begynnelsen av hackerkulturen. Etterhvert ble disse aktivitetene knyttet sammen med andre miljøer pga. Arpanet og det senere Internet. Arpanet var det første transkontinentale, høy-hastighets computer nettverk. I utgangspunktet var det forsvarsdepartementet i USA, som bygde dette nettverket. Det var et eksperiment på digital kommunikasjon under en eventuell atomkrig. I tilfelle atomkrig, ville militæret sikre seg at kommunikasjonen ble opprettholdt selv om radio nettverket ble eliminert. Dette utviklet seg til et nettverk mellom hundrevis av universiteter og forsvars-laboratorier. Arpanet (det senere Internet) er et nettverk bestående av massevis av delnettverk. Det er de såkalte TCP/IP protokollene og Routerne som binder nettverket sammen.² I 1983 ble den militære delen utskilt i sitt eget nett, MILnet, og Arpanet opphørte i 1989. Omtrent samtidig fikk NSF (National Science Foundation) ansvaret for å bygge ut infrastrukturen i Amerika, og dette førte til bedre linjer mellom øst og vestkysten, samt internasjonalt. Lignende tiltak ble fremmet i andre land. Etterhvert fikk kommersielle aktører innpass på nettet. Idag vet vi jo hvilken enorm utvikling som har foregått innen internett og stigningen i antall utbydere. CIX (Commercial Internet Exchange) er en organisasjon som knytter internettutbydere sammen.³

Ingen eier eller har totalansvar for internettet idag. Det har virket forvirrende på mange. En internet utbyder har ansvar for og eierskap til sin "del" av nettet, enten man eier selve infrastrukturen eller leier den. Utbyderen har ansvaret for driften av sitt nett frem til dette nettets kontaktflate mot resten av internettet. Det finnes mange ideelle organisasjoner som søker mot standardisering på nettet. Grovt sagt kan man si at man forsøker å arbeide seg fram til noen felles "trafikkregeler" på "the information

1. Definisjon av hacker: <http://mitpress2.mit.edu/book-home.tcl?isbn=0262680920>

2. TCP/IP > Transmission Control Protocol/ Internet Protocol. Dette er navn på en rekke protokoller, som benyttes ved kommunikasjon via Internet. F.eks. FTP, Telnet ol. Hver maskin har sin unike adresse, og hver tjeneste har sin port. Routerne bestemmer hvordan trafikken skal sendes gjennom de forskjellige delene av nettet.

3. [http:// www.cix.org](http://www.cix.org)

highway”. Av organisasjoner som kan nevnes, er W3.org samt The Internet Society.¹ (Christensen, 2000)

Når man skal beskrive hvordan hacker-kulturen har utviklet seg, så kommer man ikke utenom internettets betydning for denne utviklingen. Det at mennesker har blitt istand til å kommunisere via computere i et globalt nettverk er selve grunnpilaren i Open Source miljøet. Det er de selvsamme hackere som har arbeidet for å utvikle den programvaren som må til for å kunne gjennomføre denne kommunikasjonen. Fra å være små sub kulturer ved forskjellige universiteter og laboratorier, ble de forskjellige miljøer knyttet sammen i et større nettverk. Tidligere ble nevnt MIT's AI som i samarbeid med LSC var første ute. Senere kom Stanford University's Artificial Intelligence Laboratory (SAIL) og Carnegie-Mellon University (CMU) til å spille en rolle som bidragsytere til den tidlige hacker-kulturen. En annen viktig bidragsyter var Xerox Park, og det berømte Palo Alto Research Center. Fra tidlig på 70 tallet til midten av 80 tallet stod dette forskningssenteret for svært mange innovasjoner – både innen utvikling av hardware og software.

Utviklingen av Unix

På Alexandra Instituttet/Universitetet i Århus arrangerte Århus Linux Group en forelesningsrekke vinteren 2002. En av foredragsholderne var Peter Salus. (*Ten years of Linux.*) Han innledet forelesningen med utgangspunkt i året 1969. Det var dette året de første mennesker landet på månen. Salus mente at historisk sett, hadde dette mindre betydning for menneskene enn noen av de andre begivenheter som fant sted samme år, nemlig at en mann ved navn Ken Thompson og en annen ved navn Dennis Richie (som utviklet programmeringsspråket C), begynte å utvikle Unix. Dessuten ble den første router lansert, IMP – Interface Message Processor – noe som også kan kalles internets fødsel. Tradisjonelt var operativ systemer skrevet i et assembler språk.² Thompson & Richie var blandt de første som innså at hardware og kompilerteknologien hadde blitt såpass utviklet, at hele operativ systemet kunne utvikles i C. Thompson & Richie var ansatt ved AT&T Bell Labs, og de begynte å utvikle Unix for eget bruk, for venner og samarbeidspartnere. Hjelpesprogrammer ble utviklet av andre,

1. <http://www.w3.org> <http://www.isoc.org>

2. Assembly language > symbolsk maskinspråk, eller lavnivåspråk, som anvender mnemoniske instruksjoner.

ettersom de hadde problemer de måtte løse. Bell Labs var ikke del av computerindustrien, og derfor ga de ut source koden til universiteter for en symbolsk betaling. (Peek, O'Reilly, Loukides, 1993, s.1)

“This had never been done before before, and the implications were enormous. If Unix could present the same face, the same capabilities, on machines of many different types, it could serve as a common software environment for all of them” (Raymond, s.9)

Dette innbar at brukere ikke behøvde å betale for ny software hver gang en maskin gikk ble foreldet, men at man kunne bruke forskjellige verktøy på forskjellige maskiner. De tidlige computere og software kostet store penger. Før microcomputeren ble et faktum, var det bare universiteter, virksomheter, offentlige institusjoner, militæret og forskningsentre som kunne finansiere stor-maskinene. Ifølge Peter Salus kostet en PDP-11 maskin 20.000 dollar rundt midten av 70 tallet. Portabiliteten var én ting, men en annen styrke med Unix og programmeringsspråket C, var at de var konstruert ut fra en KISS filosofi: “Keep It Simple, Stupid”. Tidligere var det ofte nødvendig for en utvikler hele tiden å referere med manualer når man skrev kode. Programmeringsspråket C hadde den fordel at en utvikler lett kunne ha oversikt over hele språkets logiske struktur. Unix var strukturert som et fleksibelt samling verktøy (toolkit) som kunne kombineres med hverandre. Ved AT&T utviklet Unix kulturen seg, men den ble snart spredd til mange andre miljøer. Unix miljøet utviklet et eget nettverk *Usenet*, ved hjelp av UUCP, som er et systemprogram og en rekke protokoller, som brukes til å overføre data mellom Unix maskiner. Typisk mellom to modemer sluttet til telefonnettet. (Lekanger, s.101). I 1980 begynte de første Usenet sitene å utveksle nyheter, hvor den viktigste tjenesten var nyhetsgrupper. Der kunne gruppene diskutere forskjellige emner, stille spørsmål osv.

Rundt 1980, var det i følge Raymond tre retninger som overlappet hverandre men som samlet seg rundt tre forskjellige teknologier; 1) Arpanet/PDP-10 hackers. PDP-10 var en type maskin som benyttet seg av et operativsystem som kaltes ITS. Disse brukte programmeringsspråk som LISP, MACRO, TOPS-10 og SAIL. 2) Unix og C gruppen brukte PDP-11 og VAX maskinene og telefonforbindelser. 3) En anarkistisk gjeng av tidlige microcomputer entusiaster (Apple, med Steve Jobs ble startet opp i 1977). Deres programmeringsspråk var BASIC.

Det er Berkeley varianten av Unix – på PDP-11 og VAX maskiner, som står igjen som de facto standarden. Dessuten begynte utviklingen innen microcomputere å skyte fart fra 1982-83 – med lanseringen av de første generasjoner av det vi idag kaller arbeidsstasjoner.

“In 1982, a group of Unix hackers from Stanford and Berkeley founded Sun Microsystems on the belief that Unix running on relatively inexpensive 68000-based hardware would prove a winning combination for a wide variety of applications. They were right, and their vision set pattern for an entire industry.”(Raymond, s.12)

Unix blir proprietær kode

Under kapittel-overskriften “*The Collapse of the Community*”, (*Open Sources: Voices from the Open Source Revolution*) beskriver Richard Stallman starten av 1980 årene, hvor Digital sluttet å produsere PDP-10 maskinene, og hvordan all programvare som benyttet ITS operativsystemet ble foreldet. Han beskriver også hvordan det spesielle hacker miljøet ved MIT`s AI Lab gikk i oppløsning. Fra da av var hackerne henvist til å bruke andre maskiner:

“The modern computers of the era, such as the VAX or the 68020, had their own operating systems, but none of them were free software: you had to sign a nondisclosure agreement even to get an executable copy.” (Stallman, s.2)

Peter Salus beskrev i sitt foredrag (*Ten years of Linux*), hvordan frustrasjonen blandt hackerne etter hvert økte, fordi de rett og slett ikke lenger fikk fri tilgang til en så grunnleggende teknologi som Unix. På en Usenix konferanse i Toronto i 1979, reiste AT&T`s advokater seg opp og lanserte lisenser som begrenset bruker-rettighetene og med høye priser etter datidens målestokk. Salus mente at advokatene og denne utviklingen provoserte miljøet til å skape nye systemer som kunne unngå lisensen; og hvordan utvikling startet som har utviklet seg til Open Source idag. Salus beskrev hvordan Andy Tanenbaum, en professor fra Nederland var “crestfallen” over AT&T`s lansering. Han deltok på konferansen fordi han ønsket å bruke Unix i sin undervisning rundt operativsystemer, men innså at universitetet ikke hadde så store midler til rådighet at de kunne kjøpe lisens til hver av operativsystemene på hver maskin. Dette førte til at han startet et prosjekt Minix, noe som også kan kalles en Unix klon.

Ovenfor har advokatenes krav om lisenser på Unix, blitt trukket frem som pådrivere for det som senere har utviklet seg til Open Source miljøet. Men som nevnt i innledningen, er vitensdeling også knyttet til spørsmål om kompleksitet. Mads Toftum, medlem av Århus Linux Group, mener at en av de viktigste drivkraftene for dannelsen av OSC, ligger i den måten Unix var utformet på. Som jeg tidligere var inne på består Unix av små komponenter, dvs. mange små kommandoer. Fra før av har universitetsmiljøet vært preget av at alle deler sine oppdagelser med andre. Det ble plutselig forholdsvis enkelt å bidra med små nye kommandoer, en programmør kunne f.eks. sitte og se en ny smart løsning, og skrive en liten komponent på ti minutter. Hvis andre også hadde bruk for den, så var jo det bare fint. Denne nye form for operativsystem gjorde at selve måten man benyttet kjernen på gjorde at operativ systemet ble til et multi-bruker system. Det var forholdsvis enkelt å skrive komponenter, og det var forholdsvis enkelt å sette dem sammen på nye innovative måter. Hvis man i motsetning hadde fortsatt i den gamle batch-mode, så ville det vært mye vanskeligere å dele eller samarbeide. En batch-mode innebærer at man må ha oversikt over et stort system, en mye mer tungvint måte som utelukker den form for vitensdeling som ble resultatet av den måten Unix var utformet på. Det man også kunne dele og bidra med i disse vitenskapelige miljøene, var viten om hvordan man utnyttet de muligheter som fantes.

Starten av 1990 årene var preget av frustrasjon blant hackerne, fordi den kommersielle Unix fortsatt var dyr. En lang krangel mellom de forskjellige variasjonene av Unix førte til svært mange varianter, og dermed forsvant portabiliteten.¹

“The proprietary Unix players proved so ponderous, so blind, and so inept at marketing that Microsoft was able to grab away a large part of their market with the shockingly inferior technology of its Windows operating system.” (Raymond, s.13.)

Computerindustriens ekspansjon og diskusjonene om opphavsrett

Starten av 1990 årene bar preg av de nye personal-computers eller PC ene, som baserte seg på Intels 386 chip. Dette innebar at maskinene ble såpass billige at det var mulig for hackere å skaffe sin egen private maskin, med Internetmulighet og med

1. På begynnelsen av 90-tallet ble det eksperimentert med å overføre BSD Unix source til 386 maskiner, den såkalte Free BSD. Denne bygger på Berkeley Unix sin source kode, og regnes derfor som en egen versjon av Unix.

samme hastighet og kapasitet som de tidligere minicomputere ved univeristetene og terminalrommene ti år tidligere. Raymond beskriver miljøet av microcomputer entusiaster, som delte seg inn i to fraksjoner; DOS anført av Bill Gates, senere Microsoft og Apple, anført av Steve Jobs. Denne store gruppen av hackere ble større enn den før beskrevne nettverks-kulturen av hackere, men de ble aldri selv-bevisste om at de faktisk utgjorde en kultur.

“The pace of change was so fast that fifty different technical cultures grew and died as rapidly as mayflies, never achieving quite the stability necessary to develop a common tradition of jargon, folklore, and mythic history.” (Raymond, s.13)

Diskusjonen om opphavsrett i forbindelse med software utviklet seg som følge av at computerindustrien utviklet seg, eller rett og slett det at flere og flere mennesker tok i bruk og utviklet til datamaskiner. Piratkopiering av software ble et problem for softwareindustrien. Diskusjonen rundt åpen vs. lukket kode, førte til diskusjonen om etiske spørsmål knyttet til opphavsrett. Opphavsretten eller copyright, er skapt for å verne intellektuell kapital. Den proprietære kode forhindret innsikt, og dermed kopiering eller reverse engineering, men den forhindret også programmørens mulighet for å forbedre funksjonaliteten på programvaren.

Professor dr.juris Olav Torvund involverte seg i spørsmål knyttet til edb-programmer og opphavsrett. Han skriver i en artikkel, at opphavsrett for noen kan være vanskelig å gripe, for hvis noen stjeler ens sykkel, så har man ikke lenger noen sykkel. Men hvis noen kopierer en bok man har skrevet så har man fortsatt boken selv. Hvis mange kopierer boken, så forringes ens markedsmuligheter, og hvis en slik praksis er utbredt – så gidder man kanskje ikke bryet med å skrive bøker. Opphavsretten sikrer opphavsmannens enerett til å utnytte verket. Opphavsretten verner et uttrykk, og ikke idéer, kunnskap eller metoder. Sitatet er fra 1997:

“Noen mener at åndsverk, som f.eks. dataprogrammer og litteratur, bør være gratis tilgjengelig. Dette må man gjerne mene, og man må gjerne stille det man selv lager gratis til andres disposisjon. Men ingen kan leve av idealisme, så den som stiller sine verker gratis til disposisjon, må tjene sitt brød på andre måter. Noen må betale det som det faktisk koster. I praksis har de som ivrer for at alt skal være gratis gjerne sin

lønn fra et universitet eller en annen offentlig finansiert institusjon, eller de har f.eks. skrijving eller programmering som hobby.” (Torvund, s.2)

Torvund legger vekt på at den som skaper et åndsverk ikke skaper dette i et vakuum. Opphavsmannen utnytter kunnskap som andre har utviklet tidligere, eller han har forankring i en kultur. Likeledes mener han at det ikke er urimelig at samfunnet får noe tilbake av det som skapes på denne måten, fordi samfunnet ønsker å ta del i den kunnskapsutveksling som finner sted, og også at andre fremtidige opphavsmenn skal kunne utvikle seg videre. Opphavsretten er et forsøk på å balansere opphavsmannens interesser i vern mot samfunnets interesser.

“Den konflikten som for noen år siden verserte mellom Apple og Microsoft om rett til brukergrensesnitt, illustrerer problemet. Apple saksøkte Microsoft med påstand om at Windows krenket Apples rettigheter til McIntosh brukergrensesnittet (saken var mer komplisert, men vi tillater oss en fornklung). Men Apple hadde ikke funnet på alt dette selv. Mye av det som i dag er standard i grafiske brukergrensesnitt, ble i sin tid utviklet ved Xerox' forskningslaboratorium Xerox PARC, som en del av Star-brukergrensesnitt. Mens saken verserte mellom Apple og Microsoft, ble Apple saksøkt av Xerox, med påstand om at McIntosh brukergrensesnitt krenket Xerox' rettigheter til Star-grensesnittet. Dermed kom Apple i en håpløs situasjon: jo mer overbevisende de klarte å argumentere for at Microsoft hadde krenket deres rettigheter, jo mer overbevisende argumenterte de for at de selv krenket Xerox' rettigheter. Verden bør være glad for at ingen fikk medhold at de hadde noen enerett til denne type brukergrensesnitt, selv om noen kanskje kan ha enerett til enkelte elementer i dem.” (Torvund, s.3)

Richard Stallman og Free Software Foundation

Den som imidlertid ofte blir nevnt som den som har lagt grunnlaget for det som senere utviklet seg til Open Source Community, er den omstridte Richard Stallman. Han er elsket og hatet, og blir av noen beskrevet som en altruist og eksentriker, av andre som et geni. Han mente at det at man innen miljøet ikke kunne dele på softwaren, rett ut sagt var uetisk og antisosialt. Det var i frustrasjon over at kildekoden ikke var tilgjengelig, og at man ikke fritt kunne distribuere software at han i årskiftet 1983/1984 startet GNU prosjektet. (www.gnu.org)

“As an operating system developer, I had the right skills for this job. So even though I could not take success for granted, I realized that I was elected to do the job. I chose to make the system compatible with Unix so that it would be portable, and so that Unix users could easily switch to it. The name GNU was chosen following a hacker tradition, as a recursive acronym for “GNU’s not Unix””. (Stallman, s.4)

Stallman er mest kjent for å ha laget GNU Emacs, en standard Unix tekst-editor. I 1985 dannet han Free Software Foundation. Han legger vekt på at termen “Free” ikke må misforstås som “gratis” - som f.eks. gratis øl, men at den gjenspeiler free- som i freedom. Han forsøkte å leve av å produsere free software, mot å distribuere kopier på tape. (Dette var før så mange var knyttet opp til internet.) I en tidligere artikkel “Why Software should be free” argumenterer Stallman for sitt idealistiske og altruistiske syn på softwareutvikling. Han mener at proprietær software gjenspeiler en egosentrisk tankegang:

“If you want to use a program and your neighbor wants to use the program, then in ethical concern for your neighbor, you should want *both* of you to have it. You shouldn’t be satisfied with a solution where you get it and the neighbor does not. Signing a typical software license agreement means betraying your neighbors: “I promise to be unfriendly, I promise to tell my neighbors to get stuffed. To hell with everyone else – just give me a copy! Me, me, me!” People who think this way have become bad neighbors; public spirit suffers”. (Stallman, *Computer Ethics & Social Values*, s.190)

Stallman var først og fremst motstander av at software skulle være lukket, slik at man ikke kunne modifisere den eller finne feil i den. Han var heller ikke direkte motstand av at man kunne få et mindre vedlegg for å distribuere en kopi, det var på den måten han m.fl. finansierte GNU prosjektet. Men allikevel argumenterte han mot eiendomsretten, hvor han mente at det var to hovedargumenter som gikk igjen, ett følelsesmessig og ett økonomisk:

“The emotional argument goes like this: “I put my sweat, my heart, my soul into this program. It comes from *me*, it’s *mine!*” [...] The economic argument goes like this: “I want to get rich (usually described inaccurately as 'making a living'), and if you don't allow me to get rich by programming, then I won't program.” (ibid.)

Free Software Foundation lanserte en ny lisens; den såkalte GNU GPL General Public License. Stallman ønsket å forhindre at software som var fri, skulle kunne benyttes i fremstilling av ny proprietær kode, og derfor oppfant han begrepet “copyleft”. Dette innebar at man med GPL software er fri til å kjøre, kopiere, modifisere og distribuere koden, men at man dermed binder seg til å legge den nye koden som er basert på GNU software inn under GPL lisensen. (Stallman, s.9) Dette er ut fra et ønske om å forhindre at kommersielle aktører skal forsyne seg av den (åpne/gratis) kode som er utviklet i felleskap, for så å skape nye produkter som blir proprietære, eller patenterte. Dette har computerindustrien gjort utallige ganger med free software. Men copyleft regelen medfører at man ikke har lov til å bruke kode som er under GPL lisensen og kombinere denne med andre (åpne eller proprietære) moduler – som er under en annen lisens. Man kan ikke bruke GPL sammen med andre lisenser, selv om dette f.eks. ikke er annet enn at de som har produsert softwaren krever at deres copyright eller nærmest signatur skal følge koden. På samme måte som hvis en forfatter av en artikkel ønsker at hans navn skal fremgå hvis den blir sitert. GPL “smitter”, og blir av noen betegnet som et virus som sprer seg. På den bakgrunn kan man si at GPL er åpen, men i noen tilfeller forhindrer vitensdeling innen systemutvikling. I følge Mads Toftum, forsøker noen “å oppfinne den dype tallerken om igjen”, ved å starte forfra, og lage nye implementeringer av allerede eksisterende god kode, som kun har den “feil” at de ikke er under GPL lisensen, og dermed ikke kan føyes sammen med GPL kode. I følge ham virker det som fullstendig idioti, at man bruker tid og krefter på å kopiere og omskrive en allerede komplett kode, som man har full rett til å bruke, hvis man bare tilføyer opphavsmannens copyright.¹

Linux prosjektet

Operativsystemet lot imidlertid vente på seg. Stallmans Free Software Foundation uteble med sin free Unix kernel, men i 1991 begynte Linus Torvalds, en student ved Helsinki Universitet å utvikle en Free Unix kernel for Intel 386 maskiner, ved hjelp av Free Software Foundations toolkit. (Raymond, s.15) Selv om Linus Torvalds legger vekt på at han har skrevet hele Linux kernelen fra begynnelsen av, og dermed har utviklet et helt nytt operativ system, så begynte han ved å etterligne Minix, men bestemte seg på et tidspunkt for å starte helt fra begynnelsen av og utvikle sin egen kernel. (Tor-

1. <http://www.opensource.org/licenses/index.php>

valds, *The Linux Edge*, s.2) Raymond legger vekt på at det som skiller utviklingen av Linux, fra utviklingen av BSD Unix, og fra Stallmans prosjekter, men ikke minst fra systemutvikling generelt, er det han kaller dens *bazaar* style i utvikling.¹ Det var ikke den tekniske løsningen men den sosiologiske fremgangsmåten som var revolusjonerende for Linux.

“Until the Linux development, everyone believed that any software as complex as an operating system had to be developed in a carefully coordinated way by a relatively small, tightly-knit group of people. This model was and still is typical of both commercial software and the great freeware cathedrals built by the Free Software Foundation in the 1980s.” (Raymond, *A Brief History of Hackerdom*, s.16)

Linux utviklet seg helt annerledes. Fra begynnelsen av, bar utviklingen preg av en vilkårlig hacking fra mange forskjellige frivillige, som koordinerte dette samarbeidet via Internet. Det var på denne tiden flere og flere oppdaget internettets muligheter, og hvor mailing lister, IRK kanaler og newsgrupper spredde seg på nettet. Kvaliteten ble ivaretatt ved hjelp av den enkle strategi, at Torvalds lanserte de nyeste versjonene hver uke. På den måten fikk han feed-back og kommentarer fra hundrevis av brukere over hele verden. I følge Raymond har over 200 personer bidratt til utviklingen av selve kernelen og over 1000 mennesker har bidratt med patches, og feil finning (bug fixes) på Linux.²

Ved slutten av 1993 var Linux i stand til å konkurrere med flere kommersielle Unix systemer, mht. stabilitet og pålitelighet. Linux har siden blitt et kjent merke (brand) som mange forbinder med Open Source Community. En rapport fra Patent og Varemerkestyrelsen og Teknologirådet, viser at Linux er den applikasjonen flest virksomheter og organisasjoner kjenner til. (*Patent og varemerkestyrelsen og Teknologirådet – Anvendelse og utvikling af Open Source Software*, s.36) Over hele verden har det blitt dannet forskjellige Linux grupper, basert på frivillig innsats, hvor man samles rundt utviklingen av forskjellige applikasjoner som blir utviklet rundt kjernen, og hvor man arrangerer lokale foredrag og sosialt samvær. Som jeg var inne på innledningsvis: selve måten å arbeide sammen på er utrolig innovativ, i motsetning til den tradisjonelle meto-

1. Raymond viser til dette på boktittelen, *The Cathedral & the Bazaar*.

2. http://mirror.opensource.dk/halloween/halloween1.php#_Toc427495720

den i softwareutvikling. Det er denne form for vitensdeling, denne nye metoden og organiseringen av systemutvikling, som av mange blir omtalt som en revolusjon. Denne metoden må også ses i sammenheng med den måten internettet utviklet seg på i midten av 90-årene, hvor tilgangen til Internet eksploderte.

“Linux was the first project for which a conscious and successful effort to use the entire world as its talent pool was made. [...] While cheap Internet was a necessary condition for the Linux model to evolve, I think it was not by itself a sufficient condition. Another vital factor was the development of a leadership style and set of cooperative customs that could allow developers to attract co-developers and get maximum leverage out of the medium.” (Raymond, s.51)

Andrew Tridgell (Intervju i *Linux Magazine*, juli 2001) beskriver hvordan han var del av den tidlige fasen av Linux prosjektet. “The Linux code was tiny. It was small enough that you could read it. It was small enough that you could actually get your hand on it and was buggy enough that you could contribute.” (*Linux Magazine*, s.38) Tridgell forteller hvordan de kikket på og kopierte Unix, for på den måten å se hva som trengte å gjøres på Linux kernelen. Tridgell, som er en av dem som har vært med på utviklingen av Linux kjernen, beskriver også at det at hvem som helst kunne delta, satte tonen for det som har utviklet seg til Linux Community. “In those days, Linus was a lot more accepting of changes. He became stricter and stricter, [I forhold til kernelens utforming, *min tilføyning*.] which is a good thing. He really need to be strict these days.” (*Linux Magazine*, s.39)

Idag er det The Linux Community, ledet av Torvalds selv, som vedlikeholder kernelen. Han har utnevnt noen delegater som er ansvarlig for sine områder på prosjektet, og disse igjen har sine koordinatorene. På den måten sikrer han seg at selve kjernen ikke blir utsatt for det man kaller “forking”, mao. at prosjektet - eller rettere produktet – splittes opp i mange forskjellige versjoner. Noen kaller denne form for prosjektstyring et mildt diktatur, andre for enevelde. Mads Toftum mente at det kan være et problem, ut fra et kvalitetssikringsbehov, fordi kun Torvalds har den fulle oversikten, og idag ville ingen kunne gå inn og ta hans plass. I den tiden Linux har eksistert, har det blitt produsert en hel masse applikasjoner som ligger “rundt” Linux kernelen, og det er denne aktiviteten (i tillegg til andre prosjekter) som har vært med på å forme det miljøet som kaller seg Linux Community, som igjen har vært drivkraft for dannelsen av

Open Source Community. Torvalds bestemmer mao. over selve kernelens utforming, men hva folk senere legger til, eller bygger rundt Linux kernelen, det legger han seg ikke opp i. Andrew Tridgell, som selv har startet og drevet et OS prosjekt (Samba prosjektet), forteller i intervjuet med Linux Magazine om bidrag fra miljøet.

“One of the main jobs of the code’s maintainer is to reject bad code, and that’s hard. If somebody sends you a patch that works, but it’s badly coded, you have two choices – re-code it yourself or reject it. That’s hard to do because it offends people.” (*Linux Magazine*, s.40)

Terskelen for å ta med kode er lavere i Samba prosjektet, fordi de ikke har like mye tid og energi som de som bestemmer over kernelen i Linux.

“It takes a hell of a lot of time and energy to reject stuff. You end up exchanging dozens of e-mails, where you say, “that’s bad because of this and this, “ and they say “Oh no, this programming style is great.” Then you have got to teach them a couple of years of computer science so they can understand why it’s crap.” (ibid.)

Det foregår mao. en vitensdeling i form av læring innen miljøet. De mest grunnleggende spørsmål som stilles på newsgrupper, blir ofte henvist til FAQ, Frequently Asked Questions.

Innledningsvis redegjorde jeg for et medierende perspektiv, som legger vekt på at enhver læreprosess er kulturelt betinget. Et verktøy vil i følge Vygotsky tjene som en utadrettet eksternt rettet aktivitet, som kan føre til forandringer i objektet, mens tegnet er rettet innad. Den språklige funksjon (det psykologiske verktøy) – lærer mennesket å beherske seg selv, og den indre representasjonen – forståelsen av hva man tilegner seg, utgjør selve læreprosessen. Det er den eksterne utadrettede erfaring, som går forut for den indre forståelsesprosess, og som er forutsetningen for å lære. Hvis man sammenligner den vitensdeling som foregår i OS miljøer i form av undervisning og læring via bruken av redskaper (programmeringspråk) – og hvor de uerfarne får feed-back fra de mer erfarne, så illustrerer dette også den kulturelle ramme hvor læring finner sted. Subjektet (programmøren) bruker noen verktøyer for å nå noen mål. Vitensdelingen og læringen foregår som en prosess i subjektets interaksjon med omverden. Denne prosess vil påvirke en indre symbolsk prosess, noe som fører til læring.

Linus Torvalds valgte å bruke GPL lisensen på Linux kjernen, men stod overfor problemer knyttet til lisenser. På samme måte som med Unix, kunne andre bidra med moduler, og spørsmålet var om de i tilfelle var underlagt GPL lisensen. Torvalds bestemte seg for at systemkall ikke ville bli definert som en del av kjernen:

“That is, any program running on top of Linux would not be considered covered by the GPL. This decision was made very early on and I have even added a special readme-file ...to make sure everyone knew about it. Because of this commercial vendors can write programs for Linux without having to worry about the GPL” (Torvalds, s.11)

Idag har det dukket oppe en hel masse kommersielle aktører rundt utviklingen av Linux operativsystemet. Den mest kjente er Red Hat Linux, som ved hjelp av sin ekspertise har samlet det de overveier som de beste applikasjonene i en pakkeløsning, og hvor de tjener penger på dokumentasjon, support og konsultasjon. I tillegg har mange av de store aktørene innen computerindustrien som IBM, Oracle og Sun i større og større grad sørget for å gjøre sine egne systemer kompatible, slik at disse også kan kjøre Linux, og dermed utvide valgmuligheten for brukerne/kundene.

Apache prosjektet

Det andre miljøet innen Open Source bevegelsen jeg ønsker å undersøke, er The Apache Software Foundation (ASF). Dette er en paraply-organisasjon, som danner rammen rundt mangfoldige OS prosjekter. Dette prosjektet startet i 1995 av en gruppe webmastere med felles mål. De ønsket å utvikle en web server, og fant ut av at de ville gå sammen om å utvikle en felles kode, istedenfor å drive parallellprosjekter. I likhet med måten Linux prosjektet ble utviklet på, fikk de fordelene av en masse debugging feed-back av andre i miljøet. I November 2002 utgjorde Apache serveren ca. 63% av markedet på webservere, og dette uten legale eiere, uten markedsføring, eller fast organisasjon i ryggen.¹

I 1999 ble The Apache Software Foundation startet, for å danne en organisasjon som kunne støtte forskjellige Apache prosjekter.

1. <http://www.netcraft.com/survey/>

“The mission of the Apache Software Foundation is to facilitate and support collaborative software development projects that use the Apache methods of collaboration over the Internet to create, maintain, and extend the infrastructure of the Web and enforce the standards that define it.”¹ *Roy T. Fielding, the chairman of the ASF*

I starten var det som sagt utviklingen av web serveren (http serveren) som utgjorde Apache prosjektet, men etterhvert har det blitt startet opp flere og flere prosjekter. F.eks. Jakarta prosjektet som med utgangspunkt i Java plattformen har mange mindre sub-prosjekter, som utvikler Java applikasjoner. Andre arbeider med dokumentasjon, andre arbeider med XML. Andre jobber administrativt for selve organisasjonen, som eksisterer som en støtte for hovedprosjektene med sine sub-prosjekter. Apache serveren må hele tiden vedlikeholdes og oppdateres, så dette prosjektet er aldri “ferdig” i den forstand. Hvis man ønsker å starte opp et nytt prosjekt, kan ASF være behjelpelig med dette, hvis ressursene er til rådighet. Hvis man er en del av ASF følger man noen guidelines for hvordan man oppfører seg, og som man har utarbeidet i miljøet. Mads Toftum er selv del av den gruppen som har arbeidet med Apache serveren. Han mener det man ofte i gruppen også har delt, er viten om hvordan man utnytter de muligheter som finnes. Selv om han kanskje ikke bidrar med kode, fordi han ikke har tid, så forsøker han å hjelpe folk igang. Apache kaller sin metode for systemutvikling for et *meritocracy* (merittbasert), og de forskjellige prosjektene har en klar demokratisk utforming. Dette kommer jeg tilbake til. Apache har utviklet sin egen lisens, The Apache Software License, som innebærer at alle som tar softwaren i bruk, må viderebringe selve lisensens ordlyd, som bl.a. innebærer at de skal opplyse om at det er et Apache produkt, samt at de ikke har lov til å benytte navnet 'Apache' i nye produkter.

Dannelsen av Open Source Community

En gruppe mennesker som var tilknyttet the Free Software Community i California distriktet, samlet seg våren 1997 for å diskutere muligheten for å promotere sine idèer til mennesker som unngikk konseptet. De mente at Free Software Foundations anti kommersielle budskap, avhold mange fra å delta i utviklingen og det potensiale som lå i åpen software. Disse initiativtagerne var sterkt påvirket av utviklingen av Linux prosjektet, eller det som utviklet seg til Linux Community. På mange måter skiller man

1. <http://www.apache.org>

ikke mellom det som kalles Linux Community og Open Source Community. Eric Raymond var sterkt påvirket av den måten Linux prosjektet utviklet en ny metode i systemutvikling; the bazaar style, og han var en av disse initiativtagerne, i likhet med Tim O`Reilly, og andre. De ble enige om å starte en kampanje som skulle markedsføre og vinne hva de kalte "mind share". De ble enige om en ny definisjon på den softwaren de promoterte: *Open Source*. (*Open Sources: Voices from the Open Source Revolution*, Introduction, DiBona, Ockman; Stone, s.4) Eric Raymond og Tim O`Reilly er i dag blandt de fremste talsmenn for OSC. O`Reilly er kjent for sitt bokforlag, som utgir computer relatert litteratur, samt deres nettverk med diskusjonsfora, newsgrupper ol. O`Reilly arrangerer også konferanser og foredrag, som typisk tar opp emner knyttet til Open Source Community. Hele den historiske dannelsen av OSC har vært preget av en utvikling hvor man først har forsøkt å utvikle basale teknologier, som har vært et alternativ til proprietær kode. Slik som f.eks. operativsystem, webservere, filservere, databaser ol. Etter hvert som disse prosjeketene har etablert seg, og etter hvert som fler og fler kommersielle aktører benytter OS i sin forretningsmodell, går tendensen i retning av mer brukervennlige systemer. De to store operatørene innen PC markedet: Apple og Microsoft, har satt standard for det interface vi bruker i dag (GUI - Graphical User Interface). Innen OS har det utviklet seg to teknologier og dermed to retninger som leverer GUI software; KDE og GNOME prosjektet.¹

Dannelsen av Open Source Community, og dennes mer pragmatiske holdning og ønske om å tiltrekke seg kommersielle deltagere, utviklet seg til et spørsmål om lisenser. Som jeg var inne på tidligere innebærer GPL lisensen at man kan se koden og man kan bruke den, men det finnes restriksjoner for hva man kan gjøre. På dette grunnlag ble the *Open Source Definition* dannet.² Denne definisjonen tillater større frihet enn GPL lisensen gjør, og den tillater også en større sammenblanding av proprietær kode og Open Source software. En felles organisasjon: Open Source Initiative (OSI) er dannet for å ivareta definisjonen, informere og formidle de forskjellige lisenser som er godkjent av OSI. GPL lisensen er fortsatt populær og brukes av mange. Den andre mest kjente varianten er BSD lisensen. Denne tillater at du benytter deg av den åpne koden, og at du kan gjøre med den hva du vil. Om du så ønsker å bidra *tilbake* til Open

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

2. Etter initiativ av Bruce Perens.

Source Community med det du har skapt (ved hjelp av OS software) blir mer en kulturell norm enn noe du forplikter deg til. I dag foregår det flere diskusjoner innen miljøene knyttet til lisenser. De som er knyttet til Stallmans fraksjon, legger stor vekt på de altruistiske verdispørsmål knyttet til åpen kode:

“Teaching new users about freedom became more difficult in 1998, when a part of the community decided to stop using the term “free software” and say “open source software” instead. Some who favored this term aimed to avoid the confusion of “free” with “gratis”—a valid goal. Others, however, aimed to set aside the spirit of principle that had motivated the free software movement and the GNU project, and to appeal instead to executives and business users, many of whom hold an ideology that places profit above freedom, above community, above principles. Thus, the rhetoric of “Open Source” focuses on the potential to make high quality, powerful software, but shuns the ideas of freedom, community and principle.” (Stallman, s.21)

Mads Toftum mener at for noen innen OS miljøet, er åpenheten absolutt den viktigste motivasjonsfaktor for å være del av OS miljøet, og ikke om software er gratis eller ikke. F.eks. innen kryptografi er det utrolig viktig at det ikke finnes feil. Da er åpenheten et spørsmål om tillit. Hvis produsentene ikke kan vise presis hvordan de har sørget for at et program er sikkert, så kan det gjemme seg enn feil i programmet. Hvis man ikke kan se koden, så er man henvist til å skulle teste og forsøke å bryte systemet for å finne eventuelle feil, og bli overbevist om at det virker. Hvis folk kommer med en sort kasse – så er det viktig å få mulighet for innsikt i den sorte kassen; om man så skal betale for programvaren er mindre viktig for ham. For andre er det Toftum kaller “religionskrigen” det viktigste. Da er hovedmotivasjonen for OS at man synes det er galt at noen få store selskaper, som.f.eks. Microsoft, skal ha monopol på programvare, og derfor søker man etter alternativer.

Det som bekymrer mange innen Open Source miljøet, er patentering. I motsetning til copyright, som beskytter et uttrykk, beskytter patentet en spesifikk metode/prosess. “Copyright and patent protect different things. Copyright protects expressions but not underlying ideas. Patents protect useful processes, machines, and compositions of matter.” (Brian Kahin, *Computes, Ethics & Social Values*, s.213) En rapport som er utgitt av Patent og varemærkestyrelsens rapport om Open Source i Danmark, blir kritisert i sterke ordelag av SSLUG (Skåne og Sjælland Linux Group).

“Fra SSLUG's side er vi af den opfattelse at det skal være tilladt at utvikle fri og gratis software inden for alle softwareområder, men SSLUG har aldrig fremført at al software skal være fri og gratis, og heller ikke at fri og gratis software alltid er bedre end proprietær software. Men fri software skal have mulighed for at konkurrere med proprietær software på like markedsvilkår. Softwarepatenter truer med at forskubbe denne balance fra hvem der laver den bedste software, til hvem der har de fleste advokater. Og i en sådan patent monopoløkonomi vil både Open Source-bevægelsen og de mange små danske softwarefirmaer lide skade.”¹

Flere store selskaper er svært ivrige og aktive etter å patentere software, hvis de ved hjelp av denne software fremstiller en ny måte å utnytte software på, slik f.eks. software gjorde at mange begynte å utnytte datamaskinen *som om* den var en skrivemaskin. Kritikerne av patentstyrelsens rapport stiller bl.a. spørsmålet:

“Hvordan undgås at softwarepatenter kommer til at true OSS? Betyder softwarepatenter at man overlader det til et mindre antal multinationale selskaper at råde over Open Sources-bevægelsen og den danske IT-branches fremtid.” (ibid.)

SSLUG mener at de forskjellige Open Source miljøer, i denne rapporten har blitt fremstilt som om de generelt er motstandere av copyright. Dette er en av de påstander som de tar til motmele mot. De fleste innen Open Source miljøet føler seg beslektet med den akademiske tradisjon om at man viser andre hva man har laget, men man gir aldri seg selv kreditt for noe man ikke selv har skapt; mao. så oppgir man sine kilder. Dette er også den mest vanlige klausulen i de forskjellige lisenser, navnet på opphavsmannen skal følge med koden.

Oppsummering

I denne delen har jeg analysert bakgrunnen for dannelsen av det som idag kalles the Open Source Community, som samtidig blir en fremstilling av forskjellige former for vitensdeling som springer ut av dette miljøet. Denne utvikling må ses som en kombinasjon av forskjellige faktorer. Den tidlige hacker kulturen var preget av en tankegang som kjennetegner de fleste vitenskapelige miljøer; man deler det man vet med andre. I

1. http://www.sslug.dk/patent/DKPTO_OSS_rapport_kritik/index2.shtml

tillegg har internettets utvikling gjort det mulig å dele viten og arbeide sammen på nye måter. Softwareproduksjon er knyttet til spørsmål om copyright og intellektuelle rettigheter. Åpenheten og det at software industien utviklet kommersielle produkter, skapte diskusjoner i forhold til rettigheter og lisenser. Denne diskusjonen er fortsatt like aktuell, fordi noen lisenser gir enhver rett til å gjøre hva man vil med koden, mens særlig GPL i noen henseender kan forhindre vitensdeling. Det kan være fristende å illustrere denne debatten som fortsatt er levende innenfor miljøet, ved å sette opp to dramaturgiske motsetninger: *antagonisten* – representert ved Bill Gates og Microsoft vs. *protagonisten* – representert ved Richard Stallman. Open Source Community føyer seg inn i mellom dem, som en pragmatisk alternativ til de to ytterligheter. Open Source Community har forsøkt å skape avstand til Stallmans idealisme, som kommer til uttrykk i GPL lisensen, på grunn av de begrensninger som ligger i hans idealistiske og anti-kommersielle budskap.

Selve den utvikling som har foregått innen programmeringsspråk, og den måten Unix var utformet på, oppfordret til utveksling og deltagelse fra flere deltagere. På mange måter var Linux prosjektet en gjentakelse av den tidlige hacker kulturen rundt Unix. Dette viser at vitensdeling innen systemutvikling også er knyttet til spørsmål om kompleksitet.

Videre forteller utviklingen om Linux og Apache prosjektet, hvordan Internet muliggjorde en ny metode innen systemutvikling, som bærer preg av en utpreget form for vitensdeling. Denne form for vitensdeling og metode - via en kontinuerlig feed-back fra mange deltagere - har gjennom årene blitt mer og mer organisert. Denne metoden kan sees som et alternativ til den tradisjonelle software-engineering tradisjonen i softwareutvikling.

Videre har jeg vist at systemutvikling kan være innovativt, men at det ofte innebærer å kopiere hva andre har gjort. Vygotsky åpner opp for en tankegang, hvor man forstår verktøy og språk som prosesser, som ledd i menneskets gjenstandsrettede aktivitet og hvor disse prosessene er del av den kulturhistoriske utvikling. OSC gir mulighet for mangfoldige læreprosesser; man deler kode, man deler viten, men ikke minst lærer man opp nye deltagere i miljøet.

I følge Ellen Christiansen¹ eksisterer det i vår tid et aristotelisk syn på vitensdeling, som tar utgangspunkt i språk og verktøyer som gjenstander og objekter. Man lokker f.eks. de ansatte med bonus: “hvis du legger ditt dokument ut på intranettet, så skal du få en liten bonus.....”. Vygotsky forstår språk og redskaper som prosesser, dvs. som ledd i menneskets gjenstandsrettede virksomhet, og som ledd i den kulturhistoriske utvikling. I følge Ellen Christiansen skaper Vygotskys teori en kopernikansk vending i synet på vitensdeling. Man tror, (f.eks. ved å lokke med bonus for å dele viten) – at viten er noe folk besitter og har “innenfra” og som man deler “ut” til de andre, f.eks. i form av dokumenter. Vygotsky viser hvordan denne utveksling starter i det eksterne, og at man først og fremst må tenke på dette som prosesser som mennesker kan delta i og utvikle seg (lære) ved. Han legger vekt på hvilken viktig forutsetning kulturen og de sosiale rammer er, for at slike læreprosesser skal finne sted. Slik jeg ser det, danner OS miljøene rammen for en kultur, hvor læreprosesser og læring finner sted

Idag utgjør OS miljøene ett alternativ til den proprietære softwareutvikling. Svært mange programmører arbeider innen den kommersielle softwareindustri, samtidig som de er deltagere i OSC. En omfattende læring finner sted i OS miljøene, samt mulighet for å få hjelp til å løse problemer når man trenger det. Denne form for vitensdeling, må sies å komme industrien til gode i form av oppdaterte medarbeidere.

Innenfor miljøet har man nå tatt opp en ny diskusjonen om samfunnets vs. store multinasjonale selskaper interesser. Man retter fokus mot medisinal industrien, som tar patenter på f.eks. menneskelige gener. OSC stiller seg spørsmålet om det er riktig at noen få store selskaper skal eie den form for viten om mennesket selv. Er det i samfunnets interesse at viten fra denne forskningen ikke er tilgjengelig for samfunnet? Igjen er det motsetningen mellom offentligheten og den vitenskapelige tradisjon vs. store selskapers interesser, som blir satt opp mot hverandre. Det som startet som en form for vitensdeling innen systemutvikling har mao. endt opp i et miljø som promoterer denne form for åpenhet og vitensdeling i andre deler av den teknologi-drevne industrien. OSC har spredd seg til å gjelde prosjekter som forsøker å kartlegge det menneskelige genome. Dette prosjektet går ut på å vinne kappløpet om kartleggingen av de

1. Jeg har tatt meg den frihet å referere fra forelesningsrekker og samtaler med E.C. på bakgrunn av hva jeg kan huske fra mange spennende diskusjoner knyttet til vitensdeling.

menneskelige gener, for dermed å sikre at denne viten kommer offentligheten til gode og ikke skal eies av kommersielle aktører.

Miljøet har det felles, at de utgjør en kultur, et internasjonalt nettverk av mennesker, som arbeider sammen på prosjekter over lang tid, kanskje uten noensinne å møte hverandre ansikt til ansikt. I det neste avsnittet vil jeg beskrive Open Source miljøene ut fra et kulturelt og samfunnsmessig perspektiv.

Del 3

Verdisystemer innen Open Source kulturen; merittbasert ridder-kultur og pragmatisk fellesskap i cyberspace.

Innledning

I dette avsnittet vil jeg komme med en beskrivelse av OSC, ut fra en antropologisk tankegang. Som jeg tidligere har vært inne på, er det forskjell på hva folk sier de gjør, tror de gjør og faktisk gjør. Ved å kartlegge hva folk faktisk gjør, mao. den observerbare adferd, vil jeg forsøke å finne frem til, og beskrive de underliggende verdier, mao. verdisystemene, som kjenntegner OS miljøene. Jeg vil anvende Batesons schismogenese og ethos begrep som forklaringsmodell. Min beskrivelse bygger på innsamlet materiale i form av observasjoner, diskurs, intervjuer og samtaler. Innledningsvis vil jeg gjøre rede for Eric Raymonds fremstilling i essayet *Homesteading the Noosphere*, fordi jeg i dette avsnittet anvender Raymonds antropologiske fremstilling som kilde for å underbygge min beskrivelse.

Sammenfatning og presentasjon av Raymonds fremstilling

I de foregående delene har jeg ofte referert til Eric Raymonds *The Cathedral & The Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary*. Denne boken er en samling av essays han har skrevet over tid, som omhandler hans observasjoner rundt dannelsen og utøvelsen av hacker kulturen og Open Source miljøene. Raymond, som selv er antropolog, veksler i rollen som deltager og observatør i sin fremstilling. Han har selv vært en aktiv pådriver i etableringen av det som i dag fremstår som Open Source Community, (jmf. del 2) hvor en gruppe hackere som var del av FSF miljøet gikk ut og dannet sin egen fraksjon, hvor man ønsket å skape en avstand til Stallmans anti-kommersielle budskap. I essayet *Homesteading the Noosphere*, tar Raymond utgangspunkt i motsetninger mellom den offisielle ideologien som er definert i OS lisensene, (diskursen) og hva folk faktisk gjør. Han definerer forskjellige ideologiske

syn som etter hans mening er drivkraften for hva som motiverer folk til å ta del i kulturen, og som igjen er grunnlaget for forskjellige sub-kulturer i miljøet.

“One degree of variation is zealotry; whether open source development is regarded merely as a convenient means to an end (good tools and fun toys and an interesting game to play) or as an end in itself.” (Raymond, s.68) Disse variantene av fanatisme gjelder folks motivasjon for å utvikle software ut fra et ønske om bedre verktøyer. Den andre ideologien er drevet ut fra folks forskjellige syn på kommersiell softwareutvikling. De helt anti- kommersielle personene kunne finne på å si: “Commercial software is theft and hoarding. I write free software to end this evil” (ibid.) Raymond mener at det finnes ekstreme, pragmatiske og moderate syn, som utgjør forskjellige sub-kulturer i miljøet. Enten man deltar pga. det faglige behov, eller ut fra forskjellige samfunnssyn/interesser. Videre analyserer Raymond hvilke skikker som styrer eierskap og kontroll innen OS prosjekter. Han sammenligner og viser hvordan den Lockeanske teorien om land-besittelse, ligger til grunn for et syn på eiendomsretten også innen Open Source prosjekter. Likeledes viser han at hacker kulturen kan sammenlignes med en gave kultur; hvor deltagerne konkurrerer om prestisje ved å gi fra seg tid, energi og kreativitet. Raymond undersøker hvordan konsekvensene av disse analysene kan fortelle noe om konfliktløsning i kulturen, og han kommer med forslag til hvordan man kan utvikle noen retningslinjer for kulturen i fremtiden. Han stiller spørsmål om hvorfor noen prosjekter lykkes og andre ikke, og sammenligner med den Anglo Amerikanske historiske utvikling, som baserte seg på Europeiske tradisjoner og samfunnsutvikling. Dette kommer jeg tilbake til. I annen del nevnte jeg en dramaturgisk modell, hvor antagonist Microsoft med Bill Gates ble stilt overfor protagonisten Richard Stallman og Free Software Foundation, og hvor disse utgjør to motpoler. Enten man skiller den underliggende ideologi og drivkraft for deltagelse i disse miljøene ut fra et syn på økonomi (kommersiell software vs. free software) eller ut fra et ønske om bedre verktøy (åpen software vs. lukket software) er det noe som er felles for denne kulturen, noe som skiller oss fra *dem*. Man kan nesten ikke diskutere OS kulturen, uten å redegjøre for de diskusjoner og regelrette kamper som har foregått mellom dette miljøet og Microsoft. Det neste avsnitt vil bli en redegjørelse for hvorfor Bill Gates og Microsoft har endt opp i rollen som antagonist.

Hva er felles for miljøet? Kampen mot Microsofts monopol

For å forstå den kampen som foregår mellom OSC og Microsoft, må man igjen tilbake til operativ systemers betydning. Microsofts operativsystem, er selve kronjuvelen i hele virksomheten, for det er dette operativsystemet som kjører hele Office pakken, og som dermed gjør Microsoft til verdensdominerende innen pc-desktop programvare. Microsoft, i likhet med IBM, startet opp med et operativsystem kalt MS-Dos. Dette regnes for å være et enkelt operativsystem. Microsoft utviklet Win 3.1 og IBM utviklet operativsystemet OS-2. Partene inngikk et samarbeid om å utvikle et nytt operativsystem, som kunne ta høyde for bl.a. det som kalles multitasking, dvs. at systemet kan kjøre flere parallele programmer samtidig. Dette samarbeidet pågikk rundt slutten av 80-tallet. IBM var dengang den største aktøren innen computerindustrien. De samarbeidet om utviklingen av en nytt operativsystem/plattform, og IBM ga dermed av sin viten til Microsoft med Bill Gates i spissen. Han valgte å bryte samarbeidet, fordi han ved hjelp av denne viten fra IBM hadde sideutviklet sitt eget operativsystem kalt Windows 95 (til desktop) og et som kaltes Windows NT 3.X. (NT står for New technology). Dette operativsystemet var tilpasset Office pakken, og det var fra da av Microsoft klarte å tilegne seg pc-markedet, og har vokst til den giganten de er i dag. Microsofts operativsystem dengang, var kjent for å være ustabil og en ganske primitiv teknologi, men det Microsoft var i besittelse av, var en gigantisk markedsføringsevne. De klarte å overbevise markedet om Windows 95's fortreffelighet. Store deler av operativsystemet er lukket kode, noe som vil si at man for å klare å få andre programmer eller plattformer, som f.eks. en Mac, til å virke sammen med Windows, så var man nødt til å debugge koden, å forsøke å finne ut av hvordan den var satt sammen, for å få dette til. Dette kalles også for reverse engineering. Samba prosjektet, som tidligere er omtalt, er et eksempel på dette. Tridgell startet opp dette prosjektet fordi han ville forsøke å få en Sun og en DEC maskin til å kunne virke sammen med Windows. Han debugget Microsofts SMB filesharing protocol.¹ Når nye programmer etterhvert kom på markedet, ble det Microsofts taktikk, å sørge for at deres egne produkter kunne kjøre på systemet uten problemer, mens annen programvare hadde vanskeligheter. Resultatet har blitt at når de fleste i en organisasjon har brukt Windows, og man hele tiden har hatt problemer med f.eks. å få en Mac til å kunne

1. Etter anti-trust rettsaken i Amerika, har Microsoft bestemt seg for å gi ut sine fildelingsprotokoller.

virke sammen med pc-ene pga. Microsofts bevisste valg om å gjøre dette vanskelig – så har man kjøpt flere Windows produkter. På denne måten har Microsoft nærmet seg monopol på programvaremarkedet.

Mytene og historiene rundt Bill Gates er mange. En av mange rettsaker i Amerika illustrerer det mange mener er hans personlige taktikk og forretningsmetoder. Et lite selskap, kalt Burst.com har utviklet teknologi for formidling av video og lyd over IP-nettverk, og de sitter på en rekke patenter og varemerker. Firmaet har hatt problemer med å inngå avtaler med andre virksomheter, og i søksmålet påstår de at Microsoft har presset andre aktører til ikke å inngå avtaler med dem. Videre påstår de at Microsoft forhandlet med Burst.com i nesten to år i 1999 og 2000, og at de dermed satte seg grundig inne i teknologien før de avsluttet forhandlingene med et latterlig tilbud på en million dollar for evig å alltid å sitte på eksklusive rettigheter til alle Burst.coms patenter.

“Som følge av denne innsikten i Burst-teknologien, skal Microsoft ha innarbeidet den i sin egen videospiller Corona, som ble demonstrert i fjor og som det er meningen å innlemme i Windows Media Player, som en integrert del av Windows XP. Microsoft skal dessuten ha gjort Corona inkompatibel med andre implementeringer av Burst-teknologien.” (www.digi.no, Erik Rossen, 20.6.2002)

Utfallet av denne rettsaken vites ikke, men hvis det viser seg å være riktig, viser det en hensynsløs forretningsskikk mot mindre virksomheter.

Bill Gates og Microsoft hadde ikke forutsett eksplosjonen på Internet markedet. I 1994 ble Netscape Communication etablert, som er kjent for sin web-browser som tok en stor del av markedet. De brukte åpne Web standarder, noe som vil si dss. at uansett hva slags plattform man bruker, så fungerer softwaren/browseren. Hele tanken med World Wide Web, var jo nettopp å få forskjellige maskiner til å arbeide sammen uansett plattform. Dette førte til at Microsoft utarbeidet og lanserte sin Internet Explorer, som tok markedsandeler fra Netscape. Det de imidlertid også gjorde, og som har ført til den velkjente anti-trust rettsaken i Amerika, er at de brukte sin egen web protocol, (mao. ikke den åpne standard) og dermed tvang brukerne inn i den proprietære standard som kun Microsofts operative systemer kan betjene. Dette viser seg i praksis ved at det er enkelt å lage en web side som virker på Windows, men

hvis man bruker andre operativsystemer, eller f.eks. en Mac – så ser den annerledes og litt merkelig ut. Hvis man i dag skal produsere en web side, bør man sjekke den opp mot forskjellige browsere, for å unngå dette. I oktober 2001, valgte Microsoft å nekte andre browsere som Mozilla (som er en Open Source browser) og Opera (som er en liten norskprodusert browser) adgang på sine egne store portaler MSN.com, som bl.a. har tjenesten hotmail ol. På denne måten bruker Microsoft sin markedsrett til å utestenge irriterende konkurrenter. De påstod at Opera ikke brukte de åpne standardene og at de stengte disse brukerne ute fordi MSN.com ble “best vist” med Internet Explorer. Få dager etterpå trakk Microsoft sperringen tilbake, men i følge Ryvarden viste de en arrogant tendens, til tross for de pågående rettsaker som nettopp satte fokus på slik praksis.

“Utesperringen av Opera viser et nesten utrolig hovmod. For Opera følger de offisielle Internett-standardene bedre enn Microsoft selv. En kjapp test med W3Cs egen HTML-sjekker viser at Microsoft enten er dårlige til å programmere i HTML eller ikke vil følge standardene. Kanskje Opera kunne tilby et lite kurs?” (Einar Ryvarden, www.digi.no, 26.10.2001.)

Ryvarden mener at denne praksisen i sin ytterste konsekvens er med på å splitte det som idag kalles Internet. Det er nettopp de enkle HTML-standardene og plattform uavhengigheten som har skapt Internet og gjort det til det det er i dag. Ved å sperre noen brukere ute, vil Microsoft få resten av brukerne til å bytte til Internet Explorer.

Mange Macintosh maskiner har i dag et nytt operativsystem som kalles Mac OSX. Innebygd i dette systemet finnes et program som kan streame på Internet, slik at man kan høre radio på direkten. Hvis jeg f.eks. skal høre på norsk radio, kan jeg gå inn på NRK sine hjemmesider, og klikke på en link som Mac brukere skal benytte seg av, og dermed kan jeg bruke mitt eget lyd-program. NRK har mao. laget sin tjeneste slik at den virker mot mange forskjellige plattformer. Hvis jeg derimot klikker meg inn på DR, så finnes ikke denne muligheten. Da må jeg laste ned Microsofts egen standard for lydinnspilling, og andre muligheter finnes ikke. Man kan undre seg over hvor lite bevisste mange web designere er overfor denne problematikken. Denne form for ensretting går under betegnelsen *Microsoft infisering*.

De som forstår denne utvikling, mao. hackerne, og som gjennomskuer den, har vært vitner til hvordan Microsoft har tilegnet seg noe som nærmer seg monopol. Samtidig står de gang på gang overfor problemer knyttet til Microsofts operativsystemer, fordi annen software ikke virker, og fordi Microsoft ikke vil gi den innsikten som er påkrevd for å gjøre annen software kompatibelt. Vanlige mennesker og brukere forstår ikke hva som er hva, eller hvordan ting foregår i et operativsystem. Det eneste man forstår er at Internet Explorer “virker” – og at andre ting ser merkelige ut, og dermed “ikke virker”. Netscape forsøkte å motvirke Microsofts monopol, ved å gi fra seg koden til sin browser, og gjøre den til Open Source, uten at dette har lyktes. Man kan vel si at Microsoft har vunnet denne kampen over Netscape, fordi de sitter på en så stor del av pc-markedet. Det var på denne bakgrunn amerikanske myndigheter gikk til rettsak mot Microsoft. I en separat stevning har 20 delstater i Columbia distriktet saksøkt Microsoft, fordi de mener at de har gjort seg skyldig i en rekke lovbrudd.

“I den føderale stevningen søker justisdepartementet en foreløping forføyning som skal tvinge Microsoft til å inkludere konkurrenten Netscape sin nettleser for å gi forbrukere et valg i tillegg til Microsofts Internet Explorer.[...] Microsoft skal ha forsøkt å snyte PC-eiere for fordelene med et fritt og konkurransepreget marked ved å utnytte sin sterke posisjon innen operativsystem til å dominere nettlesermarkedet. De saksøkende parter hevder at konsekvensen av Microsofts praksis kan bli at selskapet til slutt får absolutt kontroll over programvareindustrien. Det som kan se ut som bra for konsumentene, kan ende opp som et mareritt, i henhold til en talsmann for delstatene.”(Jens Kamden, www.digi.no, 19.5.98)

Det har versert mange og kompliserte rettsaker mot Microsoft i en årrekke, og de har blitt pålagt å skille ut sitt operativsystem fra resten av Office pakken, for å unngå denne praksisen. Diskusjonene fortsetter, og man må vel kunne si at Microsoft er meget motvillig til å åpne opp for innsikt, eller gi slipp på sitt monopol.

Halloween dokumentene

På slutten av 90 tallet begynte Linux å bli tatt i bruk av flere og flere brukere. På samme tid jobbet man på spreng i Microsoft for å utarbeide og lansere Windows 2000 (XP kom som en bedre utgave rett etterpå,) som er et moderne operativsystem. Denne utviklingen var forsinket og lanseringen ble utsatt flere ganger. Linux ble av

Microsoft oppfattet som en trussel, og denne frykten førte til ett av de tilfeller som virkelig har satt sinnene i kok innen Open Source miljøet; de såkalte *Halloween Documents*. Dokumentene er signert 11 august 1998, og var undertegnet av Vinod Valloppillil, en product manager hos Microsoft. Eric Raymond mottok kopier av disse interne dokumentene fra tre anonyme kilder, og lanserte dem på nettet i begynnelsen av november samme år, med rasende kommentarer.¹ Dokumentene er skrevet som et internt notat, som analyserer Linux operativsystemet og hvilken fare Linux miljøet utgjør for Microsoft. Dokumentene ble meget omtalt i den internasjonale pressen.

“OSS [Open Source Software] poses a direct, short-term revenue and platform threat to Microsoft, particularly in server space. Additionally, the intrinsic parallelism and free idea exchange in OSS has benefits that are not replicable with our current licensing model and therefore present a long term developer mindshare threat.”²

De avslører også Microsofts strategi og motreaksjon mot Open Source bevegelsen. Dokumentet viser at man anser Open Source software som så god (i 1998), at den utgjør et alternativ til kommersiell software “the ability of the OSS process to collect and harness the collective IQ of thousands of individuals across the Internet is simply amazing”, (ibid.) og konkluderer med at Linux er vanskelig å sette ut av spill med samme taktikk som Microsoft har brukt mot tidligere konkurrenter. Hvis man skal bekjempe Linux står man ikke overfor en konkurrent i den forstand; men man må bekjempe en idé. “...to understand how to compete against OSS, we must target a process rather than a company.” (ibid.) Grunnen til at Linux er så mektig, er i følge Valloppillil, at dens tekniske protokoller: byggeklossene, er gratis, åpent distribuert og blir ikke eid av noen. Den eneste måten å bekjempe Linux på, er å late som om man adopterer protokollene, for så å “utvide” dem på en slik måte at de kun virker på Microsofts proprietære kode. På den måten vil nye versjoner av Microsoft bli inkompatible med de åpne standardene.

“OSS projects have been able to gain a foothold in many server applications because of the wide utility of highly commoditized, simple protocols. By extending these pro-

1. The Halloween Documents ble først lagt ut og kommentert på Raymonds egne hjemmesider, men er nå flyttet over til Open Source's egne hjemmesider.

2. <http://www.opensource.org/halloween/halloween1.htm>

protocols and developing new protocols, we can deny OSS projects entry into the market.” (ibid.)

Det er nettopp denne praksisen fra Microsofts side som gjør OS miljøene rasende. Microsoft innrømmet at dokumentene kom fra dem, men mente at dette var et internt notat, skrevet av en ansatt ut fra sitt ståsted som product manager og derfor ikke skulle tolkes som Microsofts offisielle strategi. Imidlertid er det ikke tvil om at Microsoft ser på Open Source Community som en trussel, og ikke en medspiller, slik mange andre større aktører i IT bransjen har valgt å gjøre. Tim O`Reilly påpeker dette i et åpent brev til Microsoft:

“The point that you seem to miss is that it is these simple, commoditized protocols and a culture of building freely on the work of others that brought us the explosion of innovation known as the Internet. And while the Internet has opened new areas of competition for Microsoft, it has also opened up enormous opportunities.[...] Internet-enabling Windows and Office has been the major source of new features that make it worthwhile for customers to buy new systems or upgrade their applications. [...] And now you want to undermine Open Source? Try to be serious!”¹

Observerbar adferd (schismogenese)

Relasjoner – De som er innenfor og de som er utenfor. Hvordan kommer man inn?

Open Source miljøene har oppstått som følge av at enkeltpersoner har tatt initiativ til å sette igang et prosjekt. Noen har blitt berømte, slik Linus Torvalds har blitt det.

For å kunne beskrive relasjonene mellom de som er innenfor vs. de som er nye, må man undersøke hvordan det foregår når man blir deltager i et prosjekt. Dette igjen henger sammen med måten prosjekter blir styrt på, og måten et prosjekt blir styrt på har med eierskap til prosjekter å gjøre. Eierskap kan i denne kulturen ikke knyttes til materielle eller økonomiske forhold; man snakker om en eiendom (programkode) som kan omformes og duplikeres, uten noen utøvende maktinstitusjoner.

1. http://press.oreilly.com/tim_msletter.html

“Actually, in the case of the open-source culture this is an easy question to answer. The owner of a software project is the person who has the exclusive right, recognized by the community at large, to *distribute modified versions*.” (Raymond, s.73)

Dette behøver ikke være en enkeltperson, men kan også være en gruppe av mennesker. I følge Open Source lisensen er alle likeverdige parter, men i praksis finnes det et klart skille mellom offisielle 'patches' som er lansert av de som styrer og driver prosjektet, samt 'rogue' patches, utarbeidet av tredje part. Disse er sjeldne og blir vanligvis ikke regnes for seriøse. “It is only when modifications are posted to the open-source community in general, to compete with the original, that ownership becomes an issue.” (ibid.)

Raymond mener at det finnes tre måter å kreve eierskap på; ved å starte ett eget prosjekt, eller ved at en grunnlegger gir prosjektet videre til en annen person eller gruppe, og den tredje måten er å kreve eierskap på et prosjekt som ligger brakk, og hvor eier/eierne har forsvunnet eller ikke viser interesse lenger. I dette tilfellet må man først forsøke å oppspore eieren, eventuelt annonsere dette initiativ på en kjent news-gruppe. Dessuten er det vanlig skikk & bruk å vente en stund før man annonserer seg selv som den nye eieren. Tidligere har jeg vært inne på, at en ting er hva folk sier de gjør, en annen ting er hva folk faktisk gjør. Raymond påpeker at det eksisterer en motsetning i miljøet, for selve GPL lisensen legger opp til at man kan ta et stykke kode, og gjøre med den hva man vil. Mao. er dette en form for diskurs som sier at det er fritt frem å gjøre hva man vil. Men allikevel eksisterer det en konsensus i miljøet om at slik gjør man bare ikke. Man går ikke ut, og tar Linux kjernen og lager en ny “offisiell” versjon. Det er dette som kalles *forking*.

Hvis man skal kunne kategorisere relasjonene mellom de som er innenfor vs. de nye i miljøet, har jeg valgt å eksemplifisere disse, ved å dele dem inn i tre hovedgrupper.

1) Linux prosjektet (her snakker vi om selve kjernen). I artikkelen *The Linux Edge*, (*Open Sources: Voices from the Open Source Revolution*) beskriver Torvalds hele utviklingsforløpet med Linux, og bakgrunnen for sine valg i utviklingsfasen.

“With a monolithic kernel such as the Linux kernel, it`s important to be very cautious about allowing new code and new features into the kernel. These decisions can effect

a number of things later on in the development cycle beyond the core kernel work.”
(Torvalds, s.6)

Torvalds valgte en strategi for å unngå komplikasjoner senere i utviklingsfasen. Han selv, samt noen få utvalgte deltagere, vokter fortsatt over selve kjernen i Linux for å unngå at prosjektet skal dele seg i mange del-prosjekter, med dertil varierende versjoner. Denne form for prosjektstyring blir av noen kalt mildt diktatur, eller enevelde. Dette har vært en nødvendighet for å holde prosjektet intakt, og fordi den tekniske utformingen på et operativ system er såpass komplekst. Det er Torvalds selv, og hans “nestkommanderende” Alan Cox, som har kontroll over nye versjoner i Linux gruppen. Noen få personer sitter i et slags mellomnivå, og det er disse som tar i mot forslag fra utenforstående. (Jmf. del 2, om læring.) Mao. kommer man som utenforstående ikke inn og får innflytelse på avgjørelser, men man må gjerne komme med forslag. Relasjonene mellom deltagerne rundt utviklingen av Linux, vil jeg karakterisere som komplementære. Dette er mao. et spørsmål om kompleksitet, men også om at Linux prosjektet fortonet seg annerledes før enn nå, hvor det først og fremst handler om det å unngå forking i den videre utviklingsfasen.

2) Apache gruppen. Mads Toftum fortalte om hvordan man forholder seg til nyankommere som har lyst til å delta i den Apache gruppen han er del av. Hvis noen ønsker å være med, så kan de bli bedt om å komme med noen patches/forslag. Det som så ligger implisitt i en slik utmelding, er at man på den måten kan se hva en person står for, eller om man kan stole på denne personen. Man vurderer dermed hva personen kan bidra med på prosjektet. Det kan være mange måter å bidra på, som f.eks. kode, det kan være dokumentasjon, det kan være folk som kan hjelpe til med å drive prosjektets infrastruktur. (Man kommuniserer v.h.a. development mailinglister og newsgroups.) Hvis personen så blir opptatt i gruppen, har han/hun stemmerett på linje med de andre. Man opererer med de før nevnte guidelines, eller 'skikk og bruk', som gir opplysninger om hvordan man skal forholde seg til stemmeavgivning, god takt og tone ol. Dette er en form for demokratisk prosjektstyring. Relasjonen mellom de som er med vs. nye bærer preg av å være komplementær inntil man eventuelt blir tatt opp i gruppen, og dermed oppnår samme rettigheter (om ikke samme status) som de andre.

3) Mindre prosjekter er ofte satt sammen av noen få mennesker som blir enige om å lage et eller annet. Måten å styre prosjektene på er basert på common sense. Fordi det ikke er noen klar leder av prosjektet, slik som i Linux, eller ikke en demokratisk utforming slik som i Apache prosjektene, kan det ved uenighet ofte ende opp i konflikter. Mads Toftum har selv vært med på å starte et slikt prosjekt, hvor de startet med å være noen få mennesker som kjente hverandre godt og var vant til å arbeide sammen. I starten var det ikke særlig problematisk at man ikke hadde noen klare retningslinjer for organiseringen, og man baserte seg på common sense. Men etterhvert som prosjektet vokste i omfang, og flere ble deltagere på prosjektet, økte uenigheten. Det ble mer og mer nødvendig med en organisering, fordi prosjektet innebar at man var nødt til å avklare juridiske spørsmål i forhold til kommuner og told & skat. Dette endte opp med en mengde uenighet og konflikter, hvor forskjellige modeller for organisering har vært foreslått, bl.a. Apache modellen. Han betegner selv denne prosessen som en fryktelig kamp, og mener at motstanden skyldes, at noen mennesker rett og slett har svært vanskelig for forandringen; det å plutselig skulle innordne seg etter noen rammer og regler. Utviklingen har gått i retning av utpregede symmetriske relasjoner mellom medlemme. I denne form for prosjekter har man tatt inn nye medlemmer, uten at rammene har vært lagt på forhold til hvordan man eventuelt skulle løse konflikter og avklare maktforhold.

Hvordan forholder man seg til konflikter innad og mot utenforstående.

Konflikter mot utenforstående

Tidligere har jeg beskrevet hvordan Microsofts metoder i kampen om markedet, har vekket vrede i hacker miljøet. Open Source miljøene ser rett og slett Microsoft som hovedfiende nr.1, fordi de føler seg presset ut av markedet og dermed ser sin egen eksistensberettigelse truet. Hvordan skal man så bekjempe fienden? OS miljøene er jo ikke en virksomhet i konkurranse med en annen, men en miljø og en kultur, og kan ikke markedsføre seg selv eller sine produkter på samme måte som Microsoft har vært i stand til. Dette er jo også Microsofts dilemma, at OS miljøet ikke utgjør en vanlig konkurrent, og at de derfor må forsøke å bekjempe en prosess eller en idé. Etter de famøse Halloween dokumentene, ble Linux som operativsystem utsatt for en massiv

kritikk av Microsoft, noe hackere opplevde som skittkasting fra Microsofts side. Diskusjonene og argumentasjonene rundt Linux var og er mange, og temperaturen høy.

Microsoft utarbeidet et notat på slutten av 90 tallet, som de la ut på sine hjemmesider, de såkalte 'Linux Myths' hvor de tok for seg myter eller påstander knyttet til Linux, og sammenlignet med sitt eget operativsystem Windows NT. Det ligger utenfor min kompetanse å vurdere den tekniske kvaliteten på de to operativsystemer satt opp mot hverandre. Allikevel kan man konstantere at selv om Linux har hatt sine begrensninger, så er dette et prosjekt som hele tiden er under utvikling, og miljøet arbeider kontinuerlig for å forbedre og utvikle systemet. Dermed ikke sagt at Microsofts operativsystem er et dårlig system. I kjølvannet av Halloween dokumentene forsøkte Microsoft å dreie diskusjonen vekk fra selve det faktum at dette er en kamp mellom åpen vs. lukket kode, til å bli en diskusjon om hvilken teknologi som var den beste, og dermed den mest lønnsomme for brukerne. I årene som har fulgt har Linux forbedret mange av de svakheter i systemet som Microsoft påpekte. Kanskje er det årsaken til at denne linken med 'Linux Myths' er fjernet fra Microsofts sider, men den er bevart via andre hjemmesider innen OS miljøet.¹

OS miljøet har ikke midler til markedsføring annet enn Internet, når de forsøker å tilbakevise kritikk. Hver gang Microsoft lanserer et nytt produkt, forsøker mange hackere å debugge koden, finner åpninger og feil for deretter å opplyse offentligheten om sikkerhetsrisikoen. Dette setter selvfølgelig Microsofts produkter i et dårlig lys. Denne fremgangsmåten er velkjent lesestoff hvis man følger med på computer relatert journalistikk. Man kan si at forholdet til Microsoft er meget symmetrisk. Språkbruken fra hacker miljøet er tydelig farvet av denne symmetriske relasjonen. Uttrykk som "The dark Force", (fra Star Wars) "Borg" (fra Star Trek) ol. er det mange av, når man betegner den kampen som foregår i cyberspace mellom det man ser som det godes kamp mot det onde. I det hele tatt er det påfallende hvor mange hackere som er opptatt av det som går under betegnelsen *fantasy*. Vår egen tid kjennetegnes av denne barne og ungdomskulturen, som preges av dataspill, rollespill, kortspill, Warhammerfigurer, filmer, ol. Er det noe man forbinder med den typiske nerden, så er det at han elsker Tolkien, science-fiction filmer ol. Språkbruken taler for seg selv i Richard Stallmans

1. <http://www.biznix.org/whylinux/microsoft1.htm>

(protagonistens) fremstilling, når han forteller om GNU prosjektet og om sin egen kamp for frihet. Han viser til skikkelsen *Yoda* fra Star Wars serien:

“TRY!

Yoda's philosophy (There is no “try”) sounds neat, but it doesn't work for me.[...] But I tried anyway, because there was no one but me between the enemy and my city. Surprising myself, I have sometimes succeeded. Sometimes I failed; some of my cities have fallen. Then I found another threatened city, and got ready for another battle. Over time I've learned to look for threats and put myself between them and my city, calling on other hackers to come and join me. [...] But the dangers are greater each year, and now Microsoft has explicitly targeted our community.” (Stallman, *Open Sources: Voices from the Open Source Revolution*, s.22)

I dette sitatet fremstiller Stallman seg selv, som en ensom ridder i kampen for frihet. Språkbruken blir svært kontrastfull når man sammenligner med sitater som er typiske for Microsoft. Sitatene er preget av en overbevisende og litt tørr og teknokratisk språkbruk, i deres forsøk på å gå offentlig ut for å advare kunder og eventuelle brukere i å benytte seg av Linux.

“The more I study Linux, the weaker I think the value proposition is for consumers.”
Ed Muth, Microsoft.¹

“The Linux community likes to talk about Linux as a stable and reliable operating system, yet there is no real world data or metrics and very limited customer evidence to back up these claims. [...] The Linux community will talk about the free or low-cost nature of Linux. It's important to understand that licensing cost is only a small part of the overall decision-making process for customers.”²(Fra Linux Myths dokumentene.)

Halloween Document IV³ har overskriften **When Software Things Were Rotten!** og starter med en ingress fra kommentatoren Raymond:

1. <http://www.linuxworld.com/linuxworld/lw-1999-03/lw-03-thesource.html>

2. <http://old.lwn.net/1999/features/MSResponse.phtml>

3. Det finnes en samling av dokumenter som går under tittelen Halloween Documents, og som alle har tilknytning til det opprinnelige.

"Microsoft executives dismiss open-source as hype. ``Complex future projects [will] require big teams and big capital," said Ed Muth, a Microsoft group marketing manager. ``These are things that Robin Hood and his merry band in Sherwood Forest aren't well attuned to do." (Raymond, 30 Dec 1998) ¹

Microsofts talsmann Ed Muth er faktisk den som skaper en allusjon og knytter Open Source miljøet til Robin Hood. Disse uttalelsene får Eric Raymond til å lage en parodi på et Robin Hood skuespill, med Linus Hood, Munken Eric (som jo må være ham selv), og hvor Ed Muth har fått den lite flatterende rollen som sheriffen av Nottingham, med sin tjener Vinod (ham som skrev Halloween dokumentene), og hvor King Billy som bor i palasset i Seattle blir omtalt. Slik jeg ser det, er denne eiendommelige form for humor, et tydelig uttrykk for en symmetrisk relasjon til motstanderen Microsoft.

I det følgende har jeg forsøkt å beskrive hvordan Open Source Community forholder seg til konflikter utad. Først og fremst er den utadrettede konflikt rettet mot Microsoft. Denne gjensidige konflikt mellom partene, foregår i form av diskusjoner på Internet, men også gjennom forskjellige offisielle medier. I tillegg forsøker hackerne å finne feil i Microsofts produkter så snart de blir lansert, og opplyse offentligheten om dette, noe som setter produktene i et dårlig lys. Denne symmetriske relasjonen til Microsoft, kommer tydelig til uttrykk i språkbruken. Disse eksemplene er ikke alene representative for den debatt som forsvarer OS miljøene. F.eks. er O'Reillys nettverk preget av en mer saklig og nøktern språkbruk. Faktisk har Tim O'Reilly opptil flere ganger forsøkt å få til en dialog med representanter fra Microsoft, og invitert dem med på konferanser, debatter ol. Tim O'Reilly, som også er en av grunnleggerne av OSC, fremstår i rollen som en slags diplomat og fredsmegler i miljøet. Allikevel er det påfallende hvordan språkbruken blandt hackerne, gjenspeiler den kampen mange oppfatter som en kamp mellom det gode og det onde, hvor språket er hentet fra fantasy-litteraturen, og hvor de som deltar i OS kulturen ser seg selv som dem som kjemper i det godes tjeneste. Dette kommer jeg tilbake til.

1. <http://www.opensource.org/halloween/halloween4.php>

Konflikter innad i miljøet

Når Torvalds begynte å skrive Linux kjernen, bestemte han seg for å lage et system med høy grad av portabilitet, og han tok en ganske utradisjonell beslutning dengang, fordi han valgte å designe det som kalles en monolithic-kernel i motsetning til datidens tankegang og diskurs som foretrakk en mikrokernel. En berømt diskusjon om disse forskjellige syn på fremgangsmåte, fant sted på en Minix-newsgroup i 1992, hvor Andrew Tanenbaum og Linus Torvalds braket sammen i en diskusjon, som andre etterhvert involverte seg i. Tanenbaum hadde hørt rykter om Linux prosjektet, (han hadde som tidligere nevnt startet Minix) og fordi han var overbevist om at mikrokernelen absolutt var å foretrekke, hadde han en litt oppgitt og nedlatende holdning til hele Linux prosjektet under overskriften LINUX is obsolete. Han fikk svar på tiltale fra den unge jyplingen Torvalds, som rasende svarte tilbake, og så var diskusjonen igang. (*Open Sources: Voices from the Open Source Revolution, Appendix A, The Tanenbaum-Torvalds Debate*) I ettertid har Torvalds ønsket å gi uttrykk for at han ikke har noe uoppgjort med Tanenbaum, men historien illustrerer at dannelsen av forskjellige prosjekter ikke har foregått uten konflikter. Man kan si at mange av dem som betegner seg selv som hackere, har et lidenskapelig forhold til det de gjør. I motsetningen til diskusjonen rundt Stallman, som er knyttet til opphavsrett og som har idealistiske overtoner, er disse konfliktene innad som regel knyttet til faglige spørsmål. Man er simpelt hen uenige om hva som er riktig å gjøre ut fra hva som teknisk sett er mest hensiktsmessig. Et begrep som går igjen innenfor miljøet er *meritocracy*. I dette ligger at det er den beste teknologien som skal "seire", uten at dette har med personlighet å gjøre.

"Attacking the author rather than the code is not done. There is an interesting subtlety here that reinforces the point; hackers feel very free to flame each other over ideological and personal differences, but it is unheard of for any hacker to publicly attack another's competence at technical work (even private criticism is unusual and tends to be muted in tone). Bug-hunting and criticism are always project-labeled, not person-labeled." (Raymond, s. 90)

Allikevel kan det være vanskelig å skille sak fra person i noen av de debattene som foregår. Som sagt ble Torvalds meget provosert av Tanenbaums uttalelser om Linux. Han har senere uttrykt anger over sin manglende nett-etikette, men man må vel

kunne si at det nettopp er motstanderens tekniske kompetanse som blir angrepet i sitater og uttalelser som denne: (The Tanenbaum- Torvalds Debate)

Torvalds:

“Re 2: your job is being a professor and researcher: That's one hell of a good excuse for some of the brain-damages of minix. I can only hope (and assume) that Amoeba doesn't suck like minix does.” (s.4)

Tanenbaum:

“I still maintain the point that designing a monolithic kernel in 1991 is a fundamental error. Be thankful you are not my student. You would not get a high grade for such a design:-)” (s.6)

Tredjeparter blandet seg stadig inn i debatten, f.eks. med svar på denne kommentaren og dermed et forsvar av Linus:

“That's ok. Einstein got lousy grades in math and physics.”

Konflikter innad i miljøet, dreier seg som sagt, først og fremst om faglige diskusjoner. Noen av disse diskusjonene, vil ikke få innflytelse på selve prosjektene. Linus Torvalds hadde integritet nok, til å overse datidens konsensus, og motsi en berømt professor, om hva som var mest hensiktsmessig når han satte i gang med å lage et operativ system.¹ Hvilke valg man skal ta innad i selve prosjektene, ender opp som et spørsmål om hvordan eller hvem som leder prosjektet - som igjen handler om hvordan man løser konflikter!

Konfliktløsning

Måten man løser konflikter på kan igjen deles inn i tre hovedformer, fordi hvem som eier prosjektet, el. mao. prosjektstyringsformen, er det som avgjør konfliktene.

1. En annen større diskurs og konflikt innad i miljøet, er knyttet til to forskjellige prosjekter som begge lager applikasjoner til desktop. Trolltech, et lite norsk firma som hadde utviklet et graphical interface toolkit kalt Qt, så behovet for et Windows lignende desktop for Linux , og derfor utviklet de K Desktop Environment (KDE), som var basert på Qt's teknologi. Trolltech benyttet seg ikke av en full Open Source lisens, (noe de senere valgte å omgjøre) og dette fikk Red Hat til å ta initiativ til et nytt parallell prosjekt kalt GNOME. Dette har ført til to retninger basert på to konkurrerende teknologier, med dertil hørende diskusjoner og konflikter innad i miljøet.

1. Enevelde formen. (Eksemplifisert ved Linux.) Det sier seg selv at enevelde-formen ikke vil plages av opprivende konflikter innad i prosjektet, selv om man kan anta at den indre kretsen av utviklere vil ha sine interne konflikter. Allikevel er det er Torvalds som "eier" Linux prosjektet. På denne måten vil han bestemme over de offisielle versjonene og lanseringene og på denne måten unngå å splitte prosjektet.

2. Demokrati formen (Eksemplifisert ved Apache.) Har man først blitt en del av en prosjektgruppe innen Apache gruppen, så har man de samme rettigheter som de andre innen dette prosjektet. Hvis noen f.eks. kommer med forslag om å endre funksjonaliteten, så må man stemme over forslaget. Man kan avgi forskjellige typer stemmer ut fra det synet man har på forandringen. Disse er som følger (kilde: Mads Toftum og Apache Guidelines):

+1 Vil si det samme som at man er enig.

+ 0 "Gjør som dere vil!" Vil si at man mener at det er en god idé, men at man ikke kan bakke opp teknisk/man har ikke fått testet koden.

- 0 Vil si at man umiddelbart ikke synes dette er en god idé, men at man ikke er i stand til å forklare hvorfor man ikke synes dette er en god måte. Man liker ikke forslaget, men vil allikevel ikke spenne ben for det.

-1 Vil si det samme som veto. Hvis man bruker denne stemmen, så kan endringen ikke gjennomføres.

Selv om det står i statuttene har jeg fått inntrykk av at praksisen viser at noens stemmer veier tyngre enn andres. Hvis f.eks. et junior medlem som stemmer -1, så veier det kanskje ikke så tungt, mens hvis det derimot er en av de gamle seniorenene som legger ned veto, så blir dette vektlagt. Man skal også forklare og argumentere for sin stemmeavgivning, hvis ikke blir ikke stemmen tatt med. Stemmeavgivning er en måte å avgjøre konflikter på, men som i alle demokratiske prosesser, vil det ikke alltid garantere for at flertallet har rett, eller at den "beste" løsningen vinner. Den som virker mest overbevisende ut fra et faglig synspunkt vil være i stand til å overbevise de andre og dermed vinne "valget".

3. Anarki formen – Denne formen er den som ved konflikter kan ende opp i indre konflikter og symmetriske relasjoner. Blir man ikke enige, vil det tilsi at man bruker

energi på diskusjoner som ikke fører til løsning og som derfor ikke kommer prosjektet til gode. Disse symmetriske relasjonene må enten gå over i en komplementær relasjon, (noen må gi seg) eller de ender i et klimaks som kan føre til totalt oppsplitning av prosjektet.

Formelle teknikker for sosial påvirkning

I enhver kultur har man utarbeidet forskjellige normer for hvordan man påvirker hverandre sosialt. Når man skal beskrive Open Source Community som et kulturellt fenomen, så er det ett moment som skiller denne kulturen fra de fleste andre kulturer. Ansikt til ansikt kommunikasjon hører til sjeldenhetene. Dessuten er det, tiltross for enkelte unntak, en mannsdominert kultur. De som deltar i forskjellige prosjekter, kommuniserer via mailing lister og newsgrupper. Deltagerne kan arbeide sammen over flere år, uten at de noen sinne møtes fysisk. Raymond slår fast, at kulturen er preget av et sterkt meritocracy "one's work is one's statement". Miljøet vil ganske enkelt undertrykke selv promotering, det er kvaliteten på koden som teller.

"What is being directly protected here is the quality of the information in the community's peer-evaluating system. That is, boasting or self importance is suppressed because it behaves like noise tending to corrupt the vital signals from experiments in creative and cooperative behaviour. [...] Potential contributors want project leaders with enough humility and class to be able to say, when objectively appropriate, "Yes, that does work better than my version, I'll use it" – and give credit where credit is due." (Raymond, s.91)

I og med at dette er et nettsamfunn, hvor ansikt tilansikt kommunikasjon hører til sjeldenhetene, og hvor kommunikasjonen foregår ved maillister og newsgrupper, så sier det seg selv at det finnes få teknikker for sosial påvirkning, bortsett fra gjennom selve det arbeid man legger for dagen.

Hvem er ledere?

Når jeg har innsamlet materiale om Open Source miljøet, har jeg dannet meg et inntrykk av hvilke lederskikkelser som trer frem, og som jeg tidligere har omtalt. Noen ledere har endt opp i rollen fordi de har startet et prosjekt som har blitt populært. Noen har endt opp som leder fordi de er fremtedende i organiseringen og i offentlig-

heten rundt OSC. Raymond og O'Reilly er representanter for denne type ledere. Er det noe man *ikke* ser eksempler på i denne kulturen, så er det hierarkiske strukturer i organiseringen av prosjektene. Høyst to eller tre ledd er normalen. I følge Raymond er det en utbredt negativ oppfatning innad i miljøet, hvis man kan merke selv-promotering og ego – dyrking. Dette får implikasjoner i synet på lederskikkelser: “So much so, in fact, that the culture's 'big men' and tribal elders are required to talk softly and humorously deprecate themselves at every turn in order to maintain their status.” (Raymond, s.89) Raymond mener at dette skyldes en europeisk-amerikansk tradisjon, som har en negativ holdning til 'ego' eller ego-dyrking, og at det innebærer en motsetning i kulturen, fordi folk ikke blir drevet av idealisme alene, men har andre motiver for å delta i miljøet. Dette kommer jeg tilbake til.

Hva er tabu?

Som jeg er inne på i avsnittene ovenfor, er ett av tabuene innen OS miljøene, dette med ego-dyrking og selv promotering. Det å angripe andres tekniske kompetanse er et tabu i følge Raymond, selv om man (som tidligere vist) kan se eksempler på slike indirekte angrep. I en debatt på slashdot.com,¹ som er ett portal for utviklere og OSC, som fulgte i kjølvannet av artikkelen *The Linux Edge*, skrevet av Torvalds selv, ble mange fortørnet over noen av hans uttalelser. Fordi han tross alt er den han er, og en slags lederskikkelse, legger folk merke til hans uttalelser. Han snakker nedsettende om Emacs, et meget velansett og brukt program som Richard Stallman har stått bak:

“A number of them I hate with a passion; the Emacs editor is horrible, for example. While Linux is larger than Emacs, at least Linux has the excuse that it needs to be.”

Denne uttalelsen fikk flere til å reagere. Under overskriften *Linus, the great diplomat* kommenterer Bryan Ischo denne uttalelsen:

“Oh my god what bullshit. If there is one thing I learned from that article, it's that Linus is not nearly as humble as I thought he was. He seems to want to make a point of pissing on everyone else (“microkernels suck and the people who made them are stupid, everything from GNU except GCC is lame, etc”) for some reason. What an ego.” (ibid.)

1. <http://slashdot.org/article.pl?sid=99/03/24/2219207&mode=thread&tid=106>

Ryan Ischo var neste kommentator:

“I agree wholeheartedly. I used to think Linus was humble and I respected that. It's a great thing to do something great and still be humble about it. It's quite the opposite to do a great thing (albeit with the help of thousands of others) and in the end use it as a platform to insult others and minimize their accomplishments. I have to say, RMS [Richard Stallman, *min tilføyning*] may have some radical ideas, and he may support them pretty strongly, but I can't remember the last time I heard of him calling someone else stupid or minimizing their accomplishments.” (ibid.)

I disse kommentarene blir Torvalds kritisert for ikke å være ydmyk nok, og tolket dit hen at han indirekte angriper Richard Stallmans kompetanse og innsats. *Ydmykhet* er et nøkkelbegrep innen miljøet, og det som egentlig ligger til grunn for dette ønsket om ydmykhet, er i følge Raymond miljøets ønske om at alle parter skal evaluere hverandre som likeverdige partnere. Hvis ydmykheten overfor andres prestasjoner skyves i bakgrunnen, til fordel for selv-promotering, så trues jo også mulighetene for å bibeholde en kultur som er merittbasert (meritocracy). Men man kan også tolke Torvalds sitat, som en klar utmelding og kommentar til selve koden. Han liker ganske enkelt ikke Emacs, uten at han dermed sier noe nedsettende om Stallman som person. De som kritiserer Linus uttalelser bryter faktisk selv kodeksen, for det at de mislikte hans uttalelser gir dem ikke berettigelse til å uttale seg nedsettende om Linus Torvalds. Men som tidligere nevnt, blir folk skuffet over ham, fordi han er en lederskikkelse. Mads Toftum mente at selv om det forekommer, så er det et klart tabu å begynne å ramme hverandre på det personlig plan. Som Toftum uttalte: “Diskusjonene dreier seg om koden, det er det du laver, ikke det at du er et dumt svin. Hvis det var dét, så var der aldri noen som ville bruke Emacs!” Denne paradoksale uttalelsen kunne ha virket krenkende mot person, hvis den hadde stått i en newsgruppe, hvilket den jo ikke gjorde. Allikevel er det vanskelig å ikke oppfatte Torvalds uttalelse om Emacs som krenkende. Det er mao. problematisk å skille kritikk av kompetanse fra kritikk av person, selv om Raymond påstår det motsatte. Slike uttalelser skaper støy i en kultur som kommuniserer via newsgrupper og mail; en form for støy som setter i gang symmetriske relasjoner innad i miljøet.

Den andre form for tabu som er fremtredende i miljøet er knyttet til det jeg har vært inne på tidligere om eiendomsrett. Det finnes en gylden regel om at den som eier pro-

sjektet er den som har tillatelse til å distribuere modifiserte versjoner. I følge Toftum vil det å lage en endring på egenhånd uten å spørre de andre, være dss. å bryte et av de strengeste tabuer innen kulturen. Likeledes ville det å ta kreditt for andres arbeid være å bryte et tabu. Enhver skal ha æren for sitt eget bidrag.

Underliggende verdier (ethos)

Hva slags forhold har man til økonomi?

Tidligere viste jeg til Raymonds definisjon av eiendomsrett i OS miljøene: *eieren av et software prosjekt er den person som har eksklusiv rett til å distribuere modifiserte versjoner*. De uskrevne regler om eiendomsrett i OS miljøene har utviklet seg over tid, etter samme mønster som andre eldre kulturer i Europa. De nordisk og germanske folkegrupper var ættesamfunn, eller stammesamfunn og utviklet lover over en tusenårsperiode. Felles for f.eks. vikingesamfunnene og det ville vesten i Amerika, var fraværet av en sentralmakt/institusjoner. Man måtte selv sette loven ut i live eller forsvare sin ære hvis den var krenket; *a man is a man, and a man has to do what a man has to*. John Locke regnes for den som systematiserte, rasjonaliserte og moderniserte lovverket, og derfor refererer man ofte til the Lockean theory of property.

Den historiske utvikling av eiendomsrett, kan sammenlignes med de anglo amerikanske lover og teorier som utviklet seg i forbindelse med landbesittelse. I disse teorier, finnes det tre forskjellige måter å kreve eierskap på (i likhet med eiendomsrett på prosjekter i OSC).

1) I landområder som ikke tidligere har hatt eiere (the frontier i amerikansk sammenheng eller f.eks. ubebodde områder på Island), kunne man arbeide og stelle med jorden, sette opp gjerder rundt den, og forsvare sin tittel. Dette betegner forfatteren som 'homesteading'. Denne fremgangsmåten kan sammenlignes med at den/de som starter et prosjekt innen OSC også eier det.

2) Den vanligste måten å skaffe seg eiendomsrett på i bosatte områder, er tittel-overføring. "The ideal proof of ownership is a chain of deeds and transfers extending back to when the land was originally homesteaded." (Raymond, s.77) Dette er samme fremgangsmåte som man benytter seg av i OSC, ved å overføre eierskapet av et prosjekt til en annen.

3) Sist viser common-law teoriene til at land-titler kan falle bort eller oppgis: “If the owner dies without heirs, or the records needed to establish chain of title to vacant land are gone:” (ibid.) På samme måte som i OS prosjekter kan andre kreve retten ved *adverse possession* – andre flytter inn, forbedrer og forsvarer tittelen som i homesteading.

Disse Lockeanske mønstrene for eiendomsrett, viser seg når det forventede utbytte av ressursene, overstiger omkostningene ved å forsvare dem. “The Lockean logic of custom suggests strongly that open-source hackers observe the customs they do in order to defend some kind of expected return from their effort” (ibid.)

Hva slags utbytte får man så ved å forsvare sitt prosjekt (territorie) innen OS kulturen? Maktbegjær er utelukket, for hvilke tvangsmidler skulle man eventuelt benytte seg av i en nettkultur? Softwaren er gratis, så rent økonomisk gevinst er også uaktuelt. Allikevel vil det implisitt ligge en mulighet for økonomisk belønning forbundet med OS aktiviteter og eierskap av prosjekter. Dette vil jeg behandle i neste avsnitt.

Hva slags former for belønning finnes i kulturen?

Det finnes en form for økonomisk belønning som kan knyttes til aktiviteter innen OS kulturen; hva Raymond kaller *reputation-game*. “Occasionally, the reputation one gains in the hacker culture can spill over into the real world in economically significant ways. It can get you a better job offer, or a consulting contract, or a book deal.” (Raymond, s. 79) Ett av tabuene innen miljøet er motstanden mot ego-satisfaction, og det er en utbredt oppfattelse av at en av de viktigste drivkrefter for å delta og bidra, er ut fra et ønske om idealisme og altruisme. Raymond ser en motsetning her: samtidig som man undertrykker ego promotering, så er det å skaffe seg et godt omdømme drevet av en tilfredsstillelse av ego, fordi det i neste omgang kan gi mulighet for økonomisk gevinst. På mange måter kan man si at Raymond påpeker en forskjell i hva folk sier og tror de gjør, og det de faktisk gjør.

Raymond foretar en *reputation-game* analyse, som kan knyttes til de underliggende verdier i OS kulturen. Han sammenligner forskjellene mellom en bytte-kultur og en gave-kultur. Evolusjonshistorien har vist at menneskeheten har en felles drivkraft; vi konkurrerer om sosial status. Opp gjennom historien har menneskene organisert og tilpasset seg i forskjellige former for å tilfredstille livsnødvendige behov, og for å

oppnå det man mangler og ønsker. Alle former for organisering bærer i seg et ønske om å oppnå sosial status. Den enkleste form for menneskelig organisering er *kommando hierarki*. Her vil en sentral autoritet allokere godene, ved hjelp av makt. Disse strukturene kan bli svært brutale og ineffektive hvis de vokser i størrelse og omfang. Sosial status oppnås ved tilgang til utøvende makt. Vårt eget industrialiserte samfunn er preget av *bytte-økonomi*. I motsetning til kommando hierarkiet, fordeler man godene ved hjelp av desentralisering. Handel og frivillig samarbeid er forutsetningen for denne organiseringen. "In an exchange economy, social status is primarily determined by having control of things (not necessarily material things) to use or trade." (Raymond, s.81) Det finnes imidlertid en tredje modell som adskiller seg fra de andre, (selv om denne først og fremst blir gjenkjent av antropologer) – *the gift culture*.

Gave kulturer, er kulturer som tilpasser seg overflod i motsetning til mangler. Man kan finne gave kulturer i opprinnelige kulturer som lever under slike klimatiske forhold at man får dekket sine livsnødvendige behov, men man kan også observere dem i noen av våre egen samfunnslag, f.eks. blandt de meget velhavende eller innen show business. Raymond mener at Open Source kulturen utgjør en slik gave-kultur. I gave kulturer tilegner man seg status, ikke gjennom hva man kontrollerer, men gjennom hva man gir vekk. Deltagerne konkurrerer om prestisje ved å gi fra seg tid, energi og kreativitet. Softwaren blir gledelig delt med andre. "This abundance creates a situation in which the only available measure of competitive success is reputation among one's peers." (ibid.)¹ Hvordan forholder man seg til utestengning/felleskap?

Open Source Community er fellesbetegnelsen for et kulturelt fenomen som har oppstått som følge av Internet teknologien. Individuer kan delta i grupper og være del av en global kultur. Indere, amerikanere, finner og polakker kan finne sammen og delta i et felleskap. Som i alle andre kulturer risikerer man utestengning fra felleskapet hvis man bryter spillereglene og tabuene. Tidligere beskrev jeg hvilke tabuer som eksisterte innenfor denne kulturen, henholdsvis den negative holdning til selv-promotering, samt det å foreta en endring på egenhånd uten å spørre de andre. Ved å bryte

1. Her skiller hacker kulturen seg fra cracker kulturen. Cracker kulturen utfolder seg i et elektronisk medium på samme måte som hacker kulturen, men adferden er forskjellig. Gruppe mentaliteten er sterk, og sosial status oppnås ved å bryte seg inn i systemer (innbrudd) eller å ødelegge systemer (hærverk). Crackerne vokter sine hemmeligheter, og i motsetning til hacker kulturen er det sjelden at man gir fra seg/deler fremgangsmåten.

tabuene kan man risikere utestengning. Man kan bli kastet ut av et prosjekt, eller bli ignorert. Kommunikasjonen på prosjektene foregår ved hjelp av mailinglister og newsgroups. Mads Toftum fortalte at man mer og mer går over til mailinglister, fordi dette lettere gir mulighet for sanksjoner overfor dem som skaper støy, (symmetriske relasjoner). Hvis man f.eks. blir en meget ivrig bruker av veto retten, så kan man kastes ut av prosjektet. I følge Toftum finnes det personer som skaper støy og ødelegger et konstruktiv samarbeid ved en oppførsel som ikke følger net-etikette eller guidelines. Da risikerer man å strykes av mailinglisten. Tidligere (Del 1.) skrev jeg om hvor vanskelig det er for en programmør å kommunisere til sin omverden hva man egentlig holder på med. Jeg påpekte at en av drivkrefter for å delta i dette miljøet var behovet for felleskap. Alle har vi behov for bekreftelser og evaluering av det vi presterer, og det å være del av et felleskap er viktig for de aller fleste mennesker. På samme måte som i andre kulturer, finnes det grenser for sosiale avvik som kommer til syne i den måten man kommuniserer med andre på. Hvis ens adferd avviker for meget i forhold til de normer som finnes, så oppnår man et dårlig rykte, og man risikerer å bli utestengt fra felleskapet. I og med at dette er en Internet kultur, hvor man ikke møtes fysisk, vil det for noen være av mindre betydning å bli utestengt. Men jeg antar at dette forholder seg annerledes for en person som har et utstrakt behov for felleskap og utfoldelsestrang. Gleden av å skape noe i felleskap med andre, er samtidig en mulighet for å oppnå sosial status og aksept, og jeg antar at de som har denne motivasjonsfaktoren vil unngå utestengning for enhver pris.

Verdisystemer innen Open Source kulturen - oppsummering

I de foregående avsnitt har jeg forsøkt å beskrive Open Source miljøene – dels i form av adferd og relasjoner, men også ved å forsøke å beskrive dens ethos. Beskrivelse av adferd kan avdekke en kulturs underliggende strukturer; hva som får denne kultur til å bevege seg i den retning den engang gjør. Det er dette man kan karakterisere som en kulturs verdisystemer.

Jeg har vist at det finnes tre hovedformer for prosjektstyring/eiendomsrett, som jeg har eksemplifisert og navngitt etter forskjellige samfunnsformer: enevelde, demokrati, anarki. De to første er preget av komplementære relasjoner mellom medlemmene på prosjektene, den sistnevnte er den form som mest avler symmetrisk adferd. De for-

skjellige prosjektformer viser til hvem som har eiendomsrett på prosjektet. Konflikter innad på prosjekter, løses i henhold til hvilken type prosjekt de hører inn under. Konflikter som oppstår, enten i miljøene eller på prosjektene, dreier seg først og fremst om faglige uenigheter. Faglige diskusjoner kan bære preg av en lidenskap i forhold til det man holder på med, og temperaturen kan være høy. Videre har jeg vist at denne kulturen unngår hierarkiske strukturer. Det finnes et utbredte ønske om å unngå symmetriske relasjoner i form av folk på prosjektene som er lite samarbeidsvillige, og hvor alt ender opp i konflikter. Man ønsker å unngå det man kaller støy, fordi dette fremmer denne form for cumulatitv adferd. Det er en utpreget nettkultur, og det å promotere seg selv – såkalt egodyrking – er ett av tabuene. Likeledes er det å foreta seg endringer på egenhånd i prosjektet å bryte ett av de sterkeste tabu. Til tross for at GPL lisensen nærmest oppfordrer enhver til å gjøre hva man vil, så eksisterer det “offisielle” versjoner av softwaren, og på den måten unngår man oppsplitning av prosjekter. Kulturen bruker betegnelsen meritocracy om seg selv. I dette ligger at det er ens arbeid som står i fokus, og ikke hvem du er som person.

Hvis man skal ha lyst til å delta og legge arbeid inn i et prosjekt, så ønsker man at de andre skal kunne gjenkjenne og respektere det man gjør ut fra hva som faglig og teknisk sett er den beste løsningen. Dette sikrer at alle deltagerne står på likefot, og at ens arbeid blir vurdert av de andre uten personlig påvirkning. Man skal mao. være ydmyk nok til å ta i bruke de beste løsninger uansett hvem den kommer fra. Dette tolker jeg som ett av de viktigste momentene for å delta. Hvis man legger arbeid og energi inn i et prosjekt, så har man muligheten for å kunne skrive noe og skape noe. Isteden for at dette produkt ligger i en skrivebordskuff, så kan man dele det man har skapt med tusenvis av andre. Det i seg selv lokker, men da må man også være overbevist om at ens innsats blir vurdert på en fair måte, at diskusjonen kun dreier seg om forskjellige syn på teknologiske løsninger, og ikke om en selv er mindre berømt enn en annen.

Det er fasinende å tenke på at man i denne kulturen faktisk produserer avansert teknologi så det koster, og at det har oppstått forskjellige former for organisering av enkeltindividers bidrag til disse prosjektene. Det negative synet på selv-promotering innebærer at de som er naturlige lederskikkelser innen miljøet, må tone seg selv ned for ikke å virke for prangende; ydmykhet, er et ord som ofte går igjen. Dette forteller noe om de underliggende verdier, ethos, fordi det finnes en motsetning her. Ray-

monds analyser, beskriver kulturen som en gavekultur, hvor man oppnår status gjennom å gi fra seg tid, kreativitet og energi. Man oppnår sosial status gjennom det han kaller et reputation game, og dette vil også si at man tilfredstiller sitt ego, uten at han dermed ser noe galt i det. Det Raymond får frem, er motsetningen mellom hva folk sier og tror de gjør, og det de faktisk gjør.

Hvis man skal peke på de underliggende strukturer som styrer adferden innenfor miljøene, så er det noe som er felles for denne kulturen, noe som skiller dem fra oss; kampen mot Microsoft. I årevis har hacker miljøene vært vitne til hvordan Microsoft har sikret seg monopol ved å stenge andre ute. Nettopp åpenhet og det å dele med andre for å sikre plattformuavhengighet er forutsetningen for hele miljøet. Når Eric Raymond holder foredrag, stiller han alltid deltagerne det samme spørsmål: hvor mange av hackerne er det som sitter i rene softwarehus og produserer ny software? Som regel viser det seg at det kun er få av deltagerne som arbeider på denne måten. Langt de fleste arbeider i IT avdelinger, som systemansvarlige, med drift, vedlikehold, og f.eks. hvor implementering av nye applikasjoner på eldre software hører til dagsorden. Når man i dagligdagen skal få forskjellige typer software til å arbeide sammen, står man overfor oppgaver som blir vanskeligere å utføre pga. Microsofts lukkethet. Når det i tillegg har vist seg at konsernet regelrett ønsker å motarbeide hele miljøet, kan man godt forstå frustrasjonen. Som jeg tidligere har vært inne på, er miljøet – utad - preget av en symmetrisk relasjon til Microsoft. I min redegjørelse for schismogenese og ethos begrepet, viste jeg til hvordan en eskalering av den ene eller den andre form for schismogenese kunne være skadelig for partene, men at elementer fra den ene relasjon kunne dempe den andre og dermed oppnå likevekt. Jeg tolker det dithen, at miljøet eller kulturen har begge disse former for schismogenese, men at man fra et helhetsperspektiv kan påstå at forholdet innad er komplementært og utad symmetrisk, og at kulturen på denne måten oppnår en slags likevekt.

Hvis man skal peke på forskjellige verdisystemer innen miljøet, har dette sammenheng med hvilken ideologi som er den sterkeste drivkraft for den enkelte. Noen er drevet av sterk altruisme – man har et utbredt ønske om frihet og et ønske om å være et alternativ til kommersielle interesser. Man kan kalle de delene av miljøet som først og fremst er drevet av idealisme for Stallmans fraksjon. Deres innstilling til Microsoft blir av Toftum betegnet som “relgionskrigen”. Andre er mer drevet av gleden av å

skape og over å få tilgang på flest mulig verktøyer. Det at koden er åpen og gratis og dermed gir mulighet for at andre kan utnytte softwaren kommersielt, har mindre betydning. La oss navngi dette verdisystem som *pragmatisk samarbeid i cyberspace*.

En av de fenomener jeg har undret meg over i lang tid, er mitt inntrykk av at den typiske hackeren eller nerden er preget av det jeg samlet kaller fantasy. Man skulle nesten tro at hackerkulturen har smittet over på resten av samfunnet. “Krigen om ringen”, “Harry Potter”, “Star Wars” trekker fulle hus på kinoene. On-line dataspill med middelalderscenarioer florerer blandt de unge, særlig unge gutter. Man ser fenomener hvor ungdom og unge gutter faktisk leker sammen, og etterligner disse middelalder scenarioer i det man kaller live-rollespill. Hva slags lengsel etter middelalderen er det som skaper denne type fenomener? I følge Raymond har utviklingen av eiendomsretten fulgt samme mønster i anglo-amerikansk utvikling som eiendomsrett på prosjektene innen OSC. Det som kjennetegner denne første fase i utvikling, er samfunn som ikke er underlagt noen statsmakt eller andre sterke institusjoner. Kulturene må selv utvikle og håndheve lovene. Dette medfører at man utvikler en kultur hvor æresbegrepene blir sterke, og hvor man respekterer hverandres territorier for å unngå konflikter. På samme måte som i dataspill eller i middelalderkulturen, så oppnår man status ved sine meritter – handlingen er sentral. Det som kjennetegner fantasy-universet er *det godes kamp mot det onde*. For å kunne illustrere de underliggende strukturer for disse miljøene, vil jeg forsøke å trekke inn en modell. Innen for litterær analyse, og strukturalistisk teori, (Vladimir Propp og A.J. Greimas) har det blitt utviklet narratologiske modeller som skjematiserer komposisjonen i undereventyr. Både oppbygning av handlingsforløp og sammensetning av persongalleri blir skjematisert. Greimas regner med 6 roller, eller aktanter, og samspillet mellom disse setter han inn i en modell, aktant modellen. Hovedpersonen, *subjektet*, (helten) retter sine ønsker og sin vilje mot et mål: *objektet*. Aksen mellom objekt og subjekt kalles prosjektaksen. I eventyr er det slik at en *avsender* må gi fra seg eller overgi objektet til *mottaker* som som regel er identisk med subjektet (f.eks. kongen gir fra seg prinsessen til helten). Dette kalles kommunikasjonsaksen, og er ikke særlig hensiktsmessig for denne illustrasjonen. Den tredje aksen kalles konfliktaksen og den viser til at subjektet får støtte av hjelperne og blir motarbeidet av motstanderne i sin streben etter å nå målet. Det som kjennetegner denne målrettede aktivitet kalles innen strukturalistisk litteratur for et prosjekt. (Svensen, s.142) Slik jeg ser det kan aktantmodellen forklare noe om underliggende

verdier i OS miljøene. Hvis man ser den enkelte deltager som et subjekt som ønsker å oppnå et objekt; mao. via prosjektaksen, kan man sammeligne dette med den måten programmører arbeider: man ønsker å finne løsningen på problemet. Hackernes selvopfattelse innen Open Source miljøene, er preget av den symmetriske relasjonen til Microsoft. Hvis man sammenligner denne relasjonen med konfliktaksen, vil man i kampen for å nå sitt mål ha noen hjelpere (andre hackere) og noen motstandere. Dette avspeiles med stor tydelighet i språkbruken, hvor hackerne kjemper det godes kamp mot det onde. I sitatet fra R. Stallman, beskriver han seg selv som en ridder som kjemper det godes sak. Han tar på seg rollen som protagonist, helteskikkelsen fra tragedien, som kjennetegnes ved heroisk mot og utholdenhet. På samme måte som i undereventyr, er det handlingen som står i fokus og man oppnår sosial status gjennom sine meritter. På denne måten får medlemmene av denne kulturen en mulighet for å delta i et kulturelt felleskap, hvor målet er å finne løsninger. Ved å være en del av denne kulturen, kjemper man også det godes kamp mot det onde, bekjemper motstanderen og oppnår status gjennom sine meritter (skrive god kode) – *uten* at man kjemper om makt og inngår i hierarkiske maktkonstellasjoner for maktens egens skyld. Det å få sin gode kode anerkjent og brukt, å finne løsningen, blir nesten det samme som å befri prinsessen fra dragen (Microsoft). Samtidig har man overvunnet vanskeligheter og seiret i en kamp mot det onde. Denne *merittbaserte ridderkultur* beskriver den ethos, mao. de underliggende verdier, som er karakteristisk for Open Source miljøene.

Litteratur & Kilder

Litteratur

Batesons, Gregory: *Steps to an ecology of mind*, University of Chicago Press, 2000

Christensen, Jens: *Teknologihistorie. Tiden efter 1950. Kompendium III, 1*, Informationsvidenskab, Universitetet i Aarhus, 2000

DiBona, Ockman, Stone; red.: *Open Sources: Voices from the Open Source Revolution*, O'Reilly, 1999

Eggen, Einar: "Metafor og metonymy", *Norskraft: arbeidsskrift for nordisk språk & litteratur*, 1976

Fjelland, Ragnar: "Uartikulert kunnskap og dens betydning for informasjonsteknologien", i *Kunnskapsteknologi og Filosofi*, Filosofisk institutts publikasjonsserie nr.9, Filosofisk institutt, AVH, Universitetet i Trondheim, 1988

Hølmebakk, Hilde: *Hvordan foregår vitensdeling hos Mjølner? Et forsøk på å oppdage mønstre...*, Informationsvidenskab, 2001

Jakobson, Roman: "Lingvistik og poetik", *Vindrosen*, 1987

Jakobson, Roman: "To aspekter af sproget og to typer afatisk forstyrrelse", *K&K* 78, 1995

Jensen, Jens F. m.fl.: *Reklame – Kultur*, Aalborg Universitetsforlag, 1993

Johnson Deborah G. & Nissenbaum, Helen; red.: *Computers Ethics & Social Values*, Prentice-Hall, 1995

Lekanger, Kurt : *Edb-ord 2000*, IDG Danmark A/S, 2000

Lytje, Inger: *Software som tekst. En teori om systemutvikling*, Aalborg Universitetsforlag, 2000

Marsh, Gorayska & Mey: *Human Interfaces. Questions of Method and Practice in Cognitive Technology*, Kapittel 7. Artikkelen: Kaptelinin & Kuutti: "Cognitive Tools Reconsidered", 1999

Mc Millan, Robert: "Samba Lessons. Andrew Tridgell's Ten-Year-Old Side Project Is One of Linux's Most Important Apps", *Linux World*, Volume 3. Number 7, 2001

Naur, Peter: *Intuition in Software Development*, Datalogisk Institutt, Universitetet i København, (1985)

Naur, Peter: "Programming as Theory Building", *Microprocessing and Microprogramming nr. 15*, North Holland Publishing Company, 1985

Peek, O'Reilly, Loukides: *Unix Power Tools*, O'Reilly 6 Associates/Bantam, 1993

Raymond, Eric S.: *The Cathedral & The Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly, 2001

Svensen, Åsfrid: *Tekstens mønstre. Innføring i litterær analyse*, Universitetsforlaget A/S, 1985

Vygotsky, L.S.: *Mind in society, The Development of Higher Psychological Processes*, Harvard University Press, 1978

Ølgaard, Bent: *Kommunikation og økomentale systemer ifølge Gregory Bateson*, Akademisk Forlag A/S, 2. utgave, 2001

Lenker

Apache site: <http://apache.org/>

Artikler fra OS konferansen august 2002: <http://www.oreillynet.com/pub/a/network/2002/08/05/sterling.html>

Commercial Internet Exchange: <http://www.cix.org>

Digi.no – nettavis: <http://www.digi.no/dtno.nsf>

Eric S. Raymonds hjemmeside: <http://www.tuxedo.org/~esr/>

Eric S. Raymond Jargon Files Resources (The New Hacker's Dictionary): <http://www.tuxedo.org/~esr/jargon/>

Freshmeat: <http://freshmeat.net/>

GNU (Free Software Foundation): <http://www.gnu.org/>

Halloween dokumentene: <http://www.opensource.org/halloween/>

Linux Myths (Microsofts notat): <http://www.biznix.org/whylinux/microsoft1.htm>

Linux International: <http://www.li.org/>

Linux International: <http://www.li.org/linuxhistory.php>

Linux site: <http://linux.com/>

Linuxworld: <http://www.linuxworld.com/>

Netcraft: <http://www.netcraft.com/>

Newsforge: <http://newsforge.com/>

Om code poets: <http://www.thinkgeek.com/tshirts/coder/27ac/>

Open Source i danmark: <http://www.opensource.dk/>

Open Source Initiative: <http://www.opensource.org/>

O'Reilly portal: <http://www.oreilly.com/>

O'Reilly net: <http://www.oreillynet.com/>

OSDN – Open Source Development Network: <http://www.osdn.com/>

Patent og varemærkestyrelsens rapport om Open Source i Danmark: http://www.dkpto.dk/publikationer/rapporter/open_source/open_source_i_danmark.pdf

Richard Stallmans hjemmeside: <http://www.stallman.org/>

Slashdot: <http://slashdot.org/>

SSLUG : http://www.sslug.dk/patent/DKPTO_OSS_rapport_kritik/index2.shtml

Streaming fra Peter Salus foredrag ved Alexandra Instituttet 2002 "Ten years of Linux" hvor jeg selv var tilstede, samt andre opptak fra forelesninger med Peter Salus og Eric Raymond, hvorav noen dessverre er av dårlig kvalitet.: <http://www.chrillesen.dk/stream/>

Teknologirådets presentasjon rundt undersøkelsen: <http://www.tekno.dk/sub-page.php3?article=579&language=dk&category=6&toppic=kategori6>

The Internet society: <http://www.isoc.org>

Torvund, Olav: "Opphavsrett – en Introduksjon" – Institutt for rettsinformatikk, Universitetet i Oslo, 1997: <http://www.torvund.net/artikler/art-opphav.asp>

Torvund, Olav: "Opphavsrett i vertikalt samarbeid": <http://www.torvund.net/artikler/Vertikalt%20samarbeid.asp>

W3org: <http://www.w3.org>