
Den magiske kedel

af Eric S. Raymond

Denne oversættelse af Eric Raymonds The Magic Cauldron er lavet af Jesper Laisen og Peter Brixen. Du kan finde originalen på <http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/>. Tak til Anders Drejer Nygaard for rettelser. Den mangler stadig finpudsning, så send endelig rettelser.

Denne tekst analyserer open source-fænomenets økonomiske underlag. Først afslører vi nogle typiske myter om finansieringen af programudvikling og prisstrukturen af software. Vi fremlægger en strategi-analyse af open source-samarbejdets stabilitet. Vi fremlægger ni modeller for bæredygtig finansiering af open source-udvikling; to almennyttige og syv profitorienterede. Vi fortsætter med en kvalitativ teori om, hvornår det er økonomisk rationelt at være lukket. Vi undersøger derefter nogle nye mekanismer til finansiering af profitorienteret udvikling af open source, som dukker op i markedet, inklusive genopfindelsen af støttesystemet og markedet for opgaver. Vi afslutter med nogle forsigtige forudsigelser af fremtiden.

1. Ikke til at skelne fra magi

I den wallisiske mytologi havde gudinden Ceridwen en stor kedel, som på magisk vi kunne lave nærende mad — når gudinden fremsagde en trylleformular, som kun hun kendte. Indenfor den moderne videnskab gav Buckminster Fuller os begrebet ”kvik teknologi”, teknologi, der både bliver mere effektiv og billigere efterhånden som de fysiske ressourcer, investeret i tidlige designs erstattes af mere og mere informationsindhold. Arthur C. Clarke forbandt de to ved at sige, at ”Enhver tilstrækkelig avanceret teknologi er ikke til at skelne fra magi”.

For mange mennesker virker open source-samfundets succeser som en urimelig form for magi. Software af høj kvalitet dukker ”gratis” frem, hvilket er rart, så længe det varer, men det virker ikke bæredygtigt i den virkelige verden med konkurrence og knappe ressourcer. Hvor ligger fælden? Er Ceridwens kedel bare en tryllekunst? Og hvis ikke, hvordan virker den kvikke teknologi så denne sammenhæng — hvilken trylleformular er det, som gudinden kender?

2. Mere end nørder med gaver

Open source-kulturens erfaringer har bestemt gjort mange antagelser til skamme, som folk uden for kulteren havde om udvikling af software. ”Katedralen og basaren” [CatB] beskrev de måder, som decentraliseret samarbejdende softwareudvikling anvender til at overvinde Brooks lov, og som fører til et på lideligheds- og kvalitetsniveau uden fortilfælde i individuelle projekter. ”Nybyggeri i noosfæren” [HtN] undersøgte den sociale dynamik i denne ’basar’-stil, og argumenterede for, at den forstås bedst, ikke som en konventionel bytteøkonomi, men som det, antropologer kalder en ’gavekultur’, hvor medlemmer konkurrerer om status ved at give ting væk. I denne tekst vil vi begynde med at afsløre nogle udbredte myter om økonomien bag softwareudvikling; dernæst bringe analysen fra [CatB] og [HtN] ind i økonomiens, strategianalysens og forretningsmodellernes område og udvikle de nye begrebmæssige værktøjer, der er nødvendige for at forstå den måde, som open source-udviklernes gavekultur kan opretholde sig selv i en bytteøkonomi.

For at udføre denne analyse uden forstyrrelser, er vi tvunget til at forlade (eller i det mindste midlertidig ignorere) forklaringen om ’gavekulturen’. [HtN] hævdede, at gavekultur-adfærden opstår i situationer, hvor varer til overlevelse findes i tilstrækkelig overflod til at gøre bytteøkonomien

mindre interessant; dette virker som en god psykologisk forklaring af adfærden, men den er ikke helt tilstrækkelig som forklaring for den blandede økonomiske kontekst, som de fleste open source-udviklere faktisk arbejder i. For de fleste har bytteøkonomien mistet sin tiltrækningskraft, men ikke sin kraft til at binde. Deres adfærd må være fornuftigt set ud fra en økonomisk situation med varemangel, for at fastholde dem i et område, der understøtter en overskudsgavekultur.

Derfor vil vi nu se på samarbejds- og byttemodellerne, som opretholder open source-udvikling (udelukkende set ud fra en økonomisk situation med varemangel). Undervejs vil vi detaljere og med eksempler besvare det pragmatiske spørgsmål, "Hvordan kan jeg tjene penge på det?" Til at begynde med vil vi dog vise, at en stor del af grunden til dette spørgsmål ligger i modeller for software-produktion som er lige lukt forkerte.

(Én sidste bemærkning før redegørelsen: Diskussionen om og støtten til open source-udvikling i denne tekst bør ikke tolkes som om, closed source-udvikling i grunden er forkert, heller ikke som et skrift mod intellektuel ejendomsret inden for software eller som en uegennyttig appel til at "dele". Selv om disse argumenter værdsættes af en højrystet minoritet indenfor open source-udviklingsmiljøet, har erfaringen siden [CatB] gjort det klart, at de er unødvendige. Open source-udviklingens styrker vises tilstrækkeligt gennem de ingeniørmæssige og økonomiske resultater — bedre kvalitet, større pålidelighed, færre omkostninger og flere valg.)

3. Vildfarelsen om en fremstillingsindustri

Vi må begynde med at bemærke, at computerprogrammer ligesom alle andre former for kapitalgoder har to særskilte typer af økonomisk værdi. De har en *brugsværdi* og en *salgsværdi*.

Et programs *brugsværdi* er dets økonomiske værdi som et værktøj. Et programs *salgsværdi* er dets værdi som en salgbar vare. (Inden for økonomiens verden er salgsværdien værdien som færdig vare, og brugsværdien er værdien som mellemvare.)

Når de fleste mennesker tænker på økonomien bag software-produktion, har de en tendens til at gå ud fra en 'fabrikmodel', der er baseret på følgende fundamentale præmisser.

1. Hovedparten af udviklingstid betales af salgsværdien.
2. Salgsværdien af et stykke software er proportional med dets udviklingsomkostninger (det vil sige udgiften til de ressourcer, der skal til for at lave en kopi af dets funktioner) og dets brugsværdi.

Med andre ord har folk en stærk tendens til at antage, at software har de egenskaberne, som almindeligvis kendetegner en fremstillet var. Men det kan demonstreres, at begge disse antagelser er forkerte.

For det første: Kode skrevet med salg for øje er kun toppen af programmeringsisbjerget. I tiden før minicomputeren var det ikke usandsynligt, at 90% af al kode i verden var skrevet til eget brug i banker og forsikringselskaber. Dette er sandsynligvis ikke længere tilfældet — andre industrier er nu meget mere softwareintensive, og finanssektorens del af totalen er følgelig faldet — men vi skal om lidt se, at der er empirisk bevis for, at omkring 95% af al kode stadig skrives til eget brug.

Denne kode er blandt andet det meste af MIS, de tilpasninger til finansiel- og database-software, som enhver mellem og stor virksomhed har brug for. Den er teknisk specialiseret kode som enhedsdrivere (få tjener penge på at sælge enhedsdrivere, hvilket vi senere vil vende tilbage til). Den er alle slags indbygget kode til vore i stigende grad mikrochip-drevne maskiner — fra maskinværktøjer og jetfly til biler og mikrobølgeovne og brødrister.

Det meste af denne kode lavet til eget brug er så integreret i dets miljø, at genbrug eller kopiering er meget vanskeligt. (Dette gælder uanset om 'miljøet' er en række forretningsprocedurer i et kontor eller systemet til indsprøjtning af brændstof i en høstmaskine.) Der er således, når miljøet ændrer sig, behov for masser af arbejde, for at holde softwaren i trit med omgivelserne.

Dette kaldes 'vedligeholdelse', og enhver softwareingeniør eller systemanalytiker vil fortælle dig, at det udgør den absolutte største del (mere end 75%) af, hvad programmører får penge for at lave. Følgelig bruges de fleste programmørtimer (og det mest af programmørernes løn) på at skrive eller vedligeholde egen kode, der overhovedet ikke har nogen salgsværdi — et faktum, som læseren uden videre kan tjekke ved at undersøge listen over programmeringsjob i en hvilken som helst avis med en jobsektion.

Det er et oplysende eksperiment at gennemlæse jobsektionen i din lokale avis, hvilket jeg opfordrer læseren til at gøre. Undersøg jobbene under programmering og databehandling for stillinger, der vedrører udvikling af software. Kategoriser hvert af de jobs i henhold til, om softwaren udvikles til brug eller salg.

Det vil hurtigt blive klart, at selv med den mest rummelige definition af 'til salg' er mindst nitten af tyve af lønningerne, som tilbydes, finansieret af brugsværdi (det vil sige som mellemvare). Det er grunden til, at vi tror, at kun 5% af industrien er drevet af salgsværdier. Bemærk, at resten af analysen i dette skrift er relativt upåvirket af dette tal; hvis det var 15% eller endda 20%, ville de økonomiske konsekvenser i grunden stadig være de samme.

(Når jeg taler ved tekniske konferencer, begynder jeg som regel min tale med at stille to spørgsmål: Hvor mange blandt tilhørerne bliver betalt for at skrive software, og for hvor mange afhænger deres løn af salgsværdien af software. Jeg får som regel en skov af hænder efter det første spørgsmål, få eller ingen efter det andet spørgsmål og en stor forundring blandt tilhørerne over forholdet mellem de to.)

For det andet pulveriseres teorien om, at salgsværdien på software er knyttet til omkostningerne ved udvikling eller erstatning af det, endnu lettere ved at undersøge forbrugernes faktiske adfærd. Der er mange varer, hvor et lignende forhold gælder (før afskrivning) — fødevarer, biler, maskinelt værktøj. Der er endda mange immaterielle varer, hvor salgsværdien er tæt knyttet til omkostningerne ved udvikling eller erstatning — rettighederne til at kopiere musik, kort eller databaser til eksempel. Sådanne varer kan beholde eller endda øge deres salgsværdi efter den oprindelige sælger er væk.

Hvis en forhandler af et softwareprodukt går konkurs (eller hvis produktet tages ud af handel) falder i modsætning hertil den maksimale pris, som forbrugere vil betale for software, til nær nul uafhængigt af dets teoretiske brugsværdi eller omkostningerne ved udvikling af samme funktionalitet. (For at tjekke denne påstand kan du blot undersøge kurven med restoplæg i enhver softwareforretning i nærheden.)

Forhandlernes adfærd, når en sælger går konkurs, er meget afslørende. Den fortæller os, at de ved noget, som sælgerne ikke ved. De ved nemlig dette: Den *forventede fremtidige værdi af sælgerens ydelser* lægger effektivt låg på prisen, som en forbruger vil betale ('ydelser' er her fortolket bredt, sådet indeholder forbedringer, opgraderinger og følgeprojekter).

Med andre ord er software stort set en serviceindustri, som fungerer under den vedvarende, men ubegrundet vildfarelse, at den er en fremstillingsindustri.

Det er også værd at bemærke, at vildfarelsen om en fremstillingsindustri opfordrer til prisstrukturer, der er patologisk ude af trit med de faktiske omkostninger til udvikling. Hvis over 75% af omkostningerne i et softwareprojekts livscyklus (hvilket er almindeligt accepteret) vil være inden for vedligeholdelse, fejlfinding og udvidelse, så vil den almindelige prispolitik med at tage en høj fast indkøbspris og relativt lavt eller måske intet vederlag uundgåeligt føre til resultater, der tjener alle parter dårligt.

Forbrugerne taber, fordi, selv om software er en serviceindustri, modarbejder tilskyndelserne i fabriksmodellen, at en sælger tilbyder *kompetent* service. Hvis sælgerens indtjening fremkommer ved at sælge bits, vil den største indsats være i forbindelse med fremstillingen af bits og med at få dem ud af døren; servicecentret, som ikke er et profitcenter, vil blive en losseplads for de mindst effektive og kun få nok ressourcer til at undgå at jage et kritisk antal kunder væk.

Det modsatte aspekt af dette er, at de fleste sælgere, som anvender denne fabriksmodel, også vil gå ned med flaget. Det er kun muligt i det lange løb at finansiere fortsatte serviceomkostninger ved hjælp af en fast pris, hvis markedet udvider sig hurtigt nok til at dække omkostningerne til service og livscyklusen, der er forbundet med gårdsdagens salg ved morgendagens salg. Når markedet først modnes og salget mindskes, vil de fleste sælgere ikke have andet valg, end at skære ned i omkostningerne ved at gøre produktet forældreløst.

Hvad enten det gøres eksplicit (ved at standse salget af produktet) eller implicit (ved at gøre det vanskeligt at få service), har det den effekt, at det driver kunderne til at skifte til konkurrenterne (fordi det ødelægger produktets fremtidige værdi, som er betinget af denne service). på kort sigt kan man undgå denne fælde ved at lade frigivelsen af fejlrettelser udgive sig for nye produkter med en ny pris, men forbrugere bliver hurtigt trætte af dette. på lang sigt kan man derfor kun undslippe

dette ved ikke at have nogen konkurrenter. I sidste ende kan der kun være n på markedet.

Og vi *har* igen og igen set dette fjerne selv stærke to'ere på nichemarkedet. (Mønsteret burde være særlig klart for enhver, som har undersøgt kommercielle styresystemer til PC'er, tekstbehandlere, regnskabsprogrammer eller almindelig forretningssoftware.) De unaturlige tilskyndelser, som stammer fra fabriksmodellen, leder til en markedsdynamik, hvor vinderen får det hele, og hvor selv vinderens kunder ender med at tabe.

Hvis ikke fabriksmodellen virker, hvad virker så? For at håndtere den rigtige omkostningsstruktur i forbindelse med softwares livscyklus effektivt (både i den uformelle og den økonomiske betydning af 'effektivitet') har vi behov for en prisstruktur, som er funderet på servicekontrakter, abonnementer og en *fortsat* udveksling af værdi mellem sælger og kunde. Under de effektivitetssøgende betingelser i det frie marked kan vi derfor forudse, at det er denne form for prisstruktur, som en moden softwareindustri i sidste ende vil følge.

Ovenstående begynder at give os nogen indsigt i, hvorfor open source-software i stigende grad ikke bare udgør en teknologisk, men også økonomisk udfordring for den fremherskende orden. Resultatet af at gøre software 'fri' synes at være, at det tvinger os ind i en verden, der er domineret af servicehonorarer — og til at afsløre hvilken relativt svag støtte, salgsværdien af proprietære bits var fra starten.

Begrebet 'fri' er også vildledende på en anden måde. Nedsættelse af prisen på en vare har en tendens til at øge snarere end nedsætte den totale investering i den infrastruktur, som understøtter den. Når prisen på en bil falder, stiger efterspørgslen på mekanikere — det er derfor, at selv de 5% af programmørerne, som nu aflønnes efter salgsværdi, formentlig ikke vil miste noget i open source-verdenen. De mennesker, som taber under denne omstilling, vil ikke være programmørerne, det vil være investorerne, som har satser på proprietære strategier, hvor de ikke var egnede.

4. Myten "information vil være fri"

Der er en anden myte på linje med og i modsætning til vildfarelsen om fabriksmodellen, som ofte forvirrer folks overvejelser omkring økonomien i open source software. Det er påstanden "information vil være fri". Det afdækkes som regel som en påstand om, at en marginalomkostning ved reproduktion af information på nul medfører, at handelsprisen burde være nul.

Den mest almindelige form af denne myte er afværget ved at se på værdien af information, som gør krav på et rivaliserende gode — et skattekort, eller for eksempel et kontonummer i en schweizisk bank, eller et krav på ydelser såsom et password til en computer. Selv om informationen kan kopieres uden omkostninger, så kan genstanden det ikke. Derfor kan marginalomkostningen for genstanden overtages af informationen, som gør krav på genstanden.

Vi nævner denne myte hovedsageligt for at fastslå, at den ikke har noget at gøre med de økonomiske nytteargumenter for open source; som vi vil se senere, vil de i almindelighed være holdbare selv under antagelsen, at software faktisk *har* en fremstillingsvares værdistruktur, som er forskellig fra nul. Vi har derfor ingen grund til at takle spørgsmålet, om software 'burde' være fri eller ej.

5. Den omvendte fælle

Lad os se, om vi kan komme med en ny teori efter at have kastet et skeptisk øje på en fremherskende teori — en hård, økonomisk forklaring af, hvad der gør open source-samarbejde bæredygtigt.

Det er et spørgsmål, der kræver en undersøgelse på et par forskellige niveauer. På den ene side må vi forklare de individers adfærd, som bidrager til open source-projekter; på den anden side har vi behov for at forstå økonomiske kræfter, der gør samarbejde i open source-projekter som Linux eller Apache bæredygtigt.

Igen må vi først tilintetgøre en udbredt model, der forstyrrer forståelsen. Over ethvert forsøg på at forklare kooperativ adfærd hænger skyggen af Garret Hardins Tragedy of the Commons.

Hardin beder os om at forestille os et græsområde, der holdes af et fællesskab af bønder, hvor de lader deres køer græsse. Men græsningen nedsætter værdien af fælleden, fordi det ødelægger græsset og efterlader mudrede stykker, der kun langsomt gror frem igen. Hvis ikke der er en fælles

besluttet (og håndhævet) politik om at tildele græsningsrettigheder, som forhindrer overgræsning, vil alle parterne være tilskyndet til at have såmange køer som muligt, såde kan få mest mulig værdi ud af fælleden, før den bliver til en søaf mudder.

De fleste mennesker har en sådan lignende intuitiv model for kooperativ adfærd. Det er faktisk ikke en god diagnose af de økonomiske problemer i open source, hvilket er friløbere (manglende forsyninger) snarere fulde af offentlige goder (overforbrug). Ikke desto mindre er det den analogi, som jeg hører bag de fleste improviserede indvendinger.

Fælledens tragedie forudsiger kun tre mulige udfald. Den ene er et søaf mudder. Den anden er, at en skuespiller med tvingende magt vil håndhæve en tildelingspolitik på vegne af landsbyen (den kommunistiske løsning). Den tredje er, at fælleden deles op, efterhånden som beboerne i landsbyen indhegner stykker, som de kan forsvare og styre bæredygtigt.

Når mennesker uvilkårligt anvender denne model på open source-samarbejdet, forventer de, at det er ustabil med en kort halveringstid. Da der ikke er nogen tydelig måde at håndhæve en tildelingspolitik for programmeringstid over Internettet, fører denne model direkte til forudsigelsen, at fælleden vil deles op med forskellige dele af software, der bliver til lukket kildekode, og en stærkt faldende mængde af arbejde, der gives tilbage til den fælles pulje.

Faktisk er det empirisk tydeligt, at trenden er lige det modsatte. Bredden og volumen af open source-udvikling (som det for eksempel måles ved bidrag pr. dag hos Metalab eller annonceringer pr. dag på freshmeat.net) stiger stødt. Der er tydeligvis et sted, hvor "Tragedy of the Commons"-modellen ikke lykkes at forklare, hvad der faktisk sker.

En del af svaret ligger i den kendsgerning, at software ikke falder i værdi ved brug. Faktisk har en udbredt brug af open source-software en tendens til at øge dets værdi, efterhånden som brugerne leverer deres egne rettelser og funktioner (rettelser af koden). I denne omvendte fælled gror græsset højere, når der græsses på det.

En anden del af svaret ligger i den kendsgerning, at den formodede markedsværdi af mindre fejlrettelser til en fælles kodebase er svær at fastslå. Lad os antage, at jeg skriver en rettelse til en irriterende fejl, og lad os antage, at mange mennesker erkender, at rettelser har økonomisk værdi; hvordan indkræver jeg penge fra alle disse mennesker? Konventionelle betalingssystemer har så høje omkostninger, at de gør det til et reelt problem for den slags mikrobetalinger, som sædvanligvis ville være passende.

Det er måske mere korrekt at sige, at denne værdi ikke blot er svær at fastslå, men generelt, at den er svær at *tildele*. Lad os som et tankeeksperiment antage, at Internettet var udstyret med det teoretisk ideelle system til mikrobetalinger — sikkert, alment tilgængeligt, uden omkostninger. Lad os sige, at du har skrevet en fejlrettelse, der hedder "Forskellige Rettelser til Linux-kernen". Hvordan vil du vide, hvilken pris, du skal tage? Hvordan vil en potentiel køber, der ikke har set fejlrettelsen endnu, vide, hvad der er den rimelige pris at betale for den?

Det vi står med her er næsten som et omvendt spejlbillede af F.A. Hayek's 'regnestykke-problem' — det ville kræve et supermenneske, der kunne stoles på, både at vurdere den funktionelle værdi af fejlrettelsen og ændre priserne, sådan at handelen kunne gøres smidigere.

Uheldigvis er der stor mangel på supermennesker, så forfatteren til rettelserne, Jens Hacker, står tilbage med to valg: at holde fejlrettelsen tilbage eller kaste den gratis ind i den fælles bunke. Det første valg gavner ingen. Det andet valg gavner eventuelt intet, eller det opmuntrer måske til gensidige bidrag fra andre, der vil løse nogle af Jens Hackers problemer i fremtiden. Det andet valg, som tilsyneladende er altruistisk, er faktisk optimalt egoistisk i en strategi-teoretisk forstand.

Når vi analyserer dette slags samarbejde, er det vigtigt at bemærke, at selv om der er et problem med friløbere (der er måske mangel på ressourcer i form af penge eller penge-kompensationer), er det ikke et problem, der skalerer med antallet af slutbrugere. De faste omkostninger ved kompleksiteten og kommunikationen i et open source-projekt er næsten udelukkende en funktion af antallet af de indblandede udviklere; der er reelt ingen ekstra omkostninger ved at have flere slutbrugere, som aldrig ser på kildekoden. Det øger måske antallet af dumme spørgsmål på projektets postlister, men det er relativt let at komme i forkøbet ved at vedligeholde en liste over Ofte Stillede Spørgsmål og frejdigt ignorere de spørgere, som tydeligvis ikke har læst den (og faktisk er begge slags adfærd almindelig).

De virkelige gratist-problemer indenfor open source mere en funktion af friktionsomkostningerne ved indsendelse af fejlrettelser end noget andet. En potentiel bidragsyder, der ikke har noget at sige i det kulturelle spil (se [HtN]), kan, i fraværet af økonomisk kompensation, tænke "Det er ikke værd at indsende denne rettelse, fordi jeg skal gennemgåfejlrrettelsen, opdatere ændringsloggen og underskrive FSF's anvisningspapirer..." Det er af denne grund, at antallet af bidragsydere (og i anden række succesen af) projekter er stærkt og omvendt knyttet til antallet af besværligheder, som en bidragsyder skal gåigennem for at kunne yde et bidrag. Sådanne friktionsomkostninger kan være såvel politiske som mekaniske. Sammen kan de forklare, hvorfor den løse, abstrakte Linux-kultur har tiltrukket en samarbejdsenergi, der er større end indsatsen i det mere stramt organiserede og centraliserede BSD, og hvorfor Free Software Foundation har fået relativt mindre betydning, efterhånden som Linux er vokset.

Det er alt sammen fint, sålangt det holder. Men det er en forklaring i bagklogskabens lys af, hvad Jens Hacker gør med sin fejlrettelse, når han har den. Den anden del, som vi mangler, er en økonomisk forklaring på, hvordan JH i det hele taget var i stand til at skrive sådan en fejlrettelse i stedet for at være nødt til at arbejde på et program med lukket kildekode, der måske ville have haft en salgsværdi. Hvilken forretningsmodel skaber en niche, hvor open source-udvikling kan blomstre?

6. Grunde til at lukke kilden

Inden vi klassificerer open source-forretningsmodellen, bør vi behandle gevinsten ved at udelukke betalinger i almindelighed. Hvad er det, vi beskytter, når vi lukker kildekoden?

Lad os sige, at du ansætter en eller anden til at bestille en specialiseret regnskabspakke til din forretning. Det problem vil ikke blive løst bedre, hvis kildekoden er lukket, end hvis den var åben; de eneste rationelle grunde, du kan have, til, at den er lukket, er hvis du vil sælge pakken til andre folk eller nægte dine konkurrenter at bruge den.

Det åbenlyse svar er, at du beskytter salgsværdien, men for de 95% af software, som er skrevet til internt brug, gælder det ikke. Såhvilke andre gevinster er der ved at være lukket?

Det andet tilfælde (beskyttelse af konkurrencefordele) kan tåle en nærmere undersøgelse. Lad os sige, at du åbner kildekoden til regnskabspakken. Den bliver populær, og den forbedres på grund af rettelser fra miljøet. Nu begynder din konkurrent også at bruge den. Konkurrenten får fordelene uden at betale udviklingsomkostningerne og kan tage en del af din forretning. Er det et argument mod at åbne kildekoden?

Måske — og måske ikke. Det egentlige spørgsmål er, om fordelene ved at sprede udviklingsbelastningen overstiger dit tab ved den øgede konkurrence fra friløberen. Mange har en tendens til at resonere forkert om dette ved (a) at ignorere de funktionelle fordele ved at rekrutere mere udviklingshjælp og (b) ved ikke at betragte udviklingsomkostningerne som tabte. Hypotetisk er du nødt til at betale udviklingsomkostningerne alligevel, sådet er forkert at betragte dem som en omkostning ved at åbne kildekoden (hvis du gør det).

En anden grund, som ofte citeres, er frygten, at åbningen af koden af en særlig regnskabsfunktion er ensbetydende med at afsløre hemmelige aspekter ved din forretningsplan. Dette er ikke et argument mod lukket kode, men mod dårlig design; i en korrekt skrevet regnskabspakke bør viden slet ikke være udtrykt i koden, men i et skema eller specifikationsprog implementeret i regnskabsprogrammet (næsten det samme gælder for måden, hvorpå database-skemaer adskiller forretningsviden fra database-motorens mekanik). Adskillelsen af funktionerne vil gøre dig i stand til at forsvare kronjuvelerne (skemaet) og fåmaksimalt ud af at åbne kilden.

Der er andre grunde til at lukke kildekoden, som er direkte irrationelle. Du kan måske være under den illusion, at lukket kildekode vil gøre dine forretningsystemer mere sikre overfor crackere og andre uvelkomne personer. I såfald anbefaler jeg, at du øjeblikkeligt tager nogle terapeutiske samtaler med en kryptograf. De virkeligt professionelle paranoide ved bedre end at stole på sikkerheden i programmer med lukket kildekode, fordi de af erfaring har lært ikke at gøre det. Sikkerhed er et aspekt ved pålidelighed; du kan kun regne med at stole på algoritmer og implementeringer, som er blevet grundigt gennemgået af ligemænd.

7. Modeller til finansiering af brugsværdien

En nøglefaktor, som vi bemærker, når vi skelner mellem brugs- og salgsværdi, er, at det er kun *salgsværdien*, der er truet af et skift fra lukket til åben kildekode; brugsværdien er det ikke.

Hvis det er brugsværdien snarere end salgsværdien, som er den store drivkraft ved udvikling af software, og (som der blev argumenteret for i *Katedralen og bazeren**) open source-udvikling virkelig er mere effektiv end closed source, såburde vi forvente at finde omstændigheder, hvor brugsværdien alene finansierer open source-udvikling på en bæredygtig måde.

Og det er faktisk ikke vanskeligt at identificere mindst to vigtige modeller, hvor det udelukkende er brugsværdien, der finansierer lønninger til fuldtidsudviklere i open source-projekter.

Tilfældet Apache: deling af omkostninger

Lad os sige, at du arbejder for en virksomhed, der har et forretningskritisk behov for en højtydende og stærkt pålidelig webserver. Måske er det til elektronisk handel, måske har du en fremtrædende medieforening, der sælger reklamer, måske har du en webportal. Du har behov for 100% opetid, du har behov for hastighed, og du har brug for kunne tilpasse serveren.

Hvordan kan du opnå disse ting? Der er tre grundlæggende strategier, som du kan forfølge:

Køb en proprietær webserver. I dette tilfælde satser du på , at sælgerens dagsorden stemmer overens med din egen, og på at sælgeren har den tekniske kompetence til at implementere den ordentligt. Selv hvis begge disse punkter holder stik, vil produktet sandsynligvis ikke være tilstrækkeligt tilpasseligt; du vil kun være i stand til at modificere det gennem de steder, sælgeren har valgt. Vi kan se fra de månedlige Netcraft-undersøgelser, at denne proprietære vej ikke er særlig populær.

Byg din egen. Vi bør ikkk omgående afvise denne model; web servere er ikke særligt komplicerede, i hvert fald mindre komplicerede end browsere, og en stærkt specialiseret webserver kan være lille og effektiv. Med denne model kan du fåpræcis den funktionalitet og tilpasselighed, som du vil have, selv om du betaler for det i form af udviklingstid. Din virksomhed vil måske ogsåopdage, at den har et problem, når du trækker dig tilbage eller forlader den.

Meld dig ind i Apache-gruppen. Apache-webserveren er udviklet af en gruppe webmastere, der kommunikerede via Internettet, og som blev klar over, at det var smartere at slåderes kræfter sammen og arbejde på en kodebase frem for at bruge kræfter på et stort antal parallelle udviklingsprojekter. på den måde var de i stand til at opnå fleste af fordelene ved at bygge sin egen webserver samt den kraftige effekt af massiv fejlfinding blandt ligemænd.

Fordelen ved at vælge Apache er meget stor. Vi kan vurdere nøjagtig hvor stor ved at se på den månedlige undersøgelse fra Netcraft, som har vist, at Apache siden sin begyndelse støt har vundet markedsandele fra alle proprietære webservere. I november 2000 havde Apache og afledte produkter en markedsandel på 61%** — selv om der slet ikke er nogen juridisk ejer, ingen promovering og ingen serviceorganisation bag dem.

Historien om Apache kan generaliseres til en model, hvor brugere af software mener, at det er til deres fordel at financiere open source-udvikling, fordi det giver dem et bedre produkt til en lavere pris, end de ellers ville have.

7.2 Tilfældets Cisco: spredning af risiko

For nogle år tilbage fik to programmører hos Cisco (producenten af netværksudstyr) til opgave at skrive et distribueret system til at styre udskrivning af papir i Ciscos virksomhedsnetværk. Det var noget af en udfordring. Ud over at understøtte muligheden for, at en vilkårlig bruger A kan udskrive på en vilkårlig printer B (som kan være i rummet ved siden af eller tusinder af kilometer væk), skulle systemet sikre, at jobbet i tilfælde af manglende papir eller blæk ville blive omdirigeret til en alternativ printer nærvæd. Systemet skulle ogsåkunne rapportere sådanne problemer til print-eradministrator.

* Se http://www.laisen.dk/Katedralen_og_basare.1064.0.html.

** Se <http://www.netcraft.com/survey/>.

Duoen fandt på nogle kløgtige modifikationer* til Unix' standardsoftware til udskrivning samt nogle scripts, der løste opgaven. Såblev de klare over, at de og Cisco havde et problem.

Problemet var, at det ikke var sandsynligt at de ville blive hos Cisco til evig tid. på et tidspunkt ville begge programmører være væk, og softwaren ville ikke blive vedligeholdt og begynde at rådne (det vil sige gradvist falde ud af trit med virkelighedens krav). Ingen udvikler ønsker at se dette ske med sit arbejde, og den dristige duo mente, at Cisco havde betalt for en løsning med en ikke urimelig forventning om, at den ville holde længere end deres egne jobs der.

Følgelig gik de til deres leder og opfordrede ham til at autorisere frigivelsen af softwaren som open source. Deres argument var, at Cisco ikke havde nogen salgsværdi at miste og meget andet at vinde. Ved at opfordre til udviklingen af et miljøaf brugere og medudviklere spredt blandt mange virksomheder, ville Cisco effektivt kunne sikre sig imod tabet af softwarens oprindelige udviklere.

Tilfældet Cisco viser, at open source ikke bare kan nedsætte omkostningerne, men også sprede og mindske risikoen. Alle parter mener, at det at åbne kildekoden og tilstedeværelsen af et samarbejdende miljø, der er finansieret af mange uafhængige indtægtskilder, giver en garanti mod fejl, der i sig selv har økonomisk værdi — tilstrækkeligt til at sikre finansieringen.

8. Hvorfor salgsværdi er problematisk

Open source gør det svært at fåfat på den direkte salgsværdi af software. Det er ikke svært teknisk; kildekode er ikke sværere at kopiere end eksekverbare programmer, og opretholdelsen af copyright og licenslove, der gør det muligt at fåfat på salgsværdien, er nødvendigvis ikke sværere for open source-produkter, end for lukkede.

Besværligheden ligger snarere i naturen af den sociale kontrakt, som støtter open source-udvikling. Af tre gensidigt støttende grunde opretholder de store open source-licenser de fleste restriktioner for brug, redistribution og modifikation, som ville gøre det muligt at fange salgsværdien. For at forstå disse grunde, må vi undersøge den sociale kontekst, hvori disse licenser udvikledes; Internettet hacker**-kulturen.

på trods af myter om hackerkulturen, som stadig tros udenfor den, har ingen af disse grunde noget at gøre med fjendtlighed til markedet. Mens et mindretal af hackere forbliver fjendtlige mod profitorienterede motiver, demonstrerer den generelle villighed fra samfundet ved at samarbejde med profitorienterede Linux-forretninger som Red Hat, SuSE og Caldera, at de fleste hackere gladeligt vil arbejde med forretningsverdenen, når det tjener deres bedste. Den virkelige grund til at hackere skyr direkte indkomst-fangende licenser er mere delikate og interessante.

En af grundene skyldes symmetri. Mens de fleste open source-udviklere i grunden ikke er imod, at andre profiterer for deres gaver, kræver de fleste også, at ingen part (med undtagelsen af skaberen af koden) er i en *priviligeret* position til at fåpenge. Jens Hacker ser godt at Fubarco får profit ud af at sælge sin software og rettelser, så længe han selv potentielt også kunne gøre det.

En anden skyldes utilsigtede konsekvenser. Hackere har observeret, at licenser, som inkluderer restriktioner på og bøder for kommerciel brug eller salg har uheldige medfølger (den mest almindelige form for forsøg er at fåfat på den direkte salgsværdi, hvilket i første øjekast ikke er urimeligt). En specifik sag er at kaste en lovmæssig skygge over aktiviteter som redistribution af billige CD-ROM-antologier, hvilket vi ideelt set vil opfordre til. Mere generelt, restriktioner på brug/salg/modifikation/distribution (og andre komplikationer ved at licensere) kræver omkostninger at overholde og (som antallet af pakker, folk har, stiger) en kombinatorisk eksplosion af usikkerhed og potentiel lovmæssig risiko. Dette udfald ses på som harmfuldt, og derfor er der et stærkt socialt pres for at holde licenser enkle og fri for restriktioner.

Den sidste og mest kritiske grund har at gøre med vedligeholdelsen af den fælde-evaluerede, gave-kultur-dynamik, som er beskrevet i *Nybyggeri i noosfæren****. Licensrestriktioner designet til at beskytte intellektuel ejerskab eller at fange den direkte salgsværdi har den konsekvens at gøre

* Se <http://www.tpp.org/CiscoPrint/>.

** Se <http://www.tuxedo.org/~esr/faqs/hacker-howto.html>

*** Se http://www.laisen.dk/Nybyggeri_i_noosfaer.1065.0.html

det umuligt legalt at dele projektet. Det er tilfældet, for eksempel, med Sun's såkaldte "samfundskildekode"-licenser (Community Source, red.) for Jini og Java. Mens der ses ned på det at dele et projekt og kun som en sidste udvej (af grunde diskuteret langvarigt i *Nybyggeri i noosfæren*), såer det meget vigtigt at den sidste udvej er til stede i tilfælde af inkompetent vedligeholdelse eller et skift (for eksempel til en mere lukket licens)*

Hackersamfundet er fleksible med hensyn til symmetri-grunden; det tolererer licenser som Netscape Public License (NPL), som giver nogle profit-privilegier til de oprindelige skabere af koden (specifikt i NPL-sagen den eksklusive rettighed til at bruge open source-programmet Mozilla's kode i afledte projekter inklusiv lukket kode). Hackersamfundet er mindre fleksible med hensyn til grunden om utilsigtede konsekvenser, og slet ikke fleksible med hensyn til at opretholde muligheden af at dele (hvilket er grunden til, at Sun's Java og Jini Sun Community Source License stort set er blevet afvist af samfundet).

(Det er værd her at gentage, at ingen i hackersamfundet *ønsker* at projekter deler sig ind i konkurrerende udviklings-linjer; faktisk (som jeg bemærkede i *Nybyggeri i noosfæren*) er der et stærk socialt pres mod at dele, af gode grunde. Ingen ønsker at være i strejke, i retten eller i en krig. Men retten til at dele et projekt er ligesom retten til at strejke, retten til at sagsøge, eller retten til at holde våben — du ønsker ikke at skulle bruge nogen af disse rettigheder, men det er et seriøst farligt signal, når nogen prøver at tage dem væk.)

Disse grunde forklarer paragrafferne i Open Source Definition, hvilken er skrevet for at udtrykke konsensusen af hackersamfundet med hensyn til de vigtigste funktioner af standardlicenserne (GPL, BSD-licensen, MIT-licensen og Artistic License). Disse paragraffer har konsekvensen (dog ikke intentionen) at gøre direkte salgsværdi meget svær at fange.

9. Indirekte modeller for salgsværdi

Ikke desto mindre er det muligt at lave markeder i software-relaterede services, som fanger en indirekte salgsværdi. Der er fem kendte og to spekulative modeller af denne type (flere vil eventuelt blive udviklet i fremtiden).

Leder-tab/markedsposition

I denne model bruger du open source software til at skabe eller vedligeholde en markedsposition for proprietær software, som danner en direkte indkomststrøm. I den mest almindelige variant gør open source klient-software det muligt at sælge server-software eller tilmelding/reklame- penge associeret med et portal-site.

Netscape Communication, Inc. forfulgte denne strategi, da de open-sourcete Mozilla-browseren i begyndelsen af 1998. Deres indkomst med hensyn til browsere var på 13% og faldende, da Microsoft først udgav Internet Explorer (IE). Intensiv markedsføring af IE (og lumske pakke-metoder, som senere skulle blive den centrale sag for en antimonopol-retssag) åd straks i Netscape's browser-markedsandel, hvilket skabte den mistanke, at Microsoft prøvede at monopolisere browsermarkedet og som en konsekvens udøve kontrol over HTML og HTTP og dermed føre Netscape ud af server-markedet.

Ved at open-source den stadig udbredt populære Netscape browser, forhindrede Netscape muligheden for et browser-monopol. De regnede med, at open source-samarbejdet ville accelerere udviklingen og fejlrettelsen af browseren og håbede på, at Microsoft's IE ville komme frygteligt bagved og forhindre dem fra eksklusivt af definere HTML.

Denne strategi virkede. I november 1998 begyndte Netscape rent faktisk at indhente marked-sandele fra IE. på det tidspunkt, hvor Netscape blev erhvervet af AOL i det tidlige 1999, var den konkurrerende fordel ved at holde Mozilla i gang tilstrækkelig klar til at en af AOL's første offentlige bestræbelser at fortsætte med at støtte Mozilla-projektet, selv om det stadig var i alfa-stadiet.

Komponent-frysning

* <http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/ar01s19.html#SSH>

Denne model er beregnet til skabere af hardware (hardware inkluderer her alt ligefra Ethernet eller andre flytbare kort til komplette computersystemer). Markedspress har tvunget hardware-firmaer til at skrive og vedligeholde software (fra enhedsdrivere og konfigurationsværktøjer op til komplette operativsystemer), men softwaren selv giver ikke profit. Det er en omkostning — ofte en substantiel n.

I denne situation er det meget nemt at åbne kilden. Der er ingen indkomststrøm at tabe, og der er ingen bagside af medaljen. Det, sælgeren får, er en dramatisk større udviklermængde, kvikkere og mere fleksibelt besvarelser af kundernes behov og bedre sikkerhed gennem faglig bedømmelse. Modellen sikrer gratis programversioner til andre miljøer. Den sikrer sandsynligvis også større kundeloyalitet, eftersom kundernes tekniske hold lægger ekstra vægt på at forbedre koden, som de ønsker.

Nogle sælgere er specifikt uenige med hensyn til open source hardware-enhedsdrivere. I stedet for blande disse uenigheder med en diskussion af mere generelle sager her, har jeg skrevet en epilog* specifikt om denne sag.

”Fremtids-bevis”-effekten i open source er særlig stærk, hvad gælder komponentfrysning. Hardware-produkter har en endelig produktions og støttelivstid; efter det er kunderne på egen hånd. Men hvis de har adgang til driver-kildekode og kan rette i disse drivere, når der er brug for det, er det mere sandsynligt, at de bliver gladere stamkunder.

Et meget dramatisk eksempel på komponentfrysning-modellen var Apple Computer’s beslutning i midten af marts 1999 at åbne kilden til ”Darwin”, roden af deres Mac OS X styresystem.

Giv opskriften væk, åbn en restaurant

I denne model åbner man kilden til software ikke for at skabe en markedsposition for lukket software (som i ledertab/markedsposition-sagen), men for services.

(Jeg har før kaldte dette ’giv barbermaskinen væk, sælg barberbladene’, men koplingen er ikke helt så tæt som barbermaskine/barberblade-analogien implicerer.)

Denne model blev først brugt af Cygnus Solutions, sandsynligvis det første open source-føretagende (1989). på det tidspunkt skabte GNU-værktøjerne et fælles udviklingsmiljø på tværs af adskillige maskiner, men ethvert værktøj brugte en anderledes konfigurationsproces og krævede et anderledes sæt af rettelser for at køre på hver platform. Cygnus ”tæmmede” GNU-værktøjerne og skabte ”configure”-scripts til at centralisere byggeprocessen (opskriften) og solgte såstøtteservices og programmer pakket med deres version af GNU-værktøjerne (restauranten). De gav, i følgeskab med GPL, kunderne lov til frit at bruge, distribuere og modificere softwaren, som de distribuerede, men service-kontrakten kunne ende (eller de skulle betale en højere rate), hvis der var flere brugere, der brugte støtteservicerne, end der var planlagt i kontrakten (ingen deling ved salatbaren).

Dette er også, hvad Red Hat og andre Linux-distributører gør. Det, de rent faktisk sælger, er ikke software, bitsene selv, men værdien tillagt ved at samle og teste et kørende styresystem, som med garanti (måske kun implicit) er salgbart og er kompatibelt med andre styresystemer, der har samme brand. Andre elementer af deres værdisæt inkluderer gratis installations-støtte og muligheden af fortsat støttekontrakt.

Open source’s indflydelse på markedet kan være ekstremt kraftigt, særligt for firmaer, som uundgåeligt er i en service-position. En meget instruktiv sag i nyere tid er Digital Creations, et hjemmeside-design hus opstartet i 1998, som specialiserer sig i komplekse database- og transaktionssider. Deres hovedværktøj, firmaets intellektuelle-ejerskabs kronjuvel er et program, der har været igennem adskillige navne og inkarnationer, men som nu kaldes Zope.

Da folkene i Digital Creations søgte efter kapital, evaluerede den børsmatadoren, som de brugte, deres markedsniche, deres folk og deres værktøjer. Han anbefalede så, at Digital Creations skulle gøre Zope open source.

Efter traditionelle softwareindustri-standarder ligner dette som en absolut vild beslutning. Konventionel forretnings-skole visdom hævder, at intellektuel ejerskab som Zope er et firmas kronjuvel, som aldrig under nogen omstændigheder skal gives væk. Men børsmatadoren havde to relaterede

* Kan findes på adressen <http://www.catb.org/ēsr/writings/cathedral-bazaar/magic-cauldron/ar01s18.html>.

indsigter. n er, at Zope's virkelige værdigenstand snarere er folkets hjerner og færdigheder. En anden er, at Zope sandsynligvis skaber mere værdi som en markedsbygger, end som et hemmeligt værktøj.

For at blive overbevist om det, sammenlign to scenarier. I det konventionelle scenarie, forbliver Zope Digital Creations hemmelige våben. Lad os sige, at det er et meget effektivt våben. Resultatet bliver, at firmaet vil være i stand til at levere overlegen kvalitet i småintervaller — *men ingen ved det*. Det vil være let at tilfredsstille kunder, men sværere at bygge en kundebase i begyndelsen.

Børsmatadoren spå ede i stedet, at det kunne være en vigtig reklame for Digital Creations *virkelige* værdi, dets folk, at åbne kildekoden til Zope. Han forudsagde, at kunderne, som brugte Zope ville se det mere effektivt at ansætte eksperterne, end at udvikle intern Zope-ekspertise.

En af Zope's beslutningstagere har siden bekræftet meget åbent, at deres open source-strategi har "åbnet mange døre, vi ikke ville kunne gå ind af ellers" (ordret). Potentielle kunder svarer virkelig til situationens logik — og Digital Creations sejrer følgende.

Et andet eksempel for tiden er e-smith, inc.* Dette firma sælger støttekontrakter til brugbare Internet-server-software, som er open source, et konfigureret Linux. En af beslutningstagerne, beskrev udbredelsen af gratis downloads af e-smith's software ved at sige** "De fleste firmaer ville anse dette som piratkopiering; vi anser det som fri markedsføring".

Tilføjelser

I denne model sælger du tilføjelser til open source software. I den billige ende krus og T-shirts; i den dyre end professionelt produceret dokumentation.

O'Reilly & Associates Inc., udgiver af mange fremragende referencebøger om open source-software, er et godt eksempel på et "tilføjelses"-firma. O'Reilly ansætter og støtter rent faktisk velkendte open source-hackere (som for eksempel Larry Wall og Brian Behlendorf) som en måde at opbygge deres omdømme i dets valgte marked.

Befri fremtiden, sælg nutiden

I denne model udgiver du eksekverbare programmer og kildekode med en lukket licens, men en, som inkluderer en udløbsdato. For eksempel skriver du måske en licens, som tillader fri redistribution, forbyder kommerciel brug uden en bøde og garanterer, at softwaren kommer under GPL et år efter udgivelsen eller hvis sælgeren går fallit.

Under denne model kan kunder sikre sig, at produktet kan konfigureres til deres brug, fordi de har kildekoden. Produktet er fremtidssikret — licensen garanterer, at et open source-fællesskab kan overtage produktet, hvis det oprindelige firma dør.

Da salgsprisen og -mængden er baseret på disse forventninger fra kunderne, kan det oprindelige firma nyde godt af mere indkomst fra dets produkt, end hvis det var udgivet med en eksklusiv "lukket" licens. Ydermere vil det fåseriøs faglig bedømmelse, fejlrettelser og mindre faciliteter, hvilket fjerner noget af de 75% fra skaberen, som kræves for vedligeholdelse.

Denne model har været brugt succesfuldt af Aladdin Enterprises, skabere af det populære Ghostscript program (en PostScript-fortolker, som kan oversætte til mange printersprog).

Det største minus ved denne model er, at udløbsdatoen forhindrer faglig bedømmelse og deltagelse tidligt i produktcyklussen, præcis, hvor de er vigtigst.

Befri softwaren, sælg brand'et

Dette er en spekulativ forretningsmodel. Du åbner for kildekoden til en software-teknologi, fortsætter med at have en test suite eller et sæt af kompatibilitetskriterier, og sælger sået brand til brugerne, som sikrer, at deres implementation af teknologien er kompatibelt med alle andre med det brand.

(Dette er måden, Sun Microsystems burde havde gjort med Java og Jini.)

* Se <http://www.e-smith.net/>.

** Se <http://www.globetechnology.com/gam/News/19990625/BAND.html>.

Opdatering: I juli 2000 annoncerede Sun, at de ville gøre Star Office open source, og at de ville sælge brugen af Star Office-brandet til udviklingen af kodebasen, som er i Sun's validerings-suite.

Befri softwaren, sælg indholdet

Dette er endnu en spekulativ forretningsmodel. Forestil dig sånoget som en vare-tikkende abonnementservice. Værdien er hverken i klientsoftwaren eller i serveren, men i at give objektiv sikker information. Sådu åbner kildekoden til al software og sælger abonnementer til indholdet. Når hackere flytter klienten til nye platforme og forbedrer det på forskellige måder, udvider dit marked sig automatisk.

(Dette er grunden til, at AOL burde åbne kilden til dets klientsoftware.)

10. Hvornår man skal være åben, hvornår man skal være lukket

Nu, hvor vi har set på foretningsmodeller, som støtter open source-udvikling, kan vi adressere det generelle spørgsmål om, hvornår det er økonomisk fornuftigt at bruge åben kildekode, og hvornår den bør være lukket. Først må vi være klar over fordelene ved hver enkelt strategi.

10.1 Hvad er fordelene?

Metoden med lukket kildekode tillader dig at kræve leje for dine skjulte bits; på den anden side udelukker det muligheden for ægte, uafhængig faglig bedømmelse. Open source-metoden opsætter betingelser for faglig bedømmelse, men du får ikke leje fra dine skjulte bits.

Fordelen ved at have hemmelige bits er klar; traditionelt har software-forretningsmodeller været opbygget omkring lukket kildekode. Indtil for nyligt har fordelene ved uafhængig undersøgelse blandt ligemænd ikke været forstået tilstrækkeligt. Linux-styresystemet understreger dog, at vi formentlig skulle have lært af historien om Internettets centrale software og andre former for teknologi — at faglig bedømmelse blandt ligemænd er den eneste skalerbare måde at opnå høj pålidelighed og kvalitet.

I et konkurrerende marked vil kunder, der søger høj pålidelighed og kvalitet, derfor belønne software-producenter, der bruger open source og som finder ud af at fastholde en indtægt ved service ved at tilføje værdi og ved supplerende markeder relateret til software. Det er dette fænomen, som ligger bag Linux' forbausende succes. Linux, der kom ud af ingenting i 1996, blev det næstpopulæreste styresystem i forretnings-server-markedet i midten af 2000 (og nogle undersøgelser viste faktisk, at det passerede Microsofts plads i slutningen af 2000). I begyndelsen af 1999 forudså IDC, at Linux i løbet af 2003 ville vokse hurtigere end alle andre styresystemer tilsammen; denne forudsigelse har vist sig sand indtil videre.

En næsten lige så vigtig fordel ved open source er dets evne til at sprede åbne standarder og opbygge markeder omkring dem. Internettets dramatiske vækst skyldes det faktum, at ingen ejer TCP/IP; ingen har en proprietær nøgle til Internettets centrale protokoller.

Konsekvenserne ved TCP/IP's og Linux' succes er nogenlunde klare og kan ultimativt reduceres til sager om troskab og symmetri — potentielle parter af en delt infrastruktur kan rationelt bedre stole på det, hvis de kan se, hvordan det virker hele vejen ned, og vil foretrække en infrastruktur, hvor alle parter har ens grunde fremfor n, hvor en enkelt del har den privilegerede position at være i stand til at uddrage renter og udøve kontrol.

Men det er faktisk ikke nødvendigt at antage disse konsekvenser for at gøre symmetri-sagerne til vigtige for software-kunder. Ingen fornuftig software-kunde vil vælge at låse sig fast i en leverandør-kontrolleret monopol ved at blive afhængig af lukket kode, hvis noget åben kode af bedre kvalitet er til stede. Dette argument forstærkes, som softwaren bliver vigtigere for softwarens kundeforretning — jo vigtigere den er, desto mindre kan kunden tolerere at have det kontrolleret af en part udenfor.

Der er et modsat aspekt ved dette. Økonomer ved, at asymmetrisk information generelt får markeder til at virke dårligt. Varer med en højere kvalitet bliver drevet ud, når det er mere lukrativt at samle leje på privilegeret information, end det er at investere i at producere bedre produkter. Generelt, og ikke bare i software, er hemmelighedskræmmeri kvalitetens fjende.

Endelig er en vigtig kundefordel ved open source-software relateret til troskabssagen, at det er fremtidssikkert. Hvis koden er åben, har kunden en vej til hjælp, hvis sælgeren går ned med flaget. Dette kan være særligt vigtigt for komponentfrysning, da hardware plejer at have korte livscykluser, men konsekvensen er mere generel og medfører mere værdi for alle typer af open source-software.

10.2 Hvordan interagerer de?

Når lejen fra hemmelige bits er højere end gevinsten fra open source, gør det økonomisk mening at have lukket koden. Når gevinsten fra open source er større end lejen fra hemmelige bits, gør det mening at have åben kode.

For sig selv er dette en triviel observation. Det bliver mere besværligt, når vi ser, at fordelene ved open source er sværere at måle og forudsige, end lejen fra hemmelige bits — og når det er sagt, er fordelene underestimeret oftere end overestimeret. Indtil mainstream-forretningsverdenen begyndte at tænke på dets præmisser efter Mozillas kode-udgivelse i begyndelsen af 1998, var fordelene ved open source inkorrekt, men meget generelt anset til at være nul.

Såhvordan kan vi evaluere fordelene ved open source? Det er et generelt svært spørgsmål, men vi kan gådet i møde som ethvert andet forudsigende problem. Vi kan begynde med at observere sager, hvor open source-metoden har vundet eller fejlet. Vi kan prøve at generalisere til en model, som giver i det mindste et kvalitativt føling for kontekster, hvor open source er gavnligt for investoren eller forretningen, som prøver at få så høj gevinst som muligt. Vi kan derefter gå tilbage til dataene og prøve at refinere modellen.

Fra analysen præsenteret i *Katedralen og bazaaren** kan vi forvente, at open source er en stor fordel, hvor (a) stabilitet er vigtigt, og (b) korrektheden af design og implementation ikke kan verificeres med hjælp af andet end faglig bedømmelse. (Det andet kriterie opfyldes i praksis af de fleste ikke-trivielle programmer.)

En kundes rationelle ønske om at ikke være låst fast på en monopol-leverandør vil styrke interessen for open source (og dermed markedsværdien for at leverandører åbner kildekoden), efterhånden som softwaren bliver mere vigtig for denne kunde. Sædet andet kriterie (c) fører mod open source, når softwaren er et forretnings-kritisk kapitalgode (som for eksempel i mange informationssystemer i store firmaer).

Hvad angår programmer observerede vi ovenfor, at open source-infrastrukturen medvirker til troskab og symmetri, som, efter nogen tid, vil drage flere kunder til sig og udkonkurrere closed source-infrastrukturen; og det er bedre at have småstykker af sådanne hurtigt ekspanderende markeder, end at have et stort stykke af et lukket og stagneret et. Følgelig, for infrastruktur software, vil en open source-satsning på alsidighed vil sikkert være en større langtids fordel, end et closed-source satsning på leje fra intellektuelt ejerskab.

Faktisk medfører potentielle kunders evne til at begrunde de fremtidige konsekvenser af sælgerstrategierne og deres modvilje mod at acceptere et leverandør-monopol en større restriktion; da du ikke allerede har overvældende markedsstyrke, kan du vælge enten en open source alsidigheds-satsning eller en direkte-indkomst-fra-lukket-kode-satsning — men ikke begge. (Lignende sager af dette princip er synligt andre steder — for eksempel i elektroniske markeder, hvor kunder ofte nægter at købe designs med kun kode.) Sagen kan udtrykkes mindre negativt: Hvor konsekvenserne af netværker dominerer, er open source sandsynligvis det rette valg.

Vi kan opsummere denne logik ved at observere, at open source synes at være mest succesfuld i at få større gevinster, end closed source er, i software, som (d) etablerer en fælles computer- og kommunikations-infrastruktur.

Endelig kan vi notere, at sælgere af unikke og højt differentierede servicier har mere grund til at frygte konkurrerendes kopiering af deres metoder, end sælgere af servicier, for hvilke vigtige algoritmer og vidensbaser er godt forstået, har. Følgelig dominerer open source mere sandsynligt, når (e) nøglemetoderne (eller funktionelle metoder) er en del af fælles ingeniør-viden.

Internettets stam-software, Apache og Linux' implementationer af standard Unix API er prægtige eksempler på disse fem kriterier. Vejen mod open source i udviklingen af sådanne markeder

* Se <http://www.laisen.dk/Katedralen.og.basare.1064.0.html>.

er godt illustreret af gensynet med data-netværk på TCP/IP i midten af 1990'erne, som efterfulgte femten år med fejlede empirie-byggende forsøg på lukkede protokoller som DECNET, XNS, IPX og lignende.

på den anden side danner open source mindst mening for firmaer, som har en unik værdi-genererende software-teknologi (strengt opfyldende kriterie (e)), hvilket er (a) relativt immun overfor fejl, hvilket kan (b) straks tjekkes på andre måder end faglig bedømmelse, hvilket ikke er (c) forretningskritisk, og hvilket ikke ville få værdien substansielt forøget af (d) netværkskonsekvenser eller universalitet.

Som et eksempel på dette ekstreme tilfælde kan nævnes, at jeg i begyndelsen af 1999 blev spurgt "Skal vi åbne kilden?" af et firma, som skriver software til at beregne savmønstre til savværker, som ønsker at udnytte brædderne fra bjælkerne mest muligt. Min konklusion var "Nej." Det eneste kriterie, tilfældet nogenlunde opfylder er (c); men hvis det kniber, kan en erfaren savfører lave savmønstrene manuelt.

Bemærk, at mit svar kunne have været meget svært, hvis savmønster-beregneren var skrevet af en forhandler af savværks-udstyr. I det tilfælde ville en åbning af koden have forøget værdien af det hardware, de solgte. Bemærk også, at hvis der i forvejen eksisterede en open source mønster-beregner (måske skrevet af forhandleren af savværks-udstyr), ville det lukkede produkt have svært ved at konkurrere med det — ikke så meget på grund af prisen, men fordi kunder ville se fordelene i open source med hensyn til konfiguration og andre karaktertræk.

En vigtig pointe er, at stedet, hvor et bestemt produkt eller en teknologi sidder, kan disse skalaer ændres over tid, som vi skal se i det følgende scenarie.

For at opsummere, de følgende træk skubber hen mod open source:

- (a) Stabilitet er kritisk vigtigt.
- (b) Designets og implementationens korrekthed kan ikke straks verificeres på andre måder end uafhængig faglig bedømmelse.
- (c) Softwaren er vigtig for brugerens kontrol af hans/hendes forretning.
- (d) Softwaren etablerer en fælles computer- og kommunikations-infrastruktur.
- (e) Nøglemetoderne (eller funktionelle sådanne) er en del af fælles ingeniør-viden.

Doom: et case-studie

Historien om id software's bedst sælgende spil Doom illustrerer de måder, hvorpå pres fra markedet og produktevaluering kan kritisk ændre fordelene for lukket versus åben kildekode.

Da Doom blev udgivet første gang i det sene 1993, gjorde første persons, "real-time"-animationerne det yderst unikt (antitesen af kriterie (e)). Ikke blot var det visuelle af teknikkerne overvældende (langt overgående den flade animation i forgængeren Wolfenstein 3D), men i mange måneder kunne ingen finde ud af, hvordan det kunne gøres på de underkraftige mikroprocessorer på den tid. Disse hemmelige bits var meget leje værd. Og hvad mere er, den potentielle fordel fra open source var lav. Som et enligt spil, (a) pådragede softwaren sig tollererbare lave omkostninger i tilfælde af fejl, (b) var softwaren meget vanskelig at verificere, (c) var softwaren ikke forretningskritisk for nogen kunde, (d) profiterede ikke fra netværkskonsekvenser. Det var økonomisk rationelt for Doom at have lukket kode.

Men markedet omkring Doom stod ikke stille. Efterabende konkurrenter opfandt lignende animationsteknikker, og andre første persons skydespil som Duke Nukem begyndte at dukke op. Efterhånden som disse spil åd i Doom's markedsandel, gik lejen fra hemmelige bits ned.

på den anden side, medbragte forsøget på at udvide denne andel nye tekniske udfordringer — bedre stabilitet, flere spilfaciliteter, en større brugerbase og flere platforme. Med multiplayer 'deathmatch' komme og Doom's spilservicer, begyndte markedet at vise substentielle netværkskonsekvenser. Alt dette krævede programmørtimer, som id ville have foretrukket at bruge på det næste spil.

Fra den tid, spillet først blev udgivet, havde id set venligt på udgivelserne af tekniske specifikationer, som hjalp folk til at skabe dataobjekter til spillet, og de samarbejdede til tider direkte med hackere ved at svare på specifikke spørgsmål eller udgav deres eget specifikationsdokument. De opfordrede også til distribution af nye Doom data på Internettet.

De tekniske aspekter og markedets trend forbedrede fordelene ved at åbne koden; Doom's åbning af specifikationer og opfordring til tredjeparts-udvidelser forøgede værdien af spillet og skabte et sekundært marked, som de kunne gøre fuld brug af. på et tidspunkt krydsede fordelkurverne, og det blev økonomisk rationelt for id at tjene penge fra det sekundære marked (med produkter som spils-scenarie-antologier) og sååbne kilden til Doom. Nogen tid efter dette tidspunkt, skete det rent faktisk. Hele koden til Doom blev udgivet i det sene 1997.

At vide, hvornår man skal give slip

Doom er en interessant sag, fordi det hverken er et styresystem eller kommunikations- eller netværkssoftware; det er sålangt som nu fjernet fra de sædvanlige oplagte eksempler på succes i open source. Doom's livscyklus, komplet med det krydsende punkt, kan repræsentere applikationssoftware i nutidens kode-ekologi — n, hvor kommunikations og distribueret beregninger begge skaber seriøse robusthedsproblemer, som kun kan adresseret af faglig bedømmelse, og som hyppigt krydser grænser mellem tekniske miljøer og mellem konkurrerende aktører (med alle troskabs og symmetri-grunde, som det implicerer).

Doom udvikledes fra solo til "deathmatch"-spil. Netværkskonsekvensen er mere og mere *lig med* beregningen. Lignende tendenser er synlige i det tungeste forretningsapplikationer, såsom ERP systemer, efterhånden som forretninger netværker mere intensivt med sælgere og kunder — og, selvfølgelig, de er implicit i hele World Wide Webs artitektur. Det følger, at fordelene ved open source støt stiger næsten overalt.

Hvis disse nuværende tendenser fortsætter, vil den største udfordring for software-teknologi og produkt-behandling i det næste århundrede være at give slip — hvornår tillade at lukket kode løber ind i open source-infrastrukturen, for at udforske konsekvensen af faglig bedømmelse og fåstørre gevinster i service og sekundære markeder.

Der er selvfølgelig økonomiske incitamenter for ikke at miste skæringsspunktet for langt i den ene eller anden retning. Udover det er der en seriøs mulig risiko for at vente for længe — du kan blive skubbet ud af en konkurrent, som åbner koden i den samme markedsniche.

Grunden til at dette er vigtigt er, at både brugermængden og talentmængden, som kan rekrutteres til open source-samarbejde for en givet produktkategori er begrænset, og rekrutteringen plejer at holde. Hvis to producenter er de første og anden til at åbne kilden til konkurrerende kode med stort set samme funktion, såvil den første højst sandsynligt tiltrække flere brugere og flest motiverede udviklere; den anden bliver nødt til at nøjes med rester. Rekruttering plejer at holde, som brugere bliver kendt med produktet og udviklere nedsætter tidsinvesteringerne i koden selv.

11. Open source som et strategisk våben

Nogle gange, kan det at åbne koden ikke blot være en måde at styrke markedet, men ogsåvære en strategisk manøvre mod et konkurrerende firma. Det vil være frugtbart at undersøge nogle af forretningstaktikkerne beskrevet ovenfor igen fra sådan en vinkel; ikke direkte som indkomst-generatorer, men som en måde at bryde ind i og genskabe markeder.

Omkostningsdeling som et konkurrerende våben

Tidligere såvi på Apache som et eksempel på bedre og billigere infrastruktur-udvikling gennem omkostningsdeling i et open source-projekt. For software- og systemsælgere, der konkurrerer mod Microsoft og ISS webserveren, er Apache-projektet ogsået konkurrerende våben. Det ville være svært, måske endda umuligt, for en enkelt sælger af webservere fuldkomment at fåfordelene ved Microsofts store kampagnepenge og skrivebordsmonopol markedskraft ned. Men Apache gør enhver stor deltager i projektet i stand til at udbyde en webserver, som er både teknisk bedre end IIS og forsikrer kunder med en stor markedsandel — til en langt billigere pris. Dette forbedrer markedspositionen og produktionsomkostningerne for værdiforøgede elektronikprodukter (som IBM's WebSphere).

Dette kan generaliseres. Åbne og delt infrastruktur giver deltagerne nogle fordele. En er en lavere omkostning per deltager til at producere salgbare produkter og services. En anden er en markedsposition, som forsikrer kunder om, at de højst sandsynligt ikke bliver låst fast med forældreløst teknologi som resultat af en sælgers ændring af strategi eller taktik.

Nulstilling af konkurrencen

Da DEC i 1980'erne støttede udviklingen af open source X-vinduesystemet, var deres mål at 'nulstille konkurrencen'. på det tidspunkt var der mange konkurrerende alternative grafikmiljøer til Unix i gang, inklusiv Sun Microsystems' NeWS system. DEC strateger troede (sandsynligvis korrekt), at hvis Sun var i stand til at etablere en proprietær grafikstandard, ville de få kontrol over det blomstrende Unix-arbejdsstations-marked. Ved at støtte X og udlåne ingeniører, og ved at alliere sig med mange småsælgere til at etablere X som en de-facto standard, var DEC i stand til at neutralisere Suns og andre konkurranter fordele med mere intern ekspertise i grafik. Dette flyttede fokus fra konkurrencen i arbejdsstations-markedet til hardware, hvor DEC var stærk historisk.

Ogsådette kan generaliseres. Open source er attraktivt for kloge kunder og til potentielle allierede, som ikke er store nok til selv at opretholde konkurrerende udvikling. Et open source-projekt, opsat på rette tidspunkt, kan gøre bedre, end bare at konkurrere succesfuldt mod alternativer med lukket kode; det kan faktisk forhindre dem fra at få del i markedspladsen, nulstille konkurrencen og om dirigere det fra et område, hvor det begyndende firma er for svagt, til, hvor det er stærkt.

Lade græsset gro

Red Hat Software støttede udviklingen af RPM-pakkesystemet, for at give Linux-verdenen et standard pakke-installations-program. Ved at gøre det, var de sikre på , at den øgede tro på sådan et program ville give potentielle kunder, ville være mere værd i fremtidige kroner, end udviklingsomkostningen af softwaren eller indkomst-potentialet tabt til konkurrenter, som også var i stand til at bruge det. Nogle gange er den klogeste måde at blive en større frøat lade græsset gro hurtigere. Dette er selvfølgelig den økonomiske grund til at teknologi-firmaer har deltaget i offentlige standarder — og det kan være nyttigt at tænke på open source software som en eksekverbar standard. Ud over at være en fremragende markedsbygger, kan denne strategi være et direkte konkurrerende våben, når et lille firma bruger det til at nedsætte massen og markedskræften af et større firma udenfor den standard-baserede balance. I tilfældet Red Hat er den klare konkurrent Microsoft; standardiseringen af RPM på de fleste Linux-distributioner førte en lang vej mod at neutralisere de fordele, som Microsoft tidligere havde i den enkle systemadministration på dets Windows-maskiner.

12. Open source og strategisk forretningsrisiko

Ultimativt har grunden til, at open source synes bestemt til at blive en udbredt praksis, mere at gøre med markedspress fra kunder, end med sælgerens materialeeffektivitet. Jeg har allerede, fra sælgerens synsvinkel, diskuteret konsekvenserne af kundepres for stabilitet og infrastruktur uden en enkelt dominerende spiller, og hvordan disse har udspillet sig historisk i udviklingen af netværk. Der skal dog siges mere om kunders adfærd i et marked, hvor open source er en faktor.

Put dig for et øjeblik i en teknisk direktørs position ved et Fortune 500-firma, søgende efter at bygge eller opgradere dit firmas IT-infrastruktur. Måske skal du vælge et netværksstyresystem, som skal bruges over hele firmaet; måske involverer dine overvejelser fuldtids-webservice og online-betalinger; måske kræver din forretning at være i stand til at have højstabilitets transaktions-databaser.

Antag, at du går den konventionelle rute med lukket kode. Hvis du gør det, sålægger du dit firma i hænderne på et sælgermonopol — fordi der per definition kun er t sted, du kan gåhen for at få støtte, fejlrettelser og udvidelser. Hvis din sælger ikke gør det godt, vil du ikke have nogen effektiv ressource, fordi du er låst fast mellem din tidlige investering og trænings-omkostninger. Din sælger ved dette. Under disse omstændigheder, tror du så, at softwaren vil ændre til at møde dine behov og din forretningsplan...eller din sælgers behov og din sælgers forretningsplan?

Den brutale sandhed er: Når dine hoved-forretningsprocesser køres med solide bits, som du ikke engang kan se indefra (og slet ikke modificere) har du tabt kontrollen af din forretning. Du har mere brug for din sælger, end din sælger har brug for dig — og du vil komme til at betale og betale og betale igen for den ubalance. Du vil kaste flere penge ind, du kommer til at betale tabte muligheder, og du vil betale efter en aftale, hvilket bliver værre over tid, efterhånden som sælgeren (som har refined dets spil på mange tidligere ofre) snærer sit greb.

Sammenlign dette ved valget at gåopen source. Hvis du går ad den vej, har du kildekoden, og ingen kan tage den væk fra dig. I stedet for et sælgermonopol med fat om halsen på din forretning, har du nu flere service-firmaer, som byder på din forretning — og du får dem ikke alene til at spille mod hinanden, du har også muligheden for at bygge din egen kontrollerede støtteorganisation, hvis det ser mindre dyrt ud, end at forlade markedet. Markedet arbejder for dig

Logikken er spændende; der følger en uacceptabel strategisk forretningsrisiko med lukket kildekode. Så stor en risiko, at jeg tror, at det ikke vil vare længe, før anskaffelsen fra ”lukkede” enkelt-sælgere, når der er et open source alternativ tilgængeligt, vil blive set på som uansvarligt, og være en retfærdig grund til en retssag.

13. Open sources forretningsøkologi

Open source-miljøet har organiseret sig på en måde, som tydeliggør de produktive konsekvenser af open source. I Linux-verdenen i særdeleshed, er det et økonomisk vigtigt faktum, at der er flere konkurrerende Linux-distributører, som danner en mængde adskilt fra udviklerne.

Udviklere skriver kode, og gør den tilgængelig på Internettet. Hver distributør vælger en del af den tilgængelige kode, integrerer og pakker den, gør den til et brand og sælger den til kunderne. Brugere vælger mellem distributioner og kan supplerere en distribution ved at downloade kode direkte fra udviklersider.

Konsekvensen af denne adskillelse er et meget flydende internt marked for forbedringer. Udviklere konkurrerer mod hinanden om kvaliteten af deres software, hvilket distributører og brugere bemærker. Distributører konkurrerer for bruger-kroner om, hvor tilpas deres udvælgelse er og om værdien, som de kan tilsætte softwaren.

En første konsekvens af denne interne markedsstruktur er, at ethvert punkt i nettet kan erstattes. Udviklere kan droppe ud; selv om deres del af kodebasen ikke tages op af en anden udvikler, vil konkurrencen om bemærkelse straks generere funktionelle alternativer. Distributører kan fejle uden at ødelægge eller gå på kompromis med den fælles open source-kodebase. Økologien som helhed har en hurtigere markedsrespons til markedsbehov og bedre evne til at stå mod chok og regenerere sig selv, end en stor sælger af et ”lukket” styresystem ville have.

En anden vigtig konsekvens er at nedsætte omkostninger og forøge effektiviteten gennem specialisation. Udviklere oplever ikke det pres, som rutinemæssigt kommer fra lukkede projekter og gør dem til mudderpøle — ingen lister af ubrugelige og distraherende tjekliste-faciliteter fra Marketing, ingen mandater, som vil bruge upasselige og døde sprog eller udviklingsmiljøer, intet behov for at opfylde den dybe tallerken på en ny og inkompatibel måde for at produktdifferentiere og beskytte det intellektuelle ejerskab, og (mest vigtigst) ingen deadlines. Ingen hast for at få version 1.0 ud af døren, før det er gjort rigtigt. De Marco og Lister observerede i deres diskussion af ”væk mig, når det er overstået”-forretningsstil i ”Peopleware: Productive Projects and Teams” [DL], at dette generelt bringer ikke bare højere kvalitet, men rent faktisk en hurtigere levering af et virkende resultat.

Distributører får, på den anden side, mulighed for at specialisere sig i områder, som distributører kan gøre mest effektivt. De der er befriet fra behovet af at støtte massive software-udviklinger bare for at forblive konkurrerende, kan de koncentrere sig om system-integration, pakning, kvalitetsforsikringer og service.

Både distributører og udviklere holdes i hævd ved konstant feedback fra brugerne og deres observation, hvilket er en del af open source-metoden.

14. Håndtering af succes

”Tragedy of the Commons” kan måske ikke bruges på open source-udvikling, som det sker i dag, men det betyder ikke, at der ikke er en grund til at undre sig over om nudagens ryk fra open source-samfundet kan bevares. Vil hovedpersonerne gånede, efterhånden som budet forhøjes?

Dette spørgsmål kan besvares på flere forskellige planer. Vores omvendte ’Comedy of the Commons’-historie er baseret på argumentet, at det er svært at sætte pris på individuelle bidrag til

open source. Men dette argument har meget mindre styrke for firmaer (som for eksempel Linux-distributører), som allerede har en indkomststrøm associeret med open source. Prisen for deres bidrag bliver allerede beregnet hver dag. Er deres nuværende samarbejdsrolle stabil?

Ved at undersøge dette spørgsmål, vil vi blive ledt til nogle interessante indsigter om open source-sofwarens økonomi i den virkelige verden i dag — og om, hvad en sand service-industri-paradigme medfører for software-industrien i fremtiden.

på det praktiske plan, til open source-samfundet, som det eksisterer nu, kan dette spørgsmål stilles på n ud af to forskellige måder. t: Vil Linux fragmentere? To: Vil Linux udvikle en dominerende pseudo-monopolistisk spiller?

Den historiske analogi, som mange folk henvender sig til, når de tænker på , om Linux vil fragmentere, er adfæren fra proprietære Unix-sælgere i 1980'erne. på trods af endeløs snakken om åbne standarder, på trods af utallige alliancer og sammenslutninger og enigheder, faldte det proprietære Unix fra hinanden. Sælgernes ønske om at differentiere deres produkter ved at tilsætte og modificere styresystemsfaciliteter viste sig at være stærkere, end deres interesse for at gøre det samlede Unix-markede større ved at opretholde kompatibilitet (og dermed nedsætte både indgangsbarriererne for uafhængige software-udviklere og den samlede omkostning af ejerskab for kunder).

Dette sker højst sandsynligt ikke for Linux af den simple årsag, at alle distributører er tvunget til at handle fra en fælles base af open source-kode. Det er ikke muligt for nogen af dem at vedligeholde differentiationen, for licenserne, som Linux udvikles under, kræver, at de deler kode med alle dele. Tidspunktet, hvor en distributør udvikler en facilitet er alle konkurrenter frie til at klonet det.

Da alle parter forstår dette, tænker slet ingen på at udføre disse former for manøvrer, som fragmenterer det proprietære Unix. I stedet er Linux-distributører tvunget til at konkurrere på måder, som faktisk gavner kunden og det samlede marked. Det vil sige, de skal konkurrere om service, om støtte og deres design-valg om, hvilke grænseflader medvirker til en nemmere installation og brug.

Den fælles kodebase udelukker muligheden for monopolisation. Nr Linux-folk bekymrer sig om dette, nævnes navnet "Red Hat" oftest, som er den største og mest succesfulde distributør (med rundt regnet 90estimeret markedsandel i USA). Men det er værd at bemærke, at i dagene efter annonceringen i maj 1999 af Red Hat's længe ventede 6.0-udgivelse — før Red Hat's CD-ROM'er blev udgivet — blev der reklameret for CD-ROM-skiverne bygget fra Red Hat's egen offentlige FTP-side af en forlægger og mange andre CD-ROM-distributører med lavere pris end Red Hat's forventede pris.

Red Hat selv forblev uforandrede, fordi deres grundlæggere meget klart forstod, at de ikke kan eje bitsene i deres produkt; Linux-samfundets sociale norm forbyder det. I en parodi på John Gilmores berømte betragtning, at Internette fortolker censur som ødelæggende og går en vej rundt om det, er det blevet sagt, at hacker-samfundet, som er ansvarligt for Linux fortolker forsøg på kontrol som ødelæggende og finder en vej rundt om det. Hvis Red Hat have protesteret over kloning af deres produkter, som ikke var udgivet endnu, ville det seriøst havde kompromiteret deres evne til at fåfremtidig samarbejde fra udvikler-samfundet.

Måske mere vigtig for tiden er, at software-licenserne, som udtrykker disse samfundsnormer i en bindende gyldig form aktivt forbyder Red Hat fra at monopolisere deres kode, som deres produkter er baseret på . Det eneste, de kan sælge er et brand/service/støtte-forhold med folk, som er villige til at betale for det. Dette er ikke en kontekst, hvor muligheden af et udnyttende monopol kommer stærkt til syne.

15. Åben forskning og udvikling og genopfindelsen af støtte

Der er en andet sted, hvor inddragelsen af rigtige penge i open source-verdenen ændrer den. Samfundets stjerne ser mere og mere, at de kan blive betalt for, hvad de har lyst til at lave, i stedte for at have open source som en hobby ved siden af deres almindelige arbejde. Firmaer som Red Hat, O'Reilly & Associates og VA Linux Systems bygger, hvad svarer til næsten uafhængige research-våben med tvang til at ansætte og vedligeholde open source-talent.

Dette giver kun mening økonomisk, hvis omkostningen per hovede af at vedligeholde sådan et laboratorie let kan betales af de forventede gevinster, som det vil fåaf at lade firmaets marked gro

hurtigere. O'Reilly har råd til at betale ledere af Perl og Apache til at gøre deres ting, fordi de forventer, at deres bestræbelser vil gøre dem i stand til at sælge flere Perl og Apache-relaterede bøger og drage flere folk til deres konferencer. VA Linux Systems kan støtte deres laboratorie-afdeling, fordi brugsværdien for arbejdsstationerne og serverne, som de sælger, forstørres, når de forbedrer Linux. Og Red Hat støtter Red Hat Advanced Development Labs til at forøge værdien af deres Linux-ydelser og tiltrækker flere kunder.

For strategier fra mere traditionelle sektorer af software-industrien opfostret i kulturer, som ser på patent-sikret intellektuelt ejerskab som kronjuveler, ser denne slags adfærd (på trods af den markedsvoxsende konsekvens) uforklarlig ud. Hvorfor støtte research, som dine konkurrenter (per definition) er frie til at bruge uden omkostning?

Der synes at være to kontrollerende grunde. En er, at så længe som disse firmaer forbliver dominante spillere i deres markedsniche, kan de forvente at fange en proportional stor del af gevinsterne fra deres åbne research og udvikling. At bruge åben forskning og udvikling for at få fremtidige gevinster er ikke en ny id; det interessante er den medfølgende beregning, at de forventede fremtidige gevinster er tilstrækkeligt store til at disse firmaer kan tolerere gratister, for at få faglig bedømmelse.

Mens denne selvfølgelige forventede-værdi-analyse er nødvendig i en verden med hårdhændede kapitalister, som fokuserer på investeringens afkast, er det faktisk ikke den mest interessante forklaring på ansættelsen af stjerner, når firmaerne selv har en mere uklar selv. De vil, hvis de bliver spurgt, fortælle dig, at de simpelthen gør det rigtige fra samfundet, de kom fra. Din ydmyge forfatter er tilstrækkeligt velkendt med beslutningstagere fra de tre firmaer citeret ovenfor, til at sige, at disse protester ikke kan frasiges som løgn. Jeg blev selv hyret til VA Linux Systems i slutningen af 1998, for at jeg kunne være til rådighed til at advisere dem om "det rigtige", og de har været langt fra uvillige til at høre efter, når jeg gjorde det.

En økonom vil spørge, hvilke fordele der er her. Hvis vi accepterer, at snakken om at gøre det rigtige ikke er tom snak, må vi finde frem til, hvilket interesse fra firmaet, som "det rigtige" leverer. Svaret i sig selv er hverken overraskende eller svært at verificere ved at stille de rigtige spørgsmål. Som det gælder for altruistisk adfærd i andre industrier, tror disse firmaer rent faktisk, at de køber samarbejdsvillig attitude.

At arbejde for at tjene denne attitude og sætte pris på som en fremtidig markedsværdi er heller ikke en ny id. Det interessante er de ekstremt høje værdier, som disse firmaer med deres adfærd viser, at de lægger i den samarbejdsvillige attitude. De er, for at demonstrere, villige til at ansætte dyrt talent for projekter, som ikke er direkte indkomst-generatorer, selv i de mest kapital-hungrende faser indtil børsintroduktionen. Og markedet har, i det mindste indtil videre, rigt belønnet denne adfærd.

16. At komme dertil herfra

Markedsmekanismerne for at støtte (og lave profit fra) open source-udvikling udvikler sig stadig hurtigt. Forretningsmodellerne, som vi har set på i dette skrift, vil sandsynligvis ikke blive de sidste. Investorer tænker stadig konsekvenserne igennem af at genopfinde softwareindustrien som n med eksplicit fokus på service snarere end lukket intellektuel ejerskab, og vil være det i nogen tid fremover.

Denne konceptuelle revolution vil have en omkostning i profitter for folk, som investerer i de 5% salgsværdi af industrien; Historisk set er service-forretninger ikke så lukrative som fremstillingsforretninger (selv om enhver doktor eller advokat kan fortælle dig, at gevinsten til praktiktionærene ofte er højere). Enhver førtidig profit vil dog mere end matches at gevinsterne på omkostningssiden, efterhånden som softwarekunder høster utallige mønter og effekter fra open source-produkter. (Der er her en parallel til konsekvenserne, som erstatningen af det traditionelle stemme-telefon-netværk af Internettet har allesteder.)

Løftet om disse mønter og effekter skaber en markedsmulighed, som entreprenører og kapitalister nu rykker ind for at udforske. Da den første kladde af dette skrift var i forberedelse, tog Silicon Valley's mest respekterede kapitalist-firma lederpositionen i det første opstartfirma, som specialiserede sig i teknisk support for Linux døgnnet rundt (Linuxcare). I august 1999 var Red Hat's børsintroduktion (trods et fald i Internet- og teknologiaktier) helt vildt succesfuldt. Det forventes, at

mange Linux- og open source-relaterede børsintroduktioner vil blive børsintroduceret, før slutningen af 1999 — og at disse også vil blive ganske succesfulde. (År 2000-opdatering: det var de!)

En anden meget interessant udvikling er begyndelsen på systematiske forsøg på at lave opgavemarkeder i open source-udviklingsprojekter. SourceXchange* og CoSource** repræsenterer lidt andre måder at prøve på at bruge en omvendt auktionsmodel for at støtte open source-udvikling.

Den generelle tendens er klar. Vi nævnte tidligere, IDC spåede, at Linux vil vokse hurtigere end alle andre styresystemer sammenlagt gennem 2003. Apache har en markedsandel på 61%*** og vokser støt. Brugen af Internettet eksploderer, og undersøgelser som Internet Operating System Counter† viser, at Linux og andre open source-styresystemer allerede er på Internet værter og støt får markedsandel fra lukkede systemer. Behovet for at udforske Internettets open source-infrastruktur nødvendiggør ikke bare designet af andet software, men ogsåforretningspraktisserne og software brugen og købmønstrene af enhver sammenslutning, der eksisterer. Disse tendenser synes bare at accelerere.

17. Konklusion: Livet efter revolutionen

Nogle programmører er bekymrede for, at transitionen til open source vil fjerne eller nedsætte værdien af deres arbejde. Det almindelige mareridt er, hvad jeg kalder "Open Source Dommedag"-scenariet. Dette starter med, at markedsværdien af software går ned på nul på grund af al den frie kode. Brugsværdien alene tiltrækker ikke nok kunder til at støtte software-udvikling. Den kommercielle software-industri kollapser. Programmører sulter og forlader faget. Dommedag kommer, når open source-kulturen selv kollapser, hvilket efterlader ingen til at programmere kompetent. Alle dør. Åh, hvilken skæbne!

Vi har allerede observeret nogle tilstrækkelige grunde for at dette ikke vil ske, begyndende med det faktum, at det meste udviklerløn ikke afhænger af softwares salgsværdi. Men den allerbedste, som er værd at understrege her, er denne: Hvornår så du sidst en software-udviklingsgruppe, som ikke have meget mere end nok arbejde? I en hurtigt forandrende verden, i en hurtigt komplicerende og informations-centreret økonomi, vil der altid være meget arbejde og en sundt krav om folk, som kan fåcomputere til at gøre ting — ligemeget hvor meget tid og hvor mange hemmeligheder, de giver væk.

For at undersøge software-markedet selv, vil det være en hjælp at sortere typer af software efter, hvor meget den service de udbyder kan beskrives af åbne tekniske standarder, hvilket er relateret til hvor meget den underliggende service kan betragtes som en handelsvare.

Denne akse svarer meget vel til, hvad folk almindeligvis tænker, når de snakker om 'applikationer' (som slet ikke betragtes som varer, og som er svage eller ikke-eksisterende åbne tekniske standarder), 'infrastruktur' (servicer, stærke standarder) og 'middleware' (delvist betragtet som handelsvarer, effektive, men ukomplette tekniske standarder). Paradigme-sagerne i dag i 2000 ville være et tekstbehandlingssystem (applikation), en TCP/IP-stak (infrastruktur) og en database-motor (middleware).

Fordelsanalysen, som vi foretog tidligere, foreslår, at infrastruktur, applikationer og middleware bliver transformeret i forskellige veje og udøver forskellige magtbalancer bestående af åbent og lukket kildekode. Den foreslår også, at det udbuddet af open source i et særligt software-område vil være en funktion af, hvorvidt der er substentielle netværkskonsekvenser til steder der, hvad fejlmkostningerne er og i hvor høj grad softwaren er et forretnings-kritisk kapitalgode.

Vi kan udsige nogle forudsigelser, hvis vi benytter disse heuristiske metoder ikke til individuelle produkter, men til hele segmenter af softwaremarkedet. Lad os begynde:

Infrastruktur (Internettet, styresystemer og de lavere lag af kommunikationssoftware, som er nødt til at krydse grænser mellem konkurrerende parter) vil blive næsten helt open source, opretholdt

* Se <http://www.sourceexchange.com/process.html>.

** Se <http://www.cosource.com/>.

*** Se <http://www.netcraft.com/survey/>

† Se <http://leb.net/hzo/ioscount/>.

af bruger associationer og af profitorienterede distributions/service grupper med en rolle som den, Red Hat har i dag.

Applikationer vil på den anden side have tendensen til at forblive lukkede. Der vil være omstændigheder, hvor brugsværdien af en skjult algoritme eller teknologi vil være høj nok (og hvor omkostningerne associeret med ustabilitet vil være lave nok, og risikoen associeret med en sælgermonopol vil være tilstrækkelig tålerbare) til at kunder vil fortsætte med at betale for lukket software. Dette er mest sandsynligt for en applikation, hvor netværkskonsekvenserne er svage. Vores savværks-eksempel tidligere er sådan n; biometrisk identifikationssoftware synes, med 1999'ernes forudsigelser, at være endnu n.

Middleware (som databaser, udviklingsværktøjer eller den konfigurerbare top af applikations-protokol-stakke) vil være mere blandede. Hvorvidt middleware-kategorier vil være lukkede eller åbne synes at afhænge af fejlomkostningerne, hvor den højere omkostning skaber markedspress for mere åbenhed.

For at male billedet færdigt er vi god nødt til at bemærke, at hverken 'applikaioner' eller 'middleware' virkelig er stabile kategorier. Tidligere såvi, at individuelle software-teknologier synes at gågennem en naturlig livscyklus fra rationelt lukket til rationelt åben. Den samme logik kan bruges på større områder.

Applikationer synes at falde i kategorien middleware, efterhånden som standardiserede teknikker udvikles og portioner af servicen bliver gjort til en handelsvare. (Eksempelvis blev databaser midlware efter SQL koplede brugerflader fra motorer.) Når middleware-servicer bliver gjort til handelsvarer, vil de såfalde ind i open source-infrastrukturen — en transition, som vi ser i styresystemer lige nu.

I en fremtid, som inkluderer konkurrence fra open source, kan vi forvente, at enhver software-teknologis skæbne vil være enten af døeller at blive en del af den åbne infrastruktur selv. Mens dette næppe er gode nyheder for entreprenører, som for altid vil samle leje fra lukket software, foreslår det dog, at softwareindustrien som en helhed vil forblive entreprenør-agtig, men nye nicher konstant dukkende frem ved den øvre (applikation) ende og et begrænset livstid for lukkede IP-monopoler, da deres produkt-kategorier falder ind i infrastruktur.

Endelig vil denne magtbalance selvfølgelig være storartet for softwarekunder, som driver processen. Mere og mere højkvalitets software vil blive permanent tilgængeligt til at bruge og bygge på i stedet for at være ufærdiggjort og låst inde i nogens kælder. Ceridwen's magiske kedel er, endelig, for svag en metafor — fordi mad konsumeres eller rådner, hvorimod software-kode potentielt varer for evigt. Det frie marked kan, i den videste liberale forstand inkluderende al ulokkende aktivitet, om det er handel eller gave, producere gentagen software-rigdom for alle.

18. Appendix: Hvorfor lukkede drivere koster en sælger penge

Skabere af flytbart hardware (Ethernet-kort, disk-kontrollere, grafikkort og lignende) har historisk været uvillige til at åbne op. Dette ændrer sig nu, hvor spillere som Adaptec og Cyclades rutinemæssigt begynder at åbne specifikationer og driver-kildekode op for deres kort. Ikke desto mindre er der stadig resistans derude. I dette appendix prøver jeg at fjerne mange af de økonomiske fordomme, som opretholder denne resistans.

Hvis du er en sælger af hardware, kan du frygte, at åbningen af kilden vil afsløre vigtige ting om, hvordan din hardware opererer, som konkurrerende kan kopiere, hvilket giver en unfair konkurrerende fordel. Tilbage i dagene med tre- til fireårs produktcykluser var dette et gyldigt argument. I dag er den tid, ingeniører skulle bruge for at kopiere og forståkopien en dramatisk stor portion af produktcyklussen, tid, som de ikke bruger på at innovere eller differentierer deres eget produkt.

Dette er ikke noget nyt. Den tidligere KGB-chef Oleg Kalugin siger dette godt*:

”For eksempel, da vi stjalte IBM'er i vores blåstempler eller andre elektroniske områder, hvor vesten gjorde store fremskridt og hvor vi var bagefter, ville det tage år at implementere resultaterne

* Se <http://cnn.com/SPECIALS/cold.war/experience/spies/interviews/kalugin/>.

af vores intelligens-bestræbelser. Ved det tidspunkt, på fem eller syv år, gik vesten forrest, og vi skulle stjæle igen og igen, og vi faldt tilbage mere og mere.”

Men Rudyard Kipling sagde det bedre i sit digt *The Mary Gloster** næsten et århundrede siden. Han skrev:

Og de spurgte mig hvordan jeg gjorde det, og jeg gav dem Skriften, 'Du lader dit lys sådan skinnende foran den næste!' De kopierede alt de kunne følge, men de kunne ikke kopiere mit sind, Og jeg lod dem svede og stjæle t og et halvt år bagved.

Acceleration til Internet-tid gør denne effekt lidt sværere. Hvis du virkelig er foran, vil efterligning være en fælde, som du vil have at dine konkurrenter falder i!

Disse detaljer forbliver ikke skjult i lang tid i disse dage. Hardware-drivere er ikke som styresystemer eller applikationer; de er små, lette at skille ad og lette at klonе. Selv teenage-programmører kan gøre dette — og gør det ofte.

Der er bogstavelig talt tusinder af Linux og FreeBSD-programmører derude med både evnen og motivationen til at bygge drivere for et nyt kort. For mange klasser af enheder, som har relativt enkle grænseflader og velkendte standarder (som diskkontrollere og netværkskort) kan disse ivrige hackere ofte teste en driver næsten så hurtigt, som dit eget firma kunne, selv uden dokumentation og uden af skille en eksisterende driver ad.

Selv for ”tricky” enheder som grafik- og lyd kort er der ikke meget, du kan gøre for at forhindre, at en kløgtig programmører med en disassembler skriver en driver. Omkostningerne er lave og de legale barrierer er usikre; Linux er et international foretagende og der er altid et lovområde, hvor ”reverse-engineering” vil være lovligt.

For bevis på at alle disse på stande er rigtigt, såundersøg listen af enheder understøttet af Linux-kernen og læg mærke til tempoet, hvorved nye bliver tilsat til kernen selv uden sælger-support.

En anden god grund til at åbne op for dine drivere er, at du såkan koncentrere dig om innovation. Forestil dig, at du ikke længere bliver nødt til at bruge dine arbejders tid og løn på at genskrive, teste og distribuere nye programmer for hver ny kerne, efterhånden som den udkommer. Du har helt sikkert bedre ting at gøre med al den talent.

Endnu en god grund: Igen ønsker at vente seks måneder på fejlrettelser. Hvis du overhovedet har en open source-konkurrence, vil de sandsynligvis begrave dig af denne grund alene.

Selvfølgelig er der fremtidssikrings, som tidligere er nævnt. Kunder ønsker open source, fordi de ved, at det vil udvide livstiden af hardwaren såmeget, at det er produktivt for dig at støtte det.

Den bedste grund er dog, at du får penge for at sælge hardware. Der er ikke noget markedskrav for din hemmelighedsfuldhed; faktisk lige omvendt. Hvis dine drivere er svære at finde, hvis de er nødt til at blive opdateret hyppigt, hvis de (værst af alt) kører dårligt, reflekterer det dårligt på din hardware, og du vil sælge mindre af den. Open source kan løse disse problemer og sætte skub i dine indkomster.

Beskedene? At holde din driver hemmelig ser attraktivt ud i det korte løb, men er sandsynligvis en dårlig strategi i det lange løb (i særdeleshed, når du konkurrerer med andre sælger, som allerede er åbne). Men hvis du mågøre det, såbrænd koden ind i en ROM indbygget på kortet. Udgiv sågrænsefladen til ROM'en. Åbn såmeget som muligt, for at bygge dit marked og demonstrer til potentielle kunder, at du tror på din kapacitet til at udtænke og ud-innovere konkurrenter, hvor det gælder.

Hvis du forbliver lukket, vil du sædvanligvis fådet værste af alle verdener — dine hemmeligheder vil stadig blive udsat, du vil ikke fågratis udviklingshjælp, og du vil ikke have mistet din dummere konkurents tid på kloning. Vigtigst af alt mister du muligheden for tidlig adoption. Et stort og indflydelsesrigt marked (folkene, som håndterer serverne, som kører stort set hele Internettet og mange forretningsdata-centre) vil på korrekt frasige dit firma som dumt og defensivt, fordi du ikke ser disse ting. Derefter vil de købe deres kort fra nogen der gør.

19. Noter

* http://www.everypoet.com/archive/poetry/Rudyard.Kipling/kipling.the_mary_gloster.htm

[SC] Underprovisionsproblemet ville rent faktisk skalere lineært med antallet af brugere, hvis vi antog, at programmeringstalentet er ligeligt distribueret i brugerpopulationen, efterhånden som det udvider sig. Dette er dog ikke tilfældet.

Incitamenterne diskuteret i Nybyggeri i noosfæren [HtN] (og nogle mere konventionelt økonomiske) implicerer, at kvalificerede personer plejer at søge efter projekter, som matcher deres interesser, ligesom projekterne søger dem. Følgende, teorien foreslår (og erfaring vil konfirmere), at de mest værdifulde (bedst kvalificerede og motiverede) personer vil finde projekterne, som de passer godt sammen med relativt tidligt i projektets livscyklus, og derefter falde fra senere hen.

Hård data mangler, men på basis af erfaring tror jeg stærkt på, at integrationen af talent i et voksende projekts livstid følger en klassisk logistisk kurve.

[SH] Shawn Hargreaves har skrevet en god analyse af tilpasningen af open-source-metoder til spil; *Playing the Open Source Game**.

[DPV] Note til revisorer: Argumentet om, at service-omkostninger til sidst vil oversvømme en fikseret pris, virker stadig, hvis vi flytter fra konstante dollars til den nuværende nedskrevne værdi, for fremtidig salgsindkomst reduceres i pris parallelt med fremtidige service-omkostninger.

Et lignende, men mere sofistikeret modargument, er observeringen at per-kopi vil serviceomkostningerne gå ned på nul, når køberen slutter med at bruge softwaren; derfor kan du stadig vinde, hvis brugeren stopper, før han/hun har lavet for meget serviceomkostning. Dette er grundlæggende bare et andet type af argumentet, at fabrikspriserne belønner produktionen af hyldewarer. En måske mere instruktiv måde at fremlægge det ville være, at risikoen, som din service koster vil oversvømme stigningen af købeomkostninger med den forventede periode af brugervenlighed af softwaren. Såfabriksmodellen straffer kvalitet.

[WG] Wayne Gramlich <Wayne@Grahmlich.Net> har foreslået, at stædigheden af fabriksmodellen delvist skyldes forældede revisorregler, formuleret da maskiner og bygninger var mere vigtige og folk mindre vigtige. Bøger fra software-firmaer viser computerne, kontormøbler og bygninger som værdifulde ting og programmørerne som udgifter. I realiteten, selvfølgelig, er programmørerne de virkelige værdifulde ting, og computerne, kontor-udstyret og bygninger er næsten ligegyldige. Denne perverse vurdering opretholdes af skattevæsenet og aktiemarkedets pres for stabile og uniforme revisorregler, som reducerer kompleksiteten af at tildele kroner til firmaets værdi. Det resulterende stød har forhindret reglerne for at holde trit med realiteten.

Med dette syn er det at sætte en høj pris til bitsene i produktet (uafhængigt af fremtidig serviceværdi) delvist en slags forsvarsmekanisme, en måde at få alle involverede parter til at enes om, at den ontologiske grund ikke har faldet ned fra de almindelige revisorregler.

(Gramlich på peger også, at disse regler retfærdiggør den bizarre og ofte selvdestruktive købelyst, som mange softwarefirmaer følger efter børsnoteringen. "Sædvanligvis bruger software-firmaet nogle yderligere aktier til at opbygge penge til en kampagne. Men de kan ikke bruge nogen af disse penge til at styrke programmeringsholdet, for revisorreglerne ville vise det som forøget udgifter. I stedet bliver det nye offentlige softwarefirma nødt til at vokse ved at erhverve andre software-firmaer, fordi revisorreglerne lader dig behandle erhvervelsen som en investering.")

For et paradigmatisk eksempel af at dele et projekt efter defektion, såse på historien om OpenSSH. Dette projekt blev delt fra en tidlig version af SSH (Secure Shell) efter denne gik til en lukket licens.

20. Bibliografi

[CatB] Katedralen og bazaaren kan findes på adressen <http://www.laisen.dk/1064.0.html>.

[HtN] Nybyggeri i noosfæren kan findes på adressen <http://www.laisen.dk/1065.0.html>.

[DL] De Marco and Lister, *Peopleware: Productive Projects and Teams* (New York; Dorset House, 1987; ISBN 0-932633-05-6)

21. Tak til

* Se <http://www.talula.demon.co.uk/games.html>.

Adskillige stimulerende diskussioner med David D. Friedman hjalp mig med at finpudse den omvendte fællede-model for open source-samarbejde. Jeg skylder også Marshall van Alstyne tak for at gøre opmærksom på den begrebsmæssige vigtighed af rivaliserende informationsvarer. Ray Ontko fra Indiana Group kom med god kritik. Mange tilhører til mine foredrag i året op til juni 1999 hjalp også; hvis I er en af dem, så ved I, hvem I er.

Det er endnu et vidnesbyrd om open source-modellen, at denne tekst er blevet væsentligt forbedret af den feedback, som jeg fik på e-mail i dagene efter frigivelsen. Lloyd Wood gjorde omærksom på vigtigheden af at open source-software er 'fremtidssikret', og Doug Dante mindede mig om forretningsmodellen 'Befri fremtiden'. Et spørgsmål fra Adam Moorhouse førte til diskussionen om udelukkelsesudbytte. Lionel Oliviera Gresse gav mig et bedre navn til en af forretningsmodellerne. Stephen Trunbull var voldsomt efter mig for skødesløs behandling af medløbereffekter.