Open Source Software, *De Bron Geopend*

Een economische analyse van Open Source Software

Universiteit van Amsterdam (UvA)

Faculteit der Economische Wetenschappen en Econometrie

Afdeling: Accountancy en Informatiemanagement (AIM)

Scriptiebegeleider: Drs. E.J. de Vries

Datum: 24 oktober 2001

Bart S.J. Knubben

Studentnr.: 9579087

Studierichting: BE / Informatiemanagement

Adres: Wipmolen 6

3481 AJ Harmelen

Tel.: 06-24572155

Email: knubben@hetnet.nl

URL: http://home.hccnet.nl/b.knubben

"You use a Windows machine and the golden rule is: Save, and save often. It's scary how people have grown used to the idea that computers are unreliable when it is not the computer at all – it's the operating system that doesn't cut it." –
Linus Torvalds, geestelijk vader van Linux (Harmon, 1998).

"Linux is not in the public domain. Linux is a cancer that attaches itself in an intellectual property sense to everything it touches. That's the way that the license works." –
Steve Ballmer, Microsoft senior executive (Chicago Sun-Times, 2001).

DILBERT (http://www.dilbert.com, april 1999)

[*] *De illustratie op de voorzijde van deze scriptie is gebaseerd op het logo van het Open Source besturingssysteem Linux (de pinguï n "Tux"), dat is ontworpen door Larry Ewing (http://www.isc.tamu.edu/~lewing). De 'Sherlock Holmes'-look is door Rildo Pragana toegevoegd (http://members.tripod.com/rpragana).*

Voorwoord

De bron geopend, een (tweede) studie afgerond! Deze scriptie vormt het laatste wapenfeit van mijn studie Bedrijfseconomie, richting Informatiemanagement aan de Universiteit van Amsterdam. Was de finale van mijn eerdere studie Rechten al een hele bevalling, ook deze afsluiting heeft duidelijk haar sporen achtergelaten in mijn leventje. Echter, het trekken van de 'kar' die de sporen veroorzaakte, was in mijn ogen absoluut de moeite waard. Het onderwerp is onder meer zeer actueel en van maatschappelijk belang. Daarnaast heeft het onderwerp mijn natuurlijke interesse. Kortom, wat wil een mens nog meer?! De relatieve nieuwheid van de materie, zeker in de academische hoek, zag ik eerder als uitdaging dan als hindernis, hoewel het onderzoek daardoor zeker niet makkelijker werd.

Mijn dank gaat uit naar allen die in mijn leven een positieve rol hebben gespeeld. Echter, de volgende acteurs mogen niet ongenoemd blijven: mijn directe familieleden, vader Han, moeder José, Michiel, Ivo en Annemieke. Bedankt voor jullie geduld en ruimte! Daarnaast wil ik bij deze Erik de Vries hartelijk bedanken voor het begeleiden van deze scriptie.

Rest mij nog de lezer aan te moedigen deze pagina om te slaan en ook het vervolg van deze 'opening der bron' in zich op te nemen. "Ga door, de bron zal zich ook voor u openen!"

Bart S.J. Knubben
Harmelen, oktober 2001

Samenvatting

Deze scriptie omvat een beschrijvende, economische analyse van Open Source Software. Onder meer de historie, de economische kenmerken van software, de motivatie van deelnemende programmeurs, de organisatorische inrichting en de ontwikkelingsmethoden zullen aan de orde komen. Uiteindelijk blijken deze achtergronden in hoge mate bepalend voor het karakter van het eindproduct, dat wil zeggen van Open Source Software zelf. Open, gedistribueerde innovatie blijkt zeer krachtig, maar is niet zaligmakend. Op dit moment is Open Source Software op vele fronten goed inzetbaar, maar zijn er ook gebieden waar commerciële software voordelen biedt. Het ligt in de lijn der verwachting dat de kracht van ontwikkeling met open broncode nog verder zal toenemen. Hoewel er zeker plaats blijft voor commerciële software, zal de softwaremarkt door de toenemende kracht van Open Source Software gezonder worden.

INHOUDSOPGAVE

Hoofdstuk 1: Inleiding

"I won't tell you everything about how Open Source works; that would be like trying to explain why English works." – Larry Wall, geestesvader van Perl, 1999.

1.1 Broncode en objectcode, als recept en gerecht

Software kent twee gedaanten, namelijk de object- en de broncode. Computers worden met binaire gegevens, de zogenaamde objectcode, geïnstrueerd over de taken die ze moeten uitvoeren. De binaire gegevens worden gecompileerd uit gegevens die zijn geschreven in een 'hogere orde' computertaal, zoals C, Basic of Java. Deze laatste gegevens, die de zogenaamde broncode vormen, zijn begrijpelijk voor programmeurs (ofwel hackers), terwijl de objectcode voor vrijwel niemand inzichtelijk is.[1] Een veel gemaakte vergelijking ter verduidelijking is dat de broncode staat tot de objectcode als een recept tot het bijbehorende gerecht. Toegang tot de achterliggende broncode is een vereiste om een computerprogramma aan te kunnen passen. Echter, gewoonlijk wordt software tegenwoordig alleen in de vorm van objectcode aan de eindgebruiker geleverd (Wayner, 2000, pp.104-105).

Het omgekeerde proces van compilatie wordt ook wel decompilatie of reverse engineering genoemd. Het is onmogelijk om de oorspronkelijke broncode perfect af te leiden uit de binaire objectcode. De relatie tussen de programmeertaal en de binaire compilatie is namelijk niet lineair, en het compilatieproces is daardoor niet volkomen omkeerbaar. De broncode van een computerprogramma bevat bijvoorbeeld vaak vele non-functionele commentaren van de programmeurs om de broncode meer inzichtelijk te maken. Deze opmerkingen zullen ontbreken bij een uit de binaire code afgeleide broncode waardoor de verkregen broncode moeilijker leesbaar is voor andere programmeurs (Moglen, 1999, p.5).

1.2 Open Source Software, een korte introductie

Open Source Software (OSS)[2] wordt gedefinieerd door de licentie waarmee deze vergezeld gaat (Evers, p.9). Deze licentie stipuleert dat bepaalde rechten van intellectuele eigendom die rusten op de software *niet* door de oorspronkelijke rechthebbende(n) kunnen worden

[1] Hacker wordt hier gebruikt als synoniem voor (talentvol) programmeur. Dit is tevens de oorspronkelijke betekenis van het woord binnen de OSS-gemeenschap. Tegenwoordig wordt het woord voornamelijk gebruikt als synoniem voor computerkraker.

[2] In deze scriptie wordt Open Source Software als kernbegrip gebruikt. Als uitgangspunt voor de betekenis dient de definitie van het Open Source Initiative (OSI), zoals is neergelegd in de Open Source Definition (zie appendix C1). Open Source Software en de afkorting OSS zullen in het vervolg van deze scriptie afwisselend als equivalenten worden gebruikt.

ingeroepen.[3] Deze rechten worden door commerciële softwareproducenten juist ingezet om hun software optimaal te exploiteren. Geheimhouding van de broncode met als extra slot op de deur de verbodsrechten van de intellectuele eigendom vormen voor een groot deel de basis voor de exploitatie van commerciële software. Commerciële software wordt daarom ook wel closed source software ofwel gesloten broncode software genoemd.

De OSS-licentie kent verschillende vormen. Kort gezegd is OSS vrij kopieerbaar en vrij verspreidbaar, en wordt de achterliggende broncode meegeleverd. Iedere capabele gebruiker kan daarom fouten signaleren en verbeteringen aanbrengen in de software. Een belangrijk verschil tussen de twee meest gebruikte licentietypen, de BSD en de GPL[4], heeft betrekking op het hergebruik van de Open Source code in een commercieel, gesloten broncode product. De BSD-licentie staat dergelijk gebruik onder voorwaarden toe, terwijl de GPL dit pertinent verbiedt. Deze laatste verklaart zichzelf bovendien bij eventuele vermenging van softwarecode ook van toepassing op de toegevoegde code. Als overkoepelend begrip is eind jaren negentig Open Source Software officieel gedefinieerd door een nieuw in het leven geroepen instantie, genaamd Open Source Initiative (OSI). Zowel de GPL als de BSD-licentie, maar ook vele andere licenties, worden gedekt door deze definitie.

De complexiteit van software heeft ertoe geleid dat de taken verdeeld dienen te worden. Er is sprake van een coördinatieprobleem. De klassieke oplossing hiervoor is om de activiteiten te coördineren binnen een gecentraliseerde, hiërarchische structuur oftewel een onderneming. De complexiteit wordt onder controle gebracht door formele organisatie en expliciete leidinggevende autoriteit (Weber, 2000, p.4). Eric Raymond, OSS-programmeur en onofficiële etnograaf van de OSS-beweging, vergelijkt deze klassiek oplossing met de kathedraal. Een kathedraal wordt ontworpen via een sterke top-down benadering en een centrale autoriteit stuurt de werkers aan (Raymond, 1999).

Open Source Software wordt echter op een andere wijze ontwikkeld. Raymond vergelijkt het OSS-ontwikkelingsmodel metaforisch met a *"great babbling bazaar of different agendas and approaches"* (Raymond, 1999). Vele duizenden gebruikers over de hele wereld hebben, met name het laatste decennium via het internet, bijdragen geleverd aan Open Source Software

[3] Zowel het auteursrecht als het octrooirecht, beide rechten van intellectuele eigendom (in ruime zin), worden door producenten ingezet bij de exploitatie van software.

projecten. De meeste medewerkers worden niet direct financieel beloond voor hun bijdragen. Bovendien is de organisatie van OSS-projecten vaak zeer informeel. Er is vaak geen formele rechtsvorm; er is geen team om een ontwerp te ontwikkelen; er is geen management dat plant en budgetteert; er is geen personeelsafdeling die de programmeurs aanneemt; en er is geen afdeling die kantoorruimte en computerapparatuur faciliteert (Moon, 2000). Hoewel de organisatievorm meestal niet of nauwelijks geformaliseerd is, bestaat er echter vaak wel een informele structuur. De verschillende taken om een bruikbaar eindproduct te creëren, lijken via andere mechanismen te worden aangestuurd en uitgevoerd.

Sinds enige jaren is er sprake van een groeiende belangstelling voor Open Source Software. De drijfveer hierachter is met name het succes van het besturingssysteem Linux. Hoewel de hoeveelheid 'volwassen' Open Source Software nog beperkt is, wordt deze op bepaalde deelterreinen in toenemende mate ingezet. OSS wordt meer en meer beschouwd als een serieus alternatief voor commerciële software. Op basis van geregistreerde verkopen werd geschat dat Linux op de servermarkt in 1999 een aandeel had van 25 procent en in 2000 van 27 procent (Webwereld, 2001). Naast Linux is de Apache webserversoftware, die wereldwijd inmiddels op meer dan zestig procent van de internetservers wordt gebruikt, een sprekend voorbeeld van de kracht die kan uitgaan van Open Source Software (Netcraft, 2001; Bluescope, 2001). De kracht van OSS heeft verschillende intermediairs - zoals Red Hat, SuSE, Mandrake, (non-profit) Debian, Caldera en TurboLinux - naar de markt doen trekken. Deze distributeurs beogen een marktpositie te verwerven door via softwarebundeling de transactiekosten te verlagen en door aanvullende services te bieden (Wegberg, 2000, p.2). Ook een aantal 'klassieke' softwarebedrijven, waaronder IBM, HP en Sun, spelen sinds enige jaren een actieve rol. Deze bedrijven ondersteunen OSS, dragen actief bij aan verdere ontwikkeling of brengen zelf producten uit in overeenstemming met de OSS-gedachte.

Open Source Software gaat niet uit van de gangbare manier van softwareontwikkeling, -exploitatie en -gebruik. De opkomst van OSS brengt aldus meer met zich mee te brengen dan alleen de broncode die openbaar toegankelijk is. De IT-industrie in het algemeen en de software-industrie in het bijzonder, zijn sedert enige jaren een belangrijke motor van economische welvaart in verschillende landen en regio's. Redmond, het thuis van Microsoft, is hiervan het voorbeeld bij uitstek.

[4] BSD staat voor Berkely Software Distribution, terwijl GPL staat voor General Public License. Zie Appendix C voor verschillende licentie-contracten.

Open Source Software lijkt in toenemende mate een prominent, geheel eigen toontje mee te blazen in deze markt waarin de afgelopen jaren het grote geld regeerde. Daarom is het van belang om de achtergrond van deze ontwikkeling te kennen en de karakteristieken van OSS in kaart te brengen. Op deze wijze kan een plaatsbepaling plaatsvinden van het concept Open Source Software en kan een indruk worden verkregen van toekomstige ontwikkelingen.

1.3 Probleemstelling

De sterke opleving van Open Source Software is een fascinerende ontwikkeling die vele vragen oproept. In deze scriptie zal een beschrijvende analyse vanuit een economische invalshoek met de nadruk op de ontwikkeling en het gebruik van Open Source Software plaatsvinden. Aan de hand van de drijvende krachten achter Open Source Software zal meer inzicht worden verkregen in de sterkten en zwaktes ten aanzien van ontwikkeling en gebruik. Hoewel bepaalde technische aspecten niet zomaar ter zijde kunnen worden geschoven, is de benadering van de problematiek hoofdzakelijk economisch van aard. De aandacht zal evenwel minder gericht zijn op exploitatiemethoden c.q. business models die bijvoorbeeld ingezet worden door eerdergenoemde distributeurs.

In het onderstaande zullen achtereenvolgens de doelstelling, de centrale vraag en de subvragen van dit onderzoek aan de orde komen.

Doelstelling:

Middels literatuuronderzoek inzicht verkrijgen in de achtergronden van Open Source Software op het gebied van ontwikkeling en gebruik, waardoor tot een plaatsbepaling van Open Source Software in de softwaremarkt kan worden gekomen.

Centrale vraag:

Wat zijn karakteristieken van Open Source Software met betrekking tot software-ontwikkeling en -gebruik?

Subvragen:

A. Wat zijn de historie en achtergrond van software in het algemeen en van Open Source Software in het bijzonder?
B. Wat zijn de achtergronden van Linux? Wat zijn de achtergronden van Apache?
C. Welke kenmerken heeft software als economisch goed?

D. Welke redenen van motivatie bewegen OSS-programmeurs?
E. Hoe zijn de OSS-programmeurs georganiseerd en op welke wijze wordt OSS ontwikkeld?
F. Welke voor- en nadelen kent Open Source Software?
G. Zal Open Source Software leiden tot fundamentele veranderingen op de softwaremarkt?

1.4 Methodische verantwoording

Deze scriptie is met name gebaseerd op literatuuronderzoek. Allereerst wordt door veel websites, die nieuws op ICT-gebied brengen, vrijwel dagelijks aandacht besteed aan Open Source Software.[5] Daarnaast zijn er reeds vele artikelen, waarin voorgaande vragen aan de orde komen, op internet verschenen. Deze zijn vaak geschreven door mensen die actief zijn binnen de Open Source beweging. Hun inzichten zijn natuurlijk van enorm belang om een volledig beeld te krijgen van OSS, maar reflecties uit objectievere hoek zijn voor een scriptie als deze natuurlijk ook van grote waarde. Pas de laatste twee jaren lijkt Open Source Software in toenemende mate de aandacht te trekken van juridische, sociale en economische wetenschappers en andere professionele onderzoekers. Hoewel academische literatuur die gewijd is aan Open Source Software daarom in mindere mate voorhanden is, levert intensieve deskresearch toch een aanzienlijke hoeveelheid publicaties op. Naast literatuur die specifiek betrekking heeft op Open Source Software zal economische literatuur worden toegepast op het fenomeen. Met name theorieën over informatie, over moderne organisatievormen en over systeemontwikkeling zullen gebruikt worden.

Er is ruim voldoende interviewinformatie van derden, zoals onder meer van IDC (IDC, 1999), beschikbaar. Bovendien kan veel persoonlijke informatie worden verkregen door de sfeer van openheid waarin Open Source Software ontwikkeling plaatsvindt. Deze open sfeer vertaalt zich onder meer in de openbaar toegankelijke mailing lists. Vanwege de ruimschoots voorhanden zijnde informatie is besloten geen vastomlijnde interviews af te nemen. Wel hebben er verschillende meer open gesprekken plaatsgevonden met mensen die ruime ervaring hebben binnen de IT-branche en met mensen die zelf direct in aanraking zijn gekomen met Open Source Software.[6] Deze gesprekken komen niet direct zichtbaar terug in de scriptie, maar hebben wel een bijdrage geleverd aan de beeldvorming over OSS, zoals weergegeven in deze scriptie.

[5] Zie "Geraadpleegde websites" voor een overzicht.
[6] Zie Appendix E voor een overzicht van deze personen.

1.5 Plan van behandeling

In hoofdstuk 2 zal eerst de historie van software in vogelvlucht worden doorgenomen. De aandacht zal vooral uitgaan naar de rol die open source door de jaren heen heeft gespeeld. In het daaropvolgende hoofdstuk zullen twee recente OSS-projecten, namelijk Linux en Apache, onder de loep worden genomen.
Hoofdstuk 4 bevat een analyse van software als economisch goed. Daarna zal in hoofdstuk 5 de motivatie van OSS-programmeurs nader worden bekeken. Hoofdstuk 6 gaat in op de kenmerken van de organisatie en het ontwikkelingsproces van OSS. In hoofdstuk 7 zal een analyse plaatsvinden van de sterkten en zwaktes van OSS. Uiteindelijk zal deze scriptie haar afronding vinden in het concluderende hoofdstuk 8.

Hoofdstuk 2: Historie en achtergronden van software

"In the end, Open Source is not so much a "phenomenon". This is the way software worked pre-80's. Hiding source seemed to make about as much sense to most people as trying to hide how a lightbulb works. Now, we're coming full circle, and people are cluing in to that. The "Open Source Phenomenon" is just a bunch of people trying to figure out how to make the intervening 20 years of industry make sense...." - Aaron Sherman, Perl Programmeur (Uytterhoeven, 1999)

2.1 De begindagen

De geschiedenis van software gaat terug tot in de jaren '50 en '60 (Blecherman, 1999). In deze begindagen werd software alleen gebruikt op mainframe computers. Slechts enkele organisaties, zoals door de staat gesubsidieerde universitaire informatica-afdelingen (bijv. MIT[7] en UC-Berkeley[8]) en commerciële onderzoeksinstellingen (bijv. Bell Telephone Labs van AT&T en Xerox PARC[9]), beschikten over deze zeer kostbare computers. Computers en software stonden nog maar in de kinderschoenen en werden voornamelijk beschouwd als onderzoeksobjecten. De computerproducent leverde de software meestal gratis bij de computer. Het meeleveren van de broncode werd als volstrekt normaal ervaren. Dit paste natuurlijk uitstekend binnen de academische traditie waarin kennisontwikkeling door kennisdeling een belangrijk plaats inneemt (Wayner, 2000, p.33). Al in deze beginjaren ontstond onder programmeurs een patroon van gemeenschappelijke normen en waarden met betrekking tot informatietechnologie, de zogenaamde *"Hackers Ethic"*. Deze omvatte onder meer de idee dat "*access to computers should be unlimited and total"*. Tot op heden speelt de *"Hackers Ethic"* een rol.

In de toenmalige context was de uitwisseling van de broncode ook uit economisch oogpunt goed verklaarbaar. Software werd vaak specifiek ontwikkeld voor één bepaald type computersysteem. De hardwareproducent verkocht de software in vaste combinatie met een computersysteem. De unieke software was een middel voor de hardwareproducent om zich te onderscheiden. Alle potentiële gebruikers hadden al een kopie van de software in combinatie met de hardware gekocht. Openbaarmaking van de broncode kon aldus niet direct het exploitatiebelang van de producent schaden. Broncode had daarom op zichzelf nauwelijks

[7] MIT: Massachusetts Institute of Technology, http://www.mit.edu
[8] University of California – Berkeley, http://www.berkeley.edu
[9] Xerox Palo Alto Research Center

waarde. Echter, het vrijgeven van de broncode juist wel. Gebruikers konden door de beschikbaarheid van de broncode namelijk verbeteringen aanbrengen aan de software. De verbeteringen werden meestal doorgegeven aan de producent, zodat de gebruiker deze bij een nieuwe uitgave niet opnieuw zelf zou moeten inpassen. Zowel de producent als zijn afnemers profiteerden als de producent deze wijzigingen opnam in zijn nieuwe uitgave van de software (Lerner, 2000, p. 4 e.v.; Weber, 2000, p. 6-7).

2.2 UNIX: Het multi-fundament

Door de deling van de broncode werd ook enorme progressie geboekt op het gebied van portabiliteit. Naarmate computertechnologie wijder verspreid raakte, botste de lage portabiliteit van software in toenemende mate met de academische traditie van kennisontwikkeling door kennisdeling. Eind jaren '60 werden het besturingssyteem "UNIX" en de aanverwante hogere orde programmeertaal "C" door Bell Telephone Labs (BTL) ontwikkeld.[10] BTL was een onderdeel van AT&T, dat op dat moment de grootste computergebruiker van Amerika was.[11] Het idee achter UNIX was om één enkel, schaalbaar besturingssysteem te ontwikkelen dat gebruikt kon worden op uiteenlopende soorten hardware (Moglen, 1999). UNIX werd het eerste besturingssysteem dat platformonafhankelijk was. De programmeertaal C, waarin UNIX vanaf 1972 in geschreven was, maakte dit mogelijk. Tot die tijd werden programma's geschreven in assembleertalen, die moeilijk waren om te leren en bovendien platformafhankelijk waren. Programma's die in C werden geschreven konden relatief eenvoudig overgezet worden naar ieder platform waarop een C-compiler gedraaid kon worden.[12]

Naast platformonafhankelijkheid was *"de schoonheid der eenvoud"* een belangrijke uitgangspunt bij het ontwerp van UNIX. *"Keep It Simple, Stupid"* (KISS) is de onder programmeurs populistische benaming van deze filosofie (Raymond, 1999). Dit komt terug in de sterk modulaire opbouw van UNIX. Naast de bovengenoemde kenmerken beschikte UNIX over zeer geavanceerde technieken zoals "multi-tasking" en "multi-user". UNIX was daarmee haar tijd ver vooruit.

[10] Ken Thompson wordt in de regel beschouwd als de 'uitvinder' van UNIX, terwijl Dennis Ritchie geldt als belangrijkste ontwikkelaar van C. Beiden waren werknemers van Bell Labs. Vandaag de dag is C, samen met de hiervan afgeleide object georiënteerde taal C++, een van de meest gebruikte talen voor zowel applicatie- als systeemprogrammering.
[11] AT&T, http://www.att.com
[12] Een compiler is een programma dat broncode die is geschreven in een programmeertaal vertaalt naar machinetaal, zodat de software kan worden uitgevoerd op een computer.

Door de toegenomen portabiliteit van de broncode kreeg software een waarde op zichzelf. Het logische gevolg was dat software afzonderlijk van de hardware werd verkocht. De geheimhouding van de broncode in combinatie met de intellectueel eigendomsrechtelijke bescherming van software bood bedrijven mogelijkheden software te controleren en naar eigen inzichten te exploiteren (McKusick, 1999).

2.3 Software als commercieel product

Ondertussen groeide UNIX uit tot een populair besturingssysteem. Hoewel formeel niet ondersteund, werd UNIX vrijwel direct na haar geboorte ingezet op verschillende plaatsen binnen AT&T. Door een presentatie over UNIX in 1973 ontstond er ook interesse van derde partijen. Echter, AT&T kon UNIX niet commercieel uitbaten. De uitkomst van een anti-trust rechtszaak uit 1956 bepaalde namelijk kortgezegd dat AT&T alleen actief mocht zijn op het gebied van telefonie (Evers, 2000). Daardoor bood AT&T UNIX aan zonder reclame te maken, tegen kostprijs en zonder ondersteuning.[13]

Door het gebrek aan ondersteuning van de zijde van AT&T organiseerden de gebruikers zich in gebruikersgroepen (oftewel user groups) om UNIX-kennis te delen. Deze groepen communiceerden via het Arpanet, dat kan worden gezien als de voorloper van het huidige internet.[14] Op deze wijze werden talrijke aanpassingen aan UNIX uitgewisseld. Op haar beurt incorporeerde Bell Labs de verbeteringen in opvolgende versies van UNIX. In die jaren was UC-Berkeley één van de meest actieve deelnemers op het gebied van UNIX-onderzoek.[15] Parallel aan de versies van AT&T ontwikkelde zij eigen applicaties en zelfs eigen versies van UNIX. Deze werden in 1977 voor het eerst onder aanvoering van William Joy uitgegeven onder de naam Berkeley Software Distribution (BSD). De bijbehorende BSD-licentie, die in vrijwel gelijke vorm nog steeds wordt gebruikt, was zeer vrij van aard. De BSD-software mocht bijvoorbeeld ook zonder meelevering van de broncode door commerciële bedrijven worden aangepast en geëxploiteerd. Echter, de naam van de universiteit als medeontwikkelaar diende dan wel duidelijk vermeld te worden (McKusick, 1999).

[13] In het Engels ook wel "as is" genoemd.
[14] Zie paragraaf 2.4.
[15] University of California – Berkeley, de Computer Science Research Group (CSRG).

Aan het begin van de jaren '80 introduceerde IBM de Personal Computer (PC). Dit had tot gevolg dat de computer evolueerde tot een breder geaccepteerd gebruikersgoed. De doorsnee consument was niet zozeer geïnteresseerd in broncode, maar wel in gebruiksvriendelijke, goede software (Wayner, 2000, p. 33). De trend van commerciële exploitatie van software werd hierdoor enorm versterkt. Microsoft (opgericht in juli 1975) is zonder enige twijfel de meest bekende adept van de veranderde exploitatiewijze. Het bedrijf veroverde binnen korte tijd een machtspositie op de markt voor PC-besturingssyteemsoftware, mede doordat IBM zich voornamelijk focuste op hardware en AT&T zich met UNIX meer op de grotere systemen richtte. Bovendien gebruikte Microsoft haar sterke positie op de markt voor PC-besturingssystemen gedurende de laatste jaren om in andere softwaremarkten, en met name die voor PC-applicaties, ook zeer sterk terrein te winnen.

De stormachtige ontwikkelingen op het gebied van commerciële exploitatie van software gingen niet voorbij aan AT&T. Vanaf begin jaren '80 begon AT&T in toenemende mate haar intellectuele eigendomsrechten op UNIX actief te gebruiken, en bovendien werd een geheel ander ontwikkelteam verantwoordelijk voor UNIX. Hierdoor was Bell Labs niet langer meer in staat om te fungeren als een "clearing-house" voor UNIX-onderzoek. Het vacuüm dat ontstond werd echter al snel grotendeels opgevuld door UC-Berkeley. In 1984 werd AT&T op basis van het Amerikaanse mededingingsrecht opgedeeld. Dit bracht tevens met zich mee dat het bedrijf niet meer gebonden was aan de eerdergenoemde gerechtelijke marktafbakening. Geheel in lijn met de toenmalige commercialiseringstrend bracht AT&T het besturingssyteem UNIX onder een commerciële licentie op de markt.

Gedurende de jaren hierna werden vele UNIX-varianten ontwikkeld. Sommige waren gebaseerd op de commerciële versies van AT&T, andere waren geënt op de meer wetenschappelijk gerichte BSD-edities. Iedere versie bevatte delen broncode waarop AT&T auteursrecht claimde. Voor iedere kopie van UNIX, ongeacht de versie, moest daarom een alsmaar duurder wordende licentie van AT&T worden aangeschaft. Inspanningen van meer dan 400 programmeurs, onder aanvoering van Keith Bostic, zorgden ervoor dat de BSD-versie in 1991 vrijwel volledig vrij was van AT&T-broncode. Niet lang daarna maakte Bill Jolitz UNIX passend voor de Personal Computer. In zijn 386BSD-versie voegde hij ook de tot dan toe 6 ontbrekende files toe, zodat BSD-broncode geheel op eigen benen leek te staan.[16]

[16] 386BSD werd speciaal ontwikkeld voor de Intel i386 chip.

De BSD-tak versplinterde echter in 1993 toen NetBSD en FreeBSD ieder een eigen weg gingen.[17] Beide groepen zijn tot op heden actief met de ontwikkeling van de BSD Unix-variant, waarbij de openbaarheid van de broncode als uitgangspunt dient.

Pas nadat een andere splintergroep "Berkeley Software Design incorporated" (BSDi) een commerciële BSD-versie, die vrij was van AT&T-code en dus ook geen dure AT&T-licentie vereiste, op de markt had gebracht, kwam AT&T in actie. Het bedrijf begon een rechtszaak tegen zowel BSDi als tegen de UC-Berkeley. Pas in januari 1994, nadat UNIX was overgenomen door Novell, kwam met een schikking een einde aan de rechtszaak. De exacte inhoud van de schikking is geheim. Op de BSD-broncode rustten vanaf dat moment officieel geen rechten meer van Novell / AT&T. Echter, door de rechtszaak was de ontwikkeling een aantal jaren vrijwel stil blijven staan. Bovendien ontstond er in 1995 nog een breuk. Een groep, onder leiding van Theo de Raadt, splitste zich af van het NetBSD-project en startte met OpenBSD.

Ondertussen waren er tientallen verschillende afstammelingen van het originele UNIX ontstaan.[18] Overdraagbaarheid en openheid, waarden die nu juist kenmerkend waren voor UNIX, werden hierdoor bedreigd. De vertraging door de rechtszaak en de verschillende afsplitsingen werkten een bredere acceptatie van UNIX tegen. Bovendien vertoonde de academische wereld steeds minder interesse voor software in het algemeen en besturingssystemen in het bijzonder. Het onderzoeksobject "software" verloor aan aantrekkingskracht door de vercommercialisering en haar gevolgen, en er bestond het gevoel dat het onderzoeksterrein reeds in grote mate was ontgonnen (Newman, 1999). Daarnaast was de (Amerikaanse) overheid gedurende de jaren '80 al in toenemende mate overgegaan tot een *'laissez faire'* beleid ten aanzien van de softwaremarkt.

Op dit moment wordt de naam UNIX beheerd door de koepelorganisatie "The Open Group".[19] Alleen als een besturingssysteem voldoet aan de door haar gestelde voorwaarden dan mag de naam UNIX voor de software worden gebruikt. Er bestaan echter veel besturingssystemen die nauw verwant zijn aan UNIX. Vrijwel iedere grote hardwareproducent geeft een eigen versie van UNIX uit. Nog steeds wordt UNIX geprezen

[17] Een splintergroep wordt in programmeursjargon ook wel een "fork" genoemd.
[18] Als meer prominente afstammelingen worden hier genoemd: IBM AIX, BSDI, FreeBSD, HP-UX, IRIX, NetBSD, OpenBSD, Pyramid, SCO UNIXware, SCO Open Server, Sun Solaris, SunOS, Compaq Tru64 UNIX.

om haar unieke eigenschappen. De afgelopen jaren werd UNIX met name ingezet als besturingssyteem voor zwaardere serversystemen.

2.4 Van ARPAnet naar Internet

Onder leiding van het aan het Amerikaanse Ministerie van Defensie gelieerde ARPA kwam in 1969 het eerste netwerk, genaamd ARPAnet, tot stand.[20] Grotendeels gesubsidieerd door de Amerikaanse overheid werd een netwerk met vier computers ofwel nodes gecreëerd. In 1972 werd het netwerk dat toen al 29 computers omvatte aan het publiek gepresenteerd. Hoewel het ARPAnet aanvankelijk officieel werd ontwikkeld voor militaristische doeleinden, was het al voor haar geboorte een onderzoeksobject voor universiteiten (Newman, 1999). De initiators achter het project voorzagen reeds in 1968 dat online-communities onder meer het programmeren van computers radicaal zouden beï nvloeden. Hoewel zij veiligheid en privacy als problemen onderkenden, gingen zij ervan uit dat software binnen een bepaald beleid van toegangscontrole vrij gebruikt en verspreid kon worden (Tuomi, 2001).

Het ARPAnet zou de basis vormen voor het internet dat in 1991 zijn levenslicht zag. Net zoals de PC eerder een revolutie betekende qua informatieverwerking, zorgde het internet voor een wereldwijde revolutie op het gebied van informatie-uitwisseling. Doordat programmeurs slechts informatie hoeven uit te wisselen, verlaagde het internet de transactiekosten van OSS-projecten enorm. Hierdoor nam met de groei en volwassenwording van het internet ook de intensiviteit van communicatie tussen programmeurs toe.

Het internet was evenals haar voorganger gebaseerd op open standaarden, zoals TCP/IP, BIND en HTML. Gedurende de jaren '90 trok de (Amerikaanse) overheid zich meer en meer terug als voorvechter van open standaarden. Het internet werd daardoor grotendeels vercommercialiseerd en geprivatiseerd. Dit opende mogelijkheden voor bedrijven om hun eigen commerciële gesloten producten tot standaard te maken. Netscape was een van de eerste bedrijven die hier aanvankelijk optimaal van profiteerde. Echter, via een offensief enkele jaren later is Microsoft nu nagenoeg een alleenheerser op de browsermarkt. Hoewel veel van de gebruikte standaarden nu nog open zijn, doen verschillende bedrijven pogingen om commerciële standaarden te creëren. Microsoft probeert bijvoorbeeld op verschillende terreinen om toonzetter en of eigenaar te worden van veel gebruikte standaarden (Newman,

[19] http://www.unix-systems.org
[20] ARPAnet staat voor Advanced Research Projects Agency Network.

1999). Maar ook andere bedrijven, zoals Macromedia en Sun, doen in mindere of meerdere mate dergelijke pogingen.[21]

Het World Wide Web, een zeer belangrijke toepassing van het internet, kan overigens qua techniek ook als grotendeels Open Source worden aangemerkt, daar de broncode van een webpagina (in met name HTML) vaak ook voor een ieder toegankelijk is. Hoewel zeker niet iedereen dit waardeert of zich hiervan bewust is, lijkt dit toch een van de stuwkrachten achter het succes van het internet te zijn geweest.

2.5 Free Software Foundation

In 1971 trad Richard Stallman als onderzoeker in dienst bij het MIT Artificial Intelligence Lab. In deze jaren werd door MIT het eigen ITS besturingssysteem gebruikt. Echter, vanaf begin jaren tachtig deed commerciële software haar intrede. Gebruikers werden hierdoor ernstig gelimiteerd in de mogelijkheid tot aanpassen (waaronder verbeteren) van de software. Stallman verwierp deze ontwikkeling op morele gronden. Als een tegenreactie op deze ontwikkeling richtte hij begin jaren '80 de Free Software Foundation (FSF) op.[22] In het kader van het GNU-project stelde de FSF zich ten doel een integraal pakket aan Free Software, die UNIX compatibel was, aan te bieden. Om vrijheid van de ontwikkelde software ook daadwerkelijk te garanderen werd de General Public License (GPL), ook wel copy*left* genoemd, ontwikkeld. Het centrale element van de licentie is dat Free Software op geen enkele wijze mag worden gemonopoliseerd. Met het woord "free" wordt gedoeld op vrijheid en niet zozeer op prijs. Voor de software kan wel of niet een prijs worden gevraagd. Echter, het programma mag altijd worden gebruikt, gekopieerd, aangepast en verspreid. De software (, ook aanpassingen daarvan of varianten daarop,) dient altijd vergezeld te gaan van de GPL, die de genoemde rechten waarborgt.

Veel belangrijke programmatuur werd binnen het zogenaamde GNU-project[23] ontwikkeld, maar de kernbijdrage kwam onverwacht in het begin van de jaren '90 toen de Finse student Linus Torvalds het inmiddels roemruchte besturingsysteem "Linux" ontwikkelde.

[21] Macromedia bijvoorbeeld met Flash en Sun met Java.
[22] http://www.fsf.org
[23] GNU is een recursieve acroniem en staat voor "GNU is Not UNIX". In het GNU-kader is software ontwikkeld zoals de Emacs text editor, de GCC compiler en de GDB debugger.

2.6 Linux

Ten tijde van de juridische gevechten tussen het BSD-team en AT&T begon Linus Torvalds, een 21 jarige student aan de Universiteit van Helsinki, in 1991 met de ontwikkeling van een eigen besturingssysteem. Torvalds wilde thuis oefenen met een volwaardig besturingssysteem op zijn eigen PC met Intel 80386-processor.

Een commerciële UNIX-variant was vanwege de enorm hoge prijshoogte geen optie (Wayner, 2000, p.133). Bovendien was Torvalds niet op de hoogte van de ontwikkelingen rond BSD-UNIX aan UC-Berkeley. Torvalds heeft meerdere malen zijn twijfel uitgesproken of hij wel met de ontwikkeling van zijn eigen besturingssysteem zou zijn begonnen, als hij van BSD op de hoogte was geweest. (Wayner, 2000, p.54).

Tijdens zijn studie was Torvalds reeds in aanraking gekomen met het PC-compatibele Minix. Dit besturingssysteem, dat veel gelijkenis vertoonde met UNIX, was ontworpen voor onderwijsdoeleinden door Andrew Tanenbaum.[24] Hoewel Tanenbaum ook de broncode bijvoegde, moest men betalen voor een licentie van Minix. Bovendien was verdere verspreiding verboden. Omdat Minix was ontworpen voor onderwijsdoeleinden bestond het slechts uit basiselementen. Ondanks veler verzoek was Tanenbaum daarom ook niet bereid Minix uit te breiden.

Het toenmalig populaire Microsoft DOS en de grafische schil Windows 3.1 achtte Torvalds om verschillende redenen niet geschikt. Ten eerste hield Microsoft de broncode strikt geheim en schermde het bedrijf actief met haar intellectuele eigendomsrechten. Daarnaast had DOS als besturingssysteem ernstige beperkingen. DOS was bijvoorbeeld onder meer geen multi-tasking en multi-user besturingssysteem (Kuwabara, 2000).

Torvalds ontwikkelde zijn besturingssysteem, dat hij Linux doopte, niet helemaal vanuit het niets.[25] Aanvankelijk gebruikte hij enkele ideeën van Minix, en daarnaast maakte hij dankbaar

[24] Andrew Tanenbaum is professor aan de Vrije Universiteit te Amsterdam, zie http://www.cs.vu.nl/~ast/

[25] Torvalds wilde zijn besturingssyteem aanvankelijk "Freax" noemen. Dit zou een samenvoeging zijn van Free, Freak en UNIX. Echter, op aandrang van Ari Lemke, die hem ruimte bood op de ftp-server van de Universiteit van Helsinki, koos Torvalds voor de naam Linux (Kuwabara, 2000). Torvalds heeft de naam Linux in verschillende landen gedeponeerd als merk. Larry Ewing ontwierp in 1996 een pinguï n als het logo van Linux. De pinguï n werd Tux genoemd. Tux staat voor "Torvalds UNIX". Tux betekent overigens ook smoking (Dellio, 2001).

gebruik van de vrije GNU-software, waaronder met name de compiler GCC, die ontwikkeld was door de FSF.[26] De eerste versies verschenen onder een licentie die zelfs commercieel gebruik verbood. Na enige tijd verklaarde Torvalds de GPL ook van toepassing op Linux (Wayner, 2000, p.62).

Op 5 oktober 1991 lanceerde Torvalds versie 0.02 van Linux via het internet. In zijn aankondiging nodigde Torvalds andere programmeurs uit om bijdragen aan de broncode te leveren; *"a program for hackers by a hacker"*. Al snel reageerden verschillende programmeurs van over de hele wereld. De eerste volwaardige, officiële versie (v.1.0) van Linux presenteerde Torvalds in 1994. In 1996 werd versie 2.0 gepubliceerd. Momenteel is versie 2.4.4 de meest recente stabiele Linux-kernel.

De Linux-kernel vormde in combinatie met de GNU-tools van de FSF een volwaardig besturingssysteem. Linux is in feite alleen een zogenaamde kernel. De kernel vormt het hart van het besturingssysteem. Deze draagt op het meest fundamentele niveau zorg voor de communicatie tussen de hardware, software en de gebruiker. De broncode van de kernel beslaat in een gangbare Linux-distributie nog geen 5% van de totale broncode. Daarom worden distributies van het besturingssyteem ook wel GNU/Linux genoemd.

2.7 Open Source Initiative en licenties

In 1998 richtten Eric Raymond en Bruce Perens, twee gevestigde namen in de Open Source wereld, het Open Source Initiative (OSI) op.[27] Het OSI heeft een pragmatische benadering van ontwikkeling van software. Dit gezichtspunt verschilt van de overwegend idealistische benadering van de FSF en haar voortrekker Stallman, die commerciële 'gesloten broncode' software verwerpt. Volgens het OSI levert open source ontwikkeling betere software op dan de traditionele softwareontwikkeling. Haar doelstelling is om OSS-ontwikkelaars en het bedrijfsleven nader tot elkaar te brengen, opdat het draagvlak voor Open Source Software toeneemt.

De organisatie heeft een eigen definitie van Open Source Software ontwikkeld, de Open Source Defenition (OSD). Als basis voor de definitie diende een beleidsdocument van de non-profit softwaredistributeur Debian. De OSD is beoogd te fungeren als een overkoepelend

[26] GCC = GNU C Compiler
[27] http://www.opensource.org

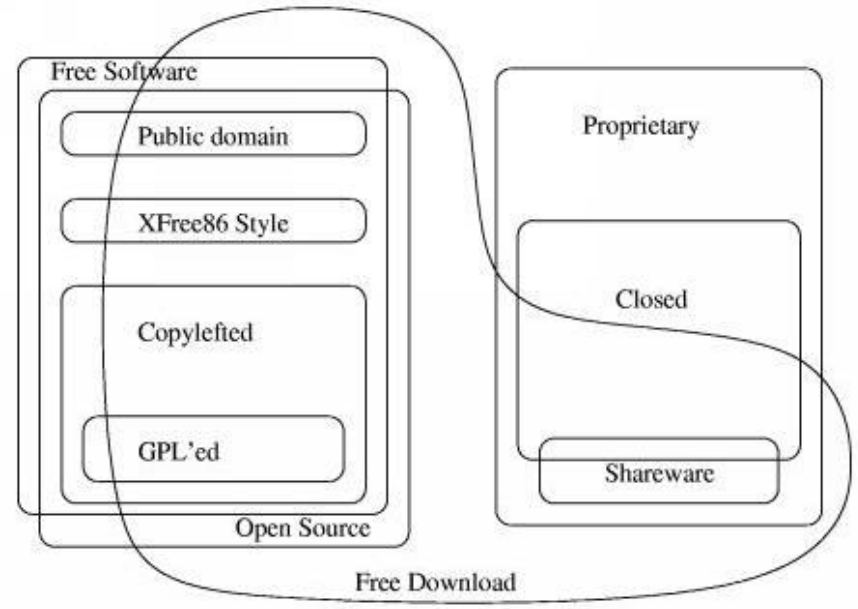
begrip. Als een licentie voldoet aan de OSD, dan mag de bijbehorende software als OSI gecertificeerd worden aangeprezen. [28] Om OSI geen lege huls te laten doen worden, is de naam als merknaam gedeponeerd. Het is belangrijk te realiseren dat in de OSI gecertificeerde licenties uitdrukkelijk geen afstand wordt gedaan van het auteursrecht, maar dit juist wordt ingezet om bepaalde vrijheden te garanderen.

In tegenstelling tot de General Public License (GPL) vereist de OSD geen viraal effect. Dit laatste wil zeggen dat de GPL ook van toepassing is op software die direct gekoppeld wordt aan software die vergezeld gaat van de GPL (of te wel Free Software). De GPL correspondeert echter wel met de bepalingen van de OSD. Ook andere licenties, zoals de BSD-licentie, voldoen aan de vereisten. In de commerciële en de open source werelden worden veel verschillende licenties gebruikt. In onderstaande figuur wordt een aantal veelgebruikte licenties uiteengezet op basis van rechten en plichten voor de gebruiker. De daaropvolgende figuur geeft een visuele weergave van de verschillen tussen de licentietypen.

Licentie-voorwaarden / **Soort licentie**		**gratis verkrijgbaar**	**vrij kopieerbaar en verspreidbaar**	**onbegrensd bruikbaar (usus)**	**broncode meegeleverd**	**aanpassing broncode toegestaan (abusus)**	**aanpassingen moeten ook vrijgegeven worden onder dezelfde licentie**	**geen vervlechting met commerciële software toegestaan**
Commerciële software								
Shareware		(X)	X					
Freeware		X	X	X				
Royalty free libraries		X	X	X	X			
Publiek domein (niet auteursrechtelijk beschermd)		X	X	X	X	X		
Open Source	XFree86, BSD, MPL, Apache	X	X	X	X	X		
	LGPL	X	X	X	X	X	X	
	GPL (Copyleft/Linux)	X	X	X	X	X	X	X

[28] De volgende licenties zijn OSI gecertificeerd: GNU General Public License (GPL), GNU Library or "Lesser" Public License (LGPL), BSD license, MIT license, Artistic license, Mozilla Public License v. 1.0 (MPL), Qt Public License (QPL), IBM Public License, MITRE Collaborative Virtual Workspace License (CVW License), Ricoh Source Code Public License, Python license, zlib/libpng license, Apache Software License, Vovida Software License v. 1.0, Sun Internet Standards Source License (SISSL), Intel Open Source License, Mozilla Public License 1.1 (MPL 1.1), Jabber Open Source License, Nokia Open Source License, Sleepycat License, Nethack General Public License, Common Public License, Apple Public Source License (20 september 2001).

Figuur 1: Overzicht van de verschillende typen licenties (Bundesministerium für Wirtschaft und Technologie, 2000, p.40).



Figuur 2: Verschillende software-categorieen (FSF, 1998a).

2.8 Huidige OSS-projecten en het bedrijfsleven

Om de opleving van OSS nader te illustreren, zal de rol van het bedrijfsleven worden toegelicht en zal een overzicht van een aantal OSS-projecten worden gepresenteerd. Er bestaat een breed spectrum aan Open Source Software. Naast FreeBSD, Linux en GNU zijn er vele duizenden andere initiatieven. Het is daarom onmogelijk om alle projecten te noemen. Echter, in het onderstaand overzicht zullen een aantal aansprekende projecten aan de orde komen ter illustratie van de huidige diversiteit aan OSS. Voor een meer volledig overzicht zij verwezen naar OSS-fora, zoals Freshmeat.[29]

Onder de initiatieven bevinden zich ook enkele softwareprojecten die van origine niet Open Source waren. Enkele bedrijven, zoals Netscape en Sun, hebben namelijk onlangs de broncode van oorspronkelijk commerciële, gesloten softwareproducten vrijgegeven. Bovendien verlenen verschillende bedrijven uit de computerbranche actief steun aan OSS-projecten. In 2000 richtten bijvoorbeeld een aantal vooraanstaande computerbedrijven - waaronder IBM, Hewlett-Packard, NEC en Intel – het zogenaamde Open Source Development Lab op.[30] In een pand in Oregon plaatsten deze bedrijven diverse grote servers

[29] http://www.freshmeat.net
[30] http://www.osdl.org

met als doel OSS-projecten de mogelijkheid te bieden de software te testen op zwaardere systemen. Onlangs maakte IBM kenbaar dat het bedrijf gedurende 2001 één miljard dollar zou investeren in de verdere ontwikkeling van Linux (Wilcox, 2000b).

Apple heeft haar nieuwe besturingssyteem OSX grotendeels gebaseerd op FreeBSD. Het bedrijf heeft de broncode van de kern van OSX vrijgegeven, ondanks dat het bedrijf hiertoe niet verplicht is op grond van de BSD-licentie die commercieel gebruik toestaat.[31] Echter, de broncode van de softwareschillen om die kern, die OSX gebruiksvriendelijk maken, heeft Apple niet vrijgegeven (Rispens, 2001). Indien FreeBSD vergezeld zou gaan van de GPL (zoals Linux), dan had Apple wel de broncode van directe aanpassingen en toevoegingen aan FreeBSD moeten publiceren.

Ook Microsoft heeft zich intussen bij verschillende gelegenheden uitgelaten over Open Source Software. Hoewel het bedrijf zijn eigen producten niet Open Source uitgeeft, maakt het zelf wel gebruik van bepaalde OSS. Uit uitgelekte interne memo's blijkt dat Microsoft Open Source Software zeer serieus neemt.[32] Daarnaast hebben verschillende Microsoft topmanagers zich onlangs uiterst kritisch en enigszins ongenuanceerd uitgelaten ten aanzien van Open Source Software (Mundie, 2001).

[31] http://www.opensource.apple.com
[32] Beter bekend als de Halloween-documenten. De eerste twee documenten werden in 1998 geschreven door Microsoft medewerker V. Valoppilil, zie http://www.opensource.org/halloween/links.html.

	OSS-project	Omschrijving	gebaseerd op:	technische specificatie / standaarden	Type licentie	Jaar van ontstaan (first release)	Initiatiefnemer (toenmalige gegevens)	Aantal programmeurs / organisatievorm	Eindgebruikers / Gebruikscijfers	Ondersteunende bedrijven	Concurrenten
Systeem Infrastructuur Software	**GCC** (GNU Compiler Collection) http://www.fsf.org/software/gcc/gcc.html	Compiler		onder meer voor C, C++, Fortran, Objective C / ondersteunt meer dan 40 chips	GPL	1987	Richard Stalmann	steering committee (14 leden)	Microsoft, Nortel, Intel, Ericsson, Ciso Systems, Alcatel, Apache, Linux	RedHat (Cygnus)	
	Linux http://www.kernel.org http://www.linux.org http://nl.linux.org	besturingssysteem	Minix / UNIX	Posix, Single UNIX Specification, hoge portabiliteit: ondersteunt minstens 14 processoren	GPL	1991	Linus Torvalds (student, 21 jaar, Finland)	inforrmele, gelaagde, open community	Shell, Telia, Sony, veel ISP's, Cisco, Siemens, Mercede-Benz, Boeing, Ikea, Nasa, Volker / ±17-20 miljoen	IBM, HP, Dell, Red Hat, SuSE, VA Linux, Nortel, NEC, Adobe, Compaq, Oracle, Nokia, SGI	MS-Windows, Novell Netware, UNIX, FreeBSD
	FreeBSD http://www.freebsd.org	besturingssyteem	UNIX, BSD	POSIX	BSD	1993	UC-Berkeley	kernteam van ± 9 programmeurs, 270 "committer" developers en +5,000 developers	Microsoft, Yahoo !	Walnut Creek, Apple	MS-Windows, Novell Netware, UNIX, Linux
	Apache http://www.apache.org	webserver software (http)	NCSA	HTTP	Apache (BSD-stijl)	1995	Brian Behlendorf (21 jaar) en anderen	kerngroep van 38 leden, 400 developers	+60% marktaandeel	IBM, Apple	Microsoft IIS, Netscape
	Sendmail http://www.sendmail.org	mail-server			BSD / Sendmail License	1981	Eric Allman (graduate student)		3Com / ± 75%	Sendmail inc., IBM	MS-Exchange, IBM Lotus Domino

	OSS-project	Omschrijving	gebaseerd op:	technische specificatie / standaarden	Type licentie	Jaar van ontstaan (first release)	Initiatiefnemer (toenmalige gegevens)	Aantal programmeurs / organisatievorm	Eindgebruikers / Gebruikscijfers	Ondersteunende bedrijven	Concurrenten
	Samba http://www.samba.org	file en print server software voor Microsoft Windows network clients		SMB	GPL	1992	Andrew Tridgell, 25 jaar, Australia	kernteam van 20 programmeurs		HP, SGI	MS-Windows server
	BIND (Berkeley Internet Name Domain) http://www.isc.org/products/BIND	(de-facto standaard) DNS-server voor internet		DNS	Vrij gebruik maar wel restricties	jaren '70	graduate student project aan UC-Berkeley ondersteund door DARPA	Nominum, ISC, Paul Vixie / ontwikkeling in relatieve geslotenheid			
	MySQL http://www.mysql.com	SQL database server		SQL (Structured Query Language)	GPL	1994	David Axmark, Allan Larsson and Michael "Monty" Widenius, resp. 2 Zweden en 1 Fin	team van 15 programmeur in dienst bij MySQL AB	Yahoo! Finance, MP3.com, Motorola, NASA, Silicon Graphics, and Texas Instruments / + 2 miljoen		MS-SQL server, Oracle
Tools	**Perl** (Practical Extraction and Reporting Language) http://www.perl.com http://www.perl.org	Tool, programmeertaal			Artistic license, GPL	1987	Larry Wall, programmeur bij Burroughs (later Unisys), project voor NSA	Kernteam van 6 leden	geschat 1 miljoen gebruikers	ActiveState, O'reilly	TCL/TK, Python
	TCL/Tk http://dev.scriptics.com/software/tcltk	Tool, programmeertaal			BSD-achtig	1991	John Ousterhout		AOL**, Cisco /** ongeveer een half miljoen gebruikers	ActiveState	Perl, Python, Java

	OSS-project	Omschrijving	gebaseerd op:	technische specificatie / standaarden	Type licentie	Jaar van ontstaan (first release)	Initiatiefnemer (toenmalige gegevens)	Aantal programmeurs / organisatievorm	Eindgebruikers / Gebruikscijfers	Ondersteunende bedrijven	Concurrenten
	Python http://www.python.org	Tool, programmeertaal		geschikt voor UNIX, Windows, DOS, OS/2, Mac	Python licentie (BSD-achtig); "absolutely free, even for commercial use (including resale)"	1990	Guido van Rossum, CWI, Amsterdam	Python Software Activity, BeOpen.com		ActiveState	Perl, Tcl/Tk, Java
Applicaties	**OpenOffice** http://www.openoffice.org	Office-pakket; broncode in 2000 vrijgegeven door Sun			SISSL / LGPL	jaren '80 / 2000 (geopend)			Pentagon	Sun (StarOffice)	MS-Office
	Abisource http://www.abisource.org	tekstverwerking, gepland is volwaardig office-pakket		voor Windows, UNIX, GNOME, BeOS and QNX	GPL	1999				SourceGear	MS-Office
	Gimp (GNU Image Manipulation Program) http://www.gimp.org	2D-grafische software			GPL	1996	Spencer Kimball & Peter Mattis (informatica-student en aan UC-Berkely)			Red Hat, Rhytm & Hues, Silicon Grail, TUX is ontworpen met The Gimp door Lawrence Lessig	Photoshop, Paint etc.

OSS-project	Omschrijving	gebaseerd op:	technische specificatie / standaarden	Type licentie	Jaar van ontstaan (first release)	Initiatiefnemer (toenmalige gegevens)	Aantal programmeurs / organisatievorm	Eindgebruikers / Gebruikscijfers	Ondersteunende bedrijven	Concurrenten
Mozilla http://www.mozilla.org	Webbrowser; broncode vrijgegeven door Netscape			MPL / NPL	1998 (geopend)	Mozilla			Netscape	MS-Explorer, Opera, Konqueror (van KDE)
LaTex http://www.latex-project.org http://www.ntg.nl	tekstopmaak programma (populair onder exacte wetenschappers)		Tex door Donald Knuth	LaTeX Project Public License (LPPL)	1985	Leslie Lamport	LaTeX3 Project			Adobe Acrobat
KDE (K Desktop Environment) http://www.kde.org http://www.koffice.org	grafische desktop-omgeving		bedoeld voor Unix	GPL	1996					MS-Office
GNOME (GNU's Network Object Model Environment) http://www.gnome.org	grafische desktop-omgeving		maakt gebruik van GTK+, die werd ontwikkeld door GIMP / bedoeld voor Unix	GPL	1997	Miguel de Icaza en Frederico Mena-Quintero	GNOME Foundation			MS-Office
XFree86 http://www.xfree86.org	X-server; software tussen hardware en GUI, zoals GNOME en KDE			X11	1984	Jim Gettys en anderen aan MIT	kernteam van 11, wereldwijd 600 ontwikkelaars, ontwikkeling in relatieve geslotenheid			

Figuur 3: Overzicht van een aantal OSS-projecten en hun karakteristieken. Aangezien niet alle informatie voorhanden is, ontbreekt deze op bepaalde plaatsen.

2.9 Afronding

De bovengeschetste geschiedenis van software laat zien dat uitwisseling van de broncode zeker geen nieuw fenomeen is. Openheid van de broncode is voortdurend van wezenlijke betekenis geweest voor de snelle ontwikkeling die software heeft doorgemaakt. Met name in de beginjaren speelde de uitwisseling van broncode een zeer belangrijke rol. Software was aan universiteiten een aansprekend onderzoeksobject en de (Amerikaanse) overheid achtte software van groot politiek belang. Op deze wijze werd een fundament gecreëerd waarvan tot op heden de vruchten worden geplukt.

De techniek van het internet is zonder enige twijfel het meest aansprekende voorbeeld van ontwikkeling met open broncode. Het internet zelf vormde vervolgens een vruchtbare ondergrond voor de huidige opkomst van de Open Source Software. De krachten van vele eenlingen konden via dit medium samen worden gebracht. Dit gebeurde al heel vroeg via de eerste op UNIX gerichte gebruikersgroepen. Echter, aan de recente opleving van Open Source Software ging de opkomst van de vercommercialisering van de softwaremarkt vooraf. Via intellectueel eigendomsrecht, technische beschermingen en geheimhouding van de broncode pogen bedrijven tot op heden software rendabel te exploiteren. Door deze trend werd de ontwikkeling en verspreiding van Open Source Software, zoals de BSD-varianten van UNIX, aanvankelijk gefrustreerd. Commerciële software werd van uitzondering tot regel.

Echter, sinds enige jaren zijn er talloze initiatieven op het gebied van Open Source Software. Een aantal hiervan kan zelfs de concurrentie aan met zijn commerciële tegenhangers. Bovendien wordt Open Source Software in toenemende mate gesteund door bedrijven, waaronder een aantal gevestigde namen in de ICT-branche. De achtergronden van twee aansprekende OSS-projecten, die beide ook op steun uit de commerciële hoek kunnen rekenen, zullen in het volgende hoofdstuk nader worden toegelicht.

Hoofdstuk 3: Twee OSS-projecten nader bekeken: Linux en Apache

"[Linux] started as a program for my own use. I was blown away by how many people there were with similar needs." – Linus Torvalds, (Kuwabara, 2000).

3.1 Inleiding

In dit hoofdstuk zullen twee succesvolle OSS-projecten nader worden bekeken. Onder meer de achtergronden, de groei, de organisatie en de ontwikkelingsmethoden zullen de aandacht krijgen.

3.2 Linux

De eerste versie van het Linux-besturingssysteem werd in 1991 ontwikkeld door Linus Torvalds. Torvalds bracht de kernel uit onder de GPL licentie (aanvankelijk onder een vergelijkbare licentie), en nodigde via de nieuwsgroep (comp.os.minix) andere programmeurs uit om bijdragen aan het programma te leveren.[33] De respons op Torvalds uitnodiging was overweldigend. Aan het eind van 1991 hadden bijna honderd mensen zich aangemeld bij de Linux nieuwsgroep en voor het eind van het jaar gaven ongeveer dertig mensen bijna 200 concrete foutmeldingen en foutoplossingen (patches) bijdragen (Weber, p.9). Bij de aankondiging van versie 0.11 in december 1991, noemde Torvalds naast zichzelf drie andere programmeurs. Versie 0.13, uitgegeven in februari 1992, bevatte al meer patches van andere programmeurs dan van Torvalds zelf (Moon, 2000). In juli 1995 hadden meer dan 15,000 mensen uit 90 landen een bijdrage geleverd in de vorm van opmerkingen, bug reports, patches en features.

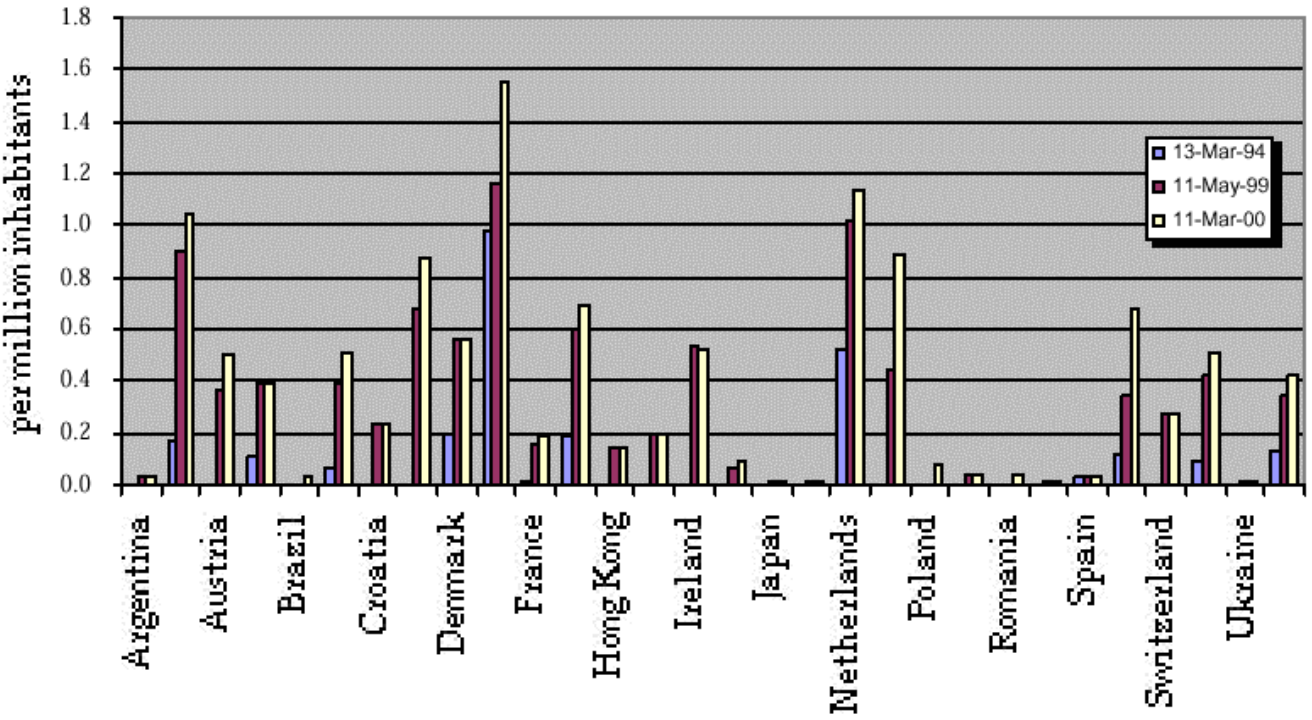
Vanaf versie 1.0.0 werd de CREDITS file toegevoegd aan de Linux. In deze file worden programmeurs genoemd die een substantiële bijdrage hebben geleverd aan de Linux-code. Hoewel de file niet alle namen van creditors bevat, geeft het wel een indicatie voor de groei van de Linux community (zie figuur 4).

[33] Zie eerste aankondigingen van Linux door Linus Torvalds in appendix D.

kernel versie	datum van uitgifte	aantal genoemde mensen
1.0.0	13 maart 1994	81
1.2.0	7 maart 1995	129
2.0.0	9 juni 1996	191
2.2.17	4 september 2000	294
2.4.0-test 10	31 oktober 2000	372 (18 Nederlands)

Figuur 4: Genoemde mensen in CREDITS file van Linux-kernel door de jaren heen.

Op deze wijze ontstond een virtual community waarvan de deelnemers verspreid waren over de hele wereld (zie figuur 5). Het internationale karakter van de Linux ontwikkelaarsgemeenschap is in de loop der jaren sterk toegenomen. Ook vanuit derdewereldlanden worden in toenemende mate bijdragen geleverd.



Figuur 5: Herkomst van Linux-programmeurs (Weber 2000, p.14).

De programmeurs communiceren met elkaar via mailing lists, emails en ftp. De openbare linux-kernel mailing list is het centrale forum voor kernelontwikkelaars.[34] Torvalds zelf participeert met hoge regelmaat ook aan dit forum. Feature freezes, code freezes en nieuwe releases worden via deze list aangekondigd. Programmeurs die code willen toevoegen aan de

kernel posten deze op de list. Andere programmeurs kunnen de code downloaden, testen op hun eigen apparatuur en suggesties doen aan de auteur of de code ondersteunen. Vanaf juli 1995 tot april 2000 hebben ongeveer 13.000 mensen bijna 175.000 boodschappen gepost op de linux-kernel mailing list (Moon, 2000). De hoeveelheid boodschappen bedraagt in bepaalde weken zelfs 6 MB (Kuwabara, 2000). Geschat wordt dat er ondertussen ongeveer 40,000 mensen een bijdrage hebben geleverd aan Linux (Raymond, 1999).

De twee belangrijkste functies binnen de gemeenschap zijn credited developer en maintainer. De functie van maintainer werd in februari 1996 officieel erkend door de MAINTAINERS file. De eerste file bevatte slechts drie namen. Tegenwoordig zijn er ongeveer 150 maintainers. Een maintainer is verantwoordelijk voor een bepaalde module van de kernel (Moon, 2000).

Iedereen kan een bijdrage ontwikkelen en aandragen via de mailing list. Er is echter sprake van een informele hiërarchie. Linus Torvalds is nog steeds de onbetwiste leider het Linux-project. Hij wordt vaak omschreven als een benevolente leider. Torvalds werkt nauw samen met de zogenaamde informele *"Inner Circle"* van technische adviseurs ofwel *"lieutenants"*. De programmeurs binnen deze groep hebben hun sporen reeds verdiend. Alan Cox is naast Torvalds de belangrijkste programmeur. Terwijl Torvalds zich met name toelegt op de experimentele versies, is Cox primair verantwoordelijk voor het onderhoud aan de stabiele versies van Linux. Torvalds kondigt de stabiele versies nog wel officieel aan, maar dit is meer een *pro forma* handeling (Moon, 2000). De stabiele versies van de Linux-kernel worden aangeduid met een even nummer (minor number) na het hoofdversienummer (major number) en de experimentele versie met een oneven nummer. Experimentele versies zijn bijvoorbeeld 1.1 en 2.3, terwijl 1.0 en 2.4 beide stabiele versies vormen (Kuwbara, 2000). Experimentele versies gaan uiteindelijk over in stabiele versies.

Linux kernel version *X.Y.Z* (bijv. 2.4.3)
X = Major Number
Y = Minor Number; even=stabiele versie | oneven=ontwikkelingsversie
Z = Increment Number

Figuur 6: Linux versie-systeem

[34] Zie voor samenvattingen en archief: http://kt.zork.net/kernel-traffic/latest.html.

Momenteel, oktober 2001, is versie 2.4.12 de meest actuele stabiele kernel. Er is na de eerste uitgifte binnen de 2.4 tak begin 2001 geen nieuwe experimentele tree opgestart. Echter, naast de 2.4 tak, die wordt onderhouden door Torvalds, onderhoudt Alan Cox een eigen stabiele 2.4-ac tak. Beide kennen verschillende implementaties voor virtual memory (VM). Dit is een ongebruikelijke situatie. Het lijkt meer op gezonde, interne concurrentie dan op een fundamentele afsplitsing. De verwachting is dat beide binnenkort grotendeels zullen integreren, en dat door Torvalds gestart wordt met de experimentele 2.5 tak.[35]

Al zeer snel werd het voor Torvalds duidelijk dat Linux zo modulair mogelijk opgebouwd diende te worden. Torvalds onderschrijft dit als volgt:*"Het open-source model vereist dit [modulariteit] echt, omdat je anders mensen niet eenvoudig parallel kan laten werken"*. Versie 2.0, die werd geïntroduceerd in 1996, vormde op het gebied van modulariteit een enorme verbetering. Voor het eerst werden zogenaamde loadable kernel modules toegevoegd (Torvalds, 1999).

Sinds eind jaren '90 tonen steeds meer computerbedrijven interesse in Linux. Hardwareproducenten leveren hun computers, met name servers, met Linux, terwijl softwarefabikanten ook versies voor Linux uitbrengen. Onder deze bedrijven bevinden zich onder andere IBM, HP, Compaq, Nokia, SAP, Motorola, Oracle, Novell, Dell, Silicon Graphics en Informix (IDC, 2000, p.18/19). In 1998 kondigden zowel Oracle en Informix, twee van de meest invloedrijke database ontwikkelaars, dat zij hun softwareproducten gereed zouden maken voor Linux (Weber, p.11). En meer dan 20 procent van de IBM servers die in het laatste kwartaal van 2000 verkocht werden, had Linux als besturingssysteem (Webwereld, 2001b).

Het marktaandeel van Linux in de servermarkt werd voor 2000 op ongeveer 27% geschat, terwijl Microsoft een aandeel van 41% zou hebben gehad (Webwereld, 2001b). Hoewel Linux de laatste jaren op de servermarkt terrein wint, speelt zij op de desktopmarkt tot op heden nauwelijks een rol. Geschat wordt dat Microsoft in 2000 92% van deze markt voor zijn rekening nam, en Linux slechts 2% (Webwereld, 2001a). Echter de laastste jaren wint Linux aan gebruiksvriendelijkheid en is er wel sprake van een sterk groei van Linuxgebruik op

[35] Zie voor overzicht Linux-kernel historie: http://www.lwn.net onder Kernel.

desktop. Bovendien wordt Linux ook meer en meer ingezet als embedded system.[36] In totaal wordt het aantal Linux-installaties wereldwijd rond de 18 miljoen geschat.[37]

3.3 Apache

De ontwikkeling van de Apache webserver-software begon in 1994. De 21- jarige Brian Behlenhof was verantwoordelijk voor één van de eerste commerciële webservers, waarop op de website van magazine Wired, genaamd Hotwired, gehost was. Op de webserver werd gebruik gemaakt van de in die tijd meest populaire webserver software van de universitaire organisatie NCSA.[38] Deze organisatie verstrekte tevens de broncode. Een ontwikkelteam was actief betrokken bij het verbeteren van de code in overleg met pioniergebruikers, onder wie Behlendorf.

Echter, Behlendorf en een aantal andere gebruikers raakten in toenemende mate gefrustreerd omdat het NCSA steeds vaker niet reageerde op hun suggesties. Samen met zeven andere gebruikers startte Behlendorf de Apache Group met als doel de ontwikkeling van nieuwe robuuste webserversoftware op basis van de NCSA-software. Afgesproken werd dat de ontwikkeling een proces van open samenwerking zou zijn. De groep koos voor een BSD-achtige licentie, die commercieel gebruik van de software onder bepaalde voorwaarden zou toelaten. Echter, de naam Apache stelden ze via een merkdepot wel veilig, zodat deze alleen door henzelf voor de software mocht worden gebruikt (Lerner, 2000).

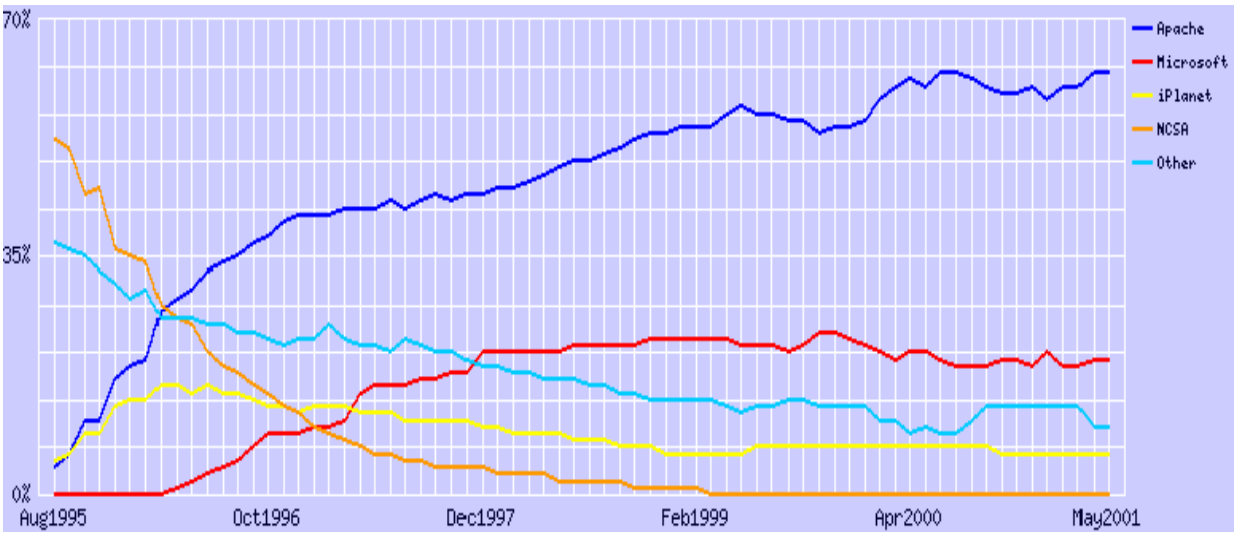
Hoewel een groot aantal mensen via de mailing list suggesties kon doen, zou een kleinere groep daadwerkelijk codewijzigingen kunnen doorvoeren. In augustus 1995 werd Apache versie 0.8 uitgebracht. Deze versie bevatte zeer belangrijke wijzigingen ten opzichte van eerder versies. De herziening van de Application Program Interface (API) zorgde ervoor dat de modulariteit van het programma enorm toenam. Programmeurs konden nu bijdragen leveren aan een bepaald onderdeel van Apache zonder dat dit uitwerking had op andere onderdelen. Behlendorf benadrukt het belang van deze verandering achteraf als volgt: *"In the Apache project, we were fortunate in that early on we developed an internal API […]. Having a powerfull API has allowed us to hand of other big pieces of functionality, […] toseparate groups of commited developers"* (Behelendorf, 1999). Op 1 december 1995 werd de eerste

[36] Embedded Linux Consortium, http://www.embedded-linux.org
[37] http://counter.li.org (22 september 2001)
[38] NCSA: National Center for Supercomputer Applications, University of Illinois

volwassen versie, Apache 1.0, uitgegeven. Momenteel is versie 1.3.20 de meest recente Apache uitgave.

De mailing list is nog steeds het primaire communicatiemiddel. Per dag worden er ongeveer 40 boodschappen op deze lijst geplaatst. Het Apache project wordt gecoördineerd door een selecte kerngroep van contribuanten. Deze kerngroep bestond eind juli 200 uit 34 personen. Via stemmingen binnen de kerngroep, die verlopen volgens geformaliseerde procedures, vinden aanpassingen van de code plaats. Iemand die frequent bijdraagt kan op voordracht en na unanieme goedkeuring lid worden van de kerngroep. In 1999 is de non-profit Apache Software Foundation om de ontwikkeling van Apache webserversoftware (en andere open source projecten) formeel te ondersteunen (Wayner, 2000, p.237). Uit een studie bleek dat tussen 1995 en 1999 ongeveer 400 programmeurs actief bijgedragen hadden aan de Apache-code, en dat meer dan 3000 mensen melding hadden gemaakt van fouten (Mockus, 2000, p.5).

Apache is binnen korte tijd zeer populair geworden. Binnen een jaar na oprichting van de groep werd NCSA voorbijgestreefd als marktleider. Onderzoeken van Netcraft laten zien dat het marktaandeel is gestegen van 31% in april 1996 naar meer dan 60% in april 2001.



Figuur 7: Marktaandelen van verschillende soorten webserversoftware (gemeten op basis van websites), http://www.netcraft.com/survey/ (10 juni 2001).

Apache wordt inmiddels ondersteunt door verschillende traditionele computerbedrijven. Met name IBM is zeer actief met betrekking tot Apache. Momenteel zijn 4 leden van de kerngroep werkzaam voor IBM. Ook Apple levert haar besturingssysteem OSX voor servers standaard

met Apache. In de kerngroep heeft één persoon van Apple zitting. Door zitting te hebben in de kerngroep kunnen bedrijven de ontwikkeling van de Apache-software direct volgen. Bovendien hebben zij enige invloed op de ontwikkeling. Beide bedrijven zullen bepaalde door hun ontwikkelde code terug laten vloeien in het project, terwijl andere code commercieel zal blijven. Dit laatste is mogelijk door de BSD-achtige licentie die verbonden is aan de Apache software.

3.4 Afronding

Linux en Apache zijn OSS-projecten die beide in relatief korte tijd een stevige positie hebben verworven op de softwaremarkt. Linux werd gestart door een eenling, terwijl Apache werd gestart door een groep. Echter, beide ontstonden uit gebrek aan geschikte software. Bovendien konden beide projecten voortborduren op reeds bestaande concepten, respectievelijk Minix/Unix en NCSA.

Modulariteit wordt in zowel het Linux- als het Apache-project beschouwd als een doorslaggevende factor op zowel technisch als organisatorisch vlak. De broncodebestanden van de projecten werden zelfs speciaal heringericht om de modulariteit te optimaliseren. Linux is zeer informeel georganiseerd, en de oprichter speelt nog steeds een cruciale rol. Apache daarentegen kent sinds kort een formele organisatiestructuur. Zowel Linux als Apache worden inmiddels in hoge mate ondersteund door traditionele ICT-leveranciers.

Beide projecten spelen momenteel met name een rol van betekenis op serverniveau. Bij Apache is dit natuurlijk inherent aan het type software. Linux kan echter ook op desktopniveau worden ingezet. Hoewel dit in toenemende mate gebeurt, is het marktaandeel van Linux hier nog erg klein.

In het volgende hoofdstuk zullen de kenmerken van software als economisch goed uiteengezet worden.

Hoofdstuk 4: Economische kenmerken van software en de softwaremarkt

"While there is no single feature that is unique to software markets, these markets do possess a number of characteristics that collectively make the application of antitrust policy particularly subtle." – Katz/Shapiro, 1998.

4.1 Inleiding

Software is een economisch goed met een unieke combinatie van eigenschappen. Deze eigenschappen hangen nauw samen met het feit dat software een informatiegoed is. Een informatiegoed kan volgens Varian worden omschreven als *"anything that can be digitized"*. Voorbeelden zijn onder meer een boek, een film, muziek en een telefoonconversatie (Varian, 1998, p.3). Daarnaast is software in tegenstelling tot de meeste andere informatiegoederen meestal een gebruiksgoed; software dient als gereedschap. Bovendien is software in tegenstelling tot veel andere informatiegoederen in zeer hoge mate volgens de wetten der logica en in veel mindere mate volgens andere waarden, zoals die der esthetiek, opgebouwd.

Het wezen van software brengt eigenschappen met zich mee die zijn relatief korte levensgeschiedenis, zoals eerder beschreven, tot een roerige hebben gemaakt. De volgende eigenschappen van software kunnen worden onderscheiden:

- Netwerkeffecten;
- Verspreidingsgemak en collectief goed;
- Hoge ontwikkelingskosten;
- Hoge complexiteit en innovatiesnelheid.

In het onderstaande zullen deze nader worden toegelicht.

4.2 Netwerkeffecten

Software vormt de communicatieve spil van de huidige informatiemaatschappij. Niettemin heeft software *op zichzelf* geen waarde. Software kan namelijk alleen worden gebruikt in combinatie met andere componenten zoals andere software, hardware en de gebruiker. Met ieder van deze componenten worden door software via interfaces koppelingen tot stand gebracht. Deze koppelingen bestaan bijvoorbeeld uit het bestandsformaat en de grafische interface.

De koppelingen zorgen ervoor dat software gepaard gaat met netwerkeffecten. Deze laatste leiden ertoe dat de voordelen van het behoren tot een bepaald netwerk van softwaregebruikers toenemen, naarmate de desbetreffende software meer gebruikers telt. Ter illustratie van het begrip "netwerkeffecten": des te meer mensen over een tekstverwerkingssoftware X beschikken, des te gemakkelijker het uitwisselen van tekstbestanden en des te meer waarde de tekstverwerkingssoftware voor de gebruiker heeft. Bovendien zullen nog meer mensen de software willen gebruiken en toe willen treden tot het netwerk. Er treedt positieve feedback op waardoor de sterke aanbieders nog sterker dreigen te worden en de zwakke nog zwakker (Katz/Shapiro, 1998, p. 2 e.v.).

Door een opeenstapeling van koppelingen kunnen netwerkeffecten via een kettingreactie doorwerking hebben op achterliggende niveaus. Als bepaalde software veel gebruikers heeft, dan zal er meer complementaire software, hardware en wetware (dat wil zeggen hersenen) beschikbaar zijn. In de grafisch wereld wordt bijvoorbeeld gewerkt met bepaalde programma's en bestandsformaten, die alleen optimaal beschikbaar zijn voor het Apple besturingssysteem, dat op haar beurt alleen draait op Apple hardware. Daarom zijn Apple computers de '*de facto'* standaard in de grafische wereld. Een ander voorbeeld vormt Microsoft dat door de beheersing van de markt voor besturingssystemen ook steeds meer de markt voor applicaties beheerst. Door een voor Microsoft positieve informatie-asymmetrie kan het bedrijf de applicaties eerder en beter koppelen aan het besturingssysteem dan de concurrentie.

Maximale economische efficiëntie wordt verkregen als alle gebruikers binnen één netwerk actief zijn. Shapiro (2000) zegt hierover: *"When all users are on a single network, the size of the network is maximized and so is the realization of network benefits."* In bepaalde gevallen zijn de netwerkeffecten zo sterk dat de markt daadwerkelijk neigt naar één netwerk. Indien er nog geen standaard is kan langs twee wegen een standaard worden bereikt. Bedrijven kunnen elkaar bestrijden met verschillende, incompatibele technieken, of bedrijven kunnen vroegtijdig een standaard overeenkomen. In het eerste geval concurreren de bedrijven *voor de markt*, en zal uiteindelijk een *de facto* standaard ontstaan, welke 'proprietary' is oftewel beheerst wordt door één bedrijf. In het tweede geval concurreren de bedrijven *in de markt* en wordt gebruik gemaakt van een *de jure* standaard, die open is en niet beheerst wordt door één aanbieder (Shapiro, 2000, p.9).

Door de netwerkeffecten kan er voor de gebruiker een zogenaamde '*lock-in*' ontstaan. De gebruiker zit als het ware gevangen in het netwerk waarbinnen hij acteert. Overschakelen naar een ander netwerk kan door de kracht van het reeds door hem gebruikte netwerk enorme kosten met zich meebrengen. Deze kosten worden '*switchingcosts*' genoemd. Volgens Meltcafe neemt de waarde van een netwerk bijna kwadratisch toe met het aantal deelnemers (Shapiro, 1999, p.8).[39] De collectieve switchingcosts kunnen daardoor tot extreme hoogte groeien.

Voor de producent, die het netwerk beheerst, ontstaan er sprake van enorme schaalvoordelen aan de vraagzijde. Evenwel voor potentiële toetreders zullen hierdoor barrières ontstaan. Dit verklaart waarom gevestigde techniek, ondanks inferioriteit, toch lange tijd de *de facto* standaard kan blijven Er is dan sprake van een collectieve lock-in.

4.3 Verspreidingsgemak en collectief goed

Naast de netwerkeffecten zijn de oneindige mogelijkheden tot verspreiding zeer kenmerkend voor software. In het digitale tijdperk kan software evenals andere informatiegoederen vrijwel kosteloos gekopieerd en gedistribueerd worden. Dit biedt enerzijds enorme voordelen voor de softwareproducenten. Anderzijds geldt dat dergelijke middelen niet langer zijn voorbehouden aan een selecte groep. Internet en CD-branders zijn de gereedschappen voor verspreiding die door iedereen, zeker ook door thuisgebruikers, ingezet kunnen worden

Software voldoet daardoor sterk aan de kenmerken van een collectief goed. Voor een puur collectief goed geldt dat zijn gebruik zowel non-rivaliserend als non-exclusief is.[40] Non-rivaliserend wil zeggen dat de consumptie van additionele eenheden geen extra (marginale) kosten met zich meebrengt.[41] Non-exclusiviteit houdt in dat het onmogelijk of erg kostbaar is om gebruikers uit te sluiten van het gebruik van het goed. Zowel technische als economische mogelijkheden tot uitsluiting bepalen aldus de mate van exclusiviteit van een goed. In de

[39] Meltcafe's rule of thumb: Waarde van een netwerk = $n*(n-1) = n^2-n$ (waarbij n het aantal deelnemers is in het netwerk) (Shapiro, 1999, p.8).

[40] Rivaliteit en exclusiviteit dienen mijns inziens als relatieve begrippen te worden beschouwd. In het vervolg zal dan ook meestal gesproken worden van "de mate van rivaliteit of exclusiviteit". Met non-rivaliteit en non-exclusiviteit wordt bedoeld dat de desbetreffende eigenschap niet significant aanwezig is.

[41] Een standaard voorbeeld van een non-rivaliserend gebruik in de materiële wereld is het rijden over een brug in de *niet*-spitsuren. Men kan echter zeer wel verdedigen dat non-rivaliserend gebruik van een materieel goed naar zijn aard onmogelijk is; omdat er altijd sprake zal zijn van enige consumptie ofwel verbruik; 'de brug slijt'. Alleen immateriële goederen kunnen volgens deze visie volledig non-rivaliserend zijn (Ouwersloot, 1994, p.22).

praktijk leidt non-exclusiviteit tot de free-riderproblematiek. De free-rider weigert te betalen voor een (non-exclusief) goed, omdat hij desondanks toch kan genieten van het goed. De genoemde eigenschappen leiden ertoe dat een collectief goed *niet* of moeizaam in individuele eenheden exploitabel is. Hoewel er individuele behoefte is naar een bepaald collectief goed, faalt het marktmechanisme in beginsel in het bepalen van een individuele prijs (Varian, 1998, p.6-7)

Software is in hoge mate non-rivaliserend en non-exclusief. Door de netwerkeffecten en de verwaarloosbare verspreidingskosten kan software zelfs anti-rivaliserend genoemd worden (Weber, 2000, p.28). Verdere verspreiding gaat namelijk gepaard met zowel collectieve als individuele voordelen. De eindgebruiker wordt door de netwerkeffecten aangemoedigd om software te kopiëren en te verspreiden. Informatie-uitwisseling wordt namelijk eenvoudiger als binnen je naaste omgeving dezelfde software wordt gebruikt. Het thuiskopiëren van software voor vrienden en kennissen, zodat iedereen tekstverwerker X gebruikt en daardoor informatie-uitwisseling met hun makkelijker wordt, is hiervan een voorbeeld.

Commerciële softwarebedrijven proberen de exclusiviteit (en rivaliteit) van software af te dwingen met zowel technische hindernissen als juridische middelen. Technische hindernissen bestaan voor de eindgebruiker uit bijvoorbeeld kopieerbeveilingen en voor de concurrent met name uit de geheimhouding van de broncode. De juridische middelen die ingezet worden omvatten het intellectueel eigendomsrecht (het auteursrecht en het octrooirecht) en de licentieovereenkomsten waarmee ieder software-exemplaar vergezeld gaat.

Hoewel de geheimhouding van de broncode in combinatie met de rechten van intellectuele eigendom een redelijk sterk middel blijkt te zijn om de concurrent buiten de deur te houden, levert dit ernstige nadelen op voor de eindgebruiker. De eindgebruiker heeft ten eerste geen zicht op de achterliggende processen van de software, waardoor hij min of meer dient te vertrouwen op de goedheid van de producent. Daarnaast kan de eindgebruiker de onvolkomenheden die vrijwel ieder programma bevat niet zelf (laten) oplossen en kan deze de software niet geheel aanpassen aan zijn eigen wensen.

De kopieerbeveilingen en de juridische middelen om de eindgebruiker te weerhouden van het verder verspreiden van de software blijken in de praktijk geen waterdichte blokkade. Uit

onderzoek blijkt dat software in groten getale illegaal gekopieerd wordt.[42] Overigens geldt dit niet alleen voor software, maar ook voor andere informatiegoederen, zoals muziek, film en geschreven tekst. Het illegaal verspreiden van software hoeft door de daarmee gepaard gaande netwerkeffecten niet perse een negatieve invloed te hebben op de marktpositie van een commercieel softwarebedrijf. Het succes van WordPerfect aan het begin van de jaren '90 wordt bijvoorbeeld vaak (ten dele) toegeschreven aan het veelvuldig kopiëren door particulieren. En door Internet Explorer gratis beschikbaar te stellen, bewandelt Microsoft een vergelijkbare weg, die inmiddels ook succesvol is gebleken.

4.4 Ontwikkelingskosten

In tegenstelling tot het kopiëren en verspreiden van software zijn de ontwikkelingskosten van software net als voor veel andere informatiegoederen relatief hoog. Bovendien zijn de initiële ontwikkelingskosten grotendeels te beschouwen als sunk cost (Varian, 1998, p.5-6; Zimmerman, 1997, p.28). Dit heeft tot gevolg dat indien het informatieproduct niet succesvol is, de ontwikkelingskosten als verloren beschouwd dienen te worden. Ter illustratie: het ontwikkelen van een nieuwe softwareprogramma kan miljoenen guldens kosten. Als het programma eenmaal geschreven is, zijn de kosten van kopieën of de verspreidingskosten via internet relatief laag. De marginale kosten zullen daarom voor de producent altijd lager zijn dan de gemiddelde kosten. De hoge ontwikkelkosten kunnen worden uitgesmeerd over de hoeveelheid kopieën. Des te meer softwarepakketten verkocht worden, des te lager zullen de gemiddelde kosten zijn voor de producent.

De hoge ontwikkelingskosten brengen aanzienlijke investeringsrisico's me zich mee. Zij vormen daarmee een drempel voor toetreding tot de softwaremarkt. Echter, door de lage verspreidingskosten doen zich enorme schaaleffecten voor aan de aanbodzijde. Daardoor kunnen bij succes van een softwarepakket enorme winsten worden gemaakt.

4.5 Complexiteit en innovatiesnelheid

Met name de complexiteit en de hoge innovatiesnelheid zijn de oorzaak van de hoogte van de ontwikkelingskosten. De enorme complexiteit van software vertaalt zich onder meer in de met hoge regelmaat opduikende fouten in software. Ook de omvang van de broncode kan als een

[42] "Onderzoek heeft uitgewezen dat het percentage zakelijke illegale software in Europa 39% bedroeg in 1997. […] In 1997 werd het percentage aan illegale software in Nederland berekend op 48%." (Business Software Alliance (BSA) – http://www.bsa.nl / http://www.bsa.org).

indicatie voor de complexiteit van software worden beschouwd. De broncode van een recente Linux-kernel (versie 2.1.110) bevatte bijvoorbeeld alleen al meer dan 1,5 miljoen coderegels (Moon, 2001).

De complexiteit van software lijkt onder meer toe te nemen door de groeiende informatie-uitwisseling waarbij software een spilfunctie vervult. De IT-markt is relatief jong en daardoor nog volop in ontwikkeling. De oneindige hoeveelheid omgevingsfactoren die van invloed uitoefenen op het functioneren van de software zijn vrijwel onmogelijk na te bootsen middels proefopstellingen. Bovendien is testen erg kostbaar en kan daardoor de ontwikkelingskosten aanzienlijk opdrijven (Koning,1999).

4.6 Commerciële softwaremarkt

Commerciële softwarebedrijven proberen ten voordele van zich in te spelen op de bovenbeschreven karakteristieken. Aan de ene kant wordt gepoogd door geslotenheid het collectieve karakter van software te onderdrukken. De geslotenheid komt terug in middelen zoals geheimhouding van de broncode, kopieerbeveiligingen en juridische wapens. Aan de andere kant tracht men door een combinatie van techniek en marketing te profiteren van de netwerkeffecten en het verspreidingsgemak. Middelen zoals koppelverkoop, prijsdiscriminatie, blokkeren van compatibiliteit en het controleren van interfaces worden hierbij door softwareproducenten ingezet.[43] Uiteindelijk wordt een groot deel van de opbrengst geïnvesteerd als ontwikkelingskosten, die hoog zijn door de enorme complexiteit van software, en marketingkosten.

4.7 Afronding

De netwerkeffecten en het verspreidingsgemak zijn beide enorme krachten die de softwaremarkt beheersen. De softwareproducent kan door het verspreidingsgemak profiteren van enorme schaalvoordelen aan de aanbodzijde, terwijl de netwerkeffecten zorgen voor schaaleffecten aan de vraagzijde. Echter, deze krachten zorgen er tevens voor dat software veel eigenschappen in zich draagt van een collectief goed. De gebruiker kan namelijk ook van de krachten profiteren.

[43] Tying en bundling zijn voorbeelden van koppelverkoop. In deze gaat het om het aanbieden van twee programma's als een pakket, waardoor de consument (om technische of economische redenen) gedwongen wordt de gekoppelde software ook af te nemen (Katz, 1998, p.40 e.v.).

Het internet is een belangrijke motor waardoor zowel het verspreidingsgemak als de netwerkeffecten enorm zijn toegenomen. Met de verdere volwassenwording van het internet zullen deze eigenschappen nog sterker naar voren treden. Verspreiding van software zal nog eenvoudiger worden, en de toenemende informatie-uitwisseling zal nog krachtigere netwerkeffecten met zich meebrengen.

Behalve bovengenoemde krachten wordt software gekenmerkt door een hoge mate aan de complexiteit en de innovatiesnelheid. Het ontwikkelen van software brengt daardoor hoge kosten met zich mee. Deze kosten zijn grotendeels *'sunk'*, dat wil zeggen dat zij gemaakt zijn en niet meer veranderd kunnen worden. Dit brengt grote investeringsrisico's en daardoor toetredingsdrempels met zich mee.

De gevolgen van het enorme krachtenveld op de softwaremarkt zijn uiteenlopend. Aan de ene kant kan het bedrijf dat de krachten ten voordele van zichzelf weet te richten hiervan extreem profiteren. De markt neigt hierdoor naar een monopolie. De huidige machtspositie van Microsoft onderschrijft dit als geen ander.[44] Echter, aan de andere kant lijken de krachten in toenemende mate voor bedrijven onbeheersbaar te worden. Het illegaal kopiëren van software heeft bijvoorbeeld een enorme vlucht genomen.

In dit krachtenveld blijkt Open Source Software steeds meer een geduchte concurrent voor commerciële softwareontwikkelaars. Open Source Software lijkt de krachten op geheel eigen wijze te benutten. OSS-ontwikkelaars proberen bijvoorbeeld het collectieve karakter van software niet te onderdrukken, maar juist ten eigen bate te gebruiken. In de volgende hoofdstukken worden achtereenvolgens de individuele motivatie, de organisatie en de ontwikkelingsmethode nader onderzocht. De eigenschappen van software, zoals beschreven in dit hoofdstuk, hebben op ieder van deze drie vlakken een significante invloed.

[44] Met name op de desktopmarkt heeft Microsoft vrijwel een monopolie met een geschat marktaandeel van 92 procent. In deze markt heeft Linux naar schatting slechts een marktaandeel van 2 procent (Webwereld, 2001a).

Hoofdstuk 5: Motivatie van OSS-programmeurs

"I write software because I just love doing it" – Rob Newberry, Linuxfan en programmeur (Wayner, 2000, p.160).

5.1 Inleiding

In dit hoofdstuk zal de individuele motivatie van OSS-programmeurs nader tegen het licht worden gehouden. Het sociale fundament op micro niveau van Open Source projecten, en mate name van Linux, is op het eerste gezicht verbazingwekkend. Theorie over collectieve goederen voorspelt dat non-rivaliteit en non-exclusiviteit juist aanmoedigen tot *free-riding*. Met name wanneer alleen door samenwerking van veel mensen een collectief goed van waarde kan worden verkregen zou dit in een vrije markt leiden tot een tekort aan dat specifieke goed. Toch kiezen veel getalenteerde programmeurs vrijwillig om een substantieel deel van hun schaarse tijd en hersencapaciteit te besteden aan een project waarvoor ze niet direct worden gecompenseerd en waarvan ook anderen de vruchten kunnen plukken (Weber, 2000, p.5).

Uitgaande van de economische theorie draagt een programmeur alleen bij aan een project als dit voor hem of haar een positieve nettowaarde in brede zin (ofwel nut) oplevert. Deze netto-waarde bestaat uit de opbrengst min de kosten van zijn inzet. De primaire kosten zijn de *'opportunity costs'* van tijd en moeite die de programmeur spendeert aan het project. Overigens zullen deze kosten lager worden naarmate OSS meer wordt gebruikt, doordat programmeurs reeds via school en/of universiteit vertrouwd zijn geraakt met de software en zijn broncode. Dit wordt ook wel het *'alumni-effect'* genoemd. Dit effect lijkt ook op macro niveau kostenbesparingen met zich mee te kunnen brengen (Lerner, 2000, p.16). Door de openheid van de broncode kunnen er aldus positieve netwerkeffecten op ontwikkelingsniveau ontstaan. Aan de kostenkant is behalve de inspanning van de programmeur bepalend dat de directe kosten van beschikbaarstelling van de software met broncode voor de programmeur nihil zijn.

In de literatuur is veel geschreven over de baten oftewel motivatieredenen van programmeurs om bij te dragen aan Open Source Software projecten. Deze redenen blijken uiteenlopend en kunnen per persoon verschillen. Bovendien zal het vaak gaan om een combinatie van reden. Van belang is in ieder geval de informele en vrije context waarin het bijdragen aan OSS-

projecten plaatsvindt. Een programmeur kan bijvoorbeeld zelf bepalen voor welk (deel van het) programma hij bijdragen levert. Overigens zijn de redenen niet alleen kenmerkend voor de OSS-gemeenschap, maar kan een parallel getrokken worden met de redenen van mensen om vrijwillig openbaar informatie uit te wisselen via internet op bijv. nieuwsgroepen en websites (Gosh, 1998).

In hoofdlijnen kunnen de volgende categorieën van motivatie te worden onderscheiden.

- Plezier;
- Zelfontplooiing;
- Software als beloning;
- Politiek / Idealisme;
- Financiele Beloning.

Aan de hand van de bovenstaand categorisering zal de motivatie van OSS-programmeurs in het onderstaande nader worden bekeken.

5.2 Plezier

Mensen halen plezier uit het bepaalde bezigheden, zoals het maken van muziek, het oplossen van kruiswoordraadsels of het spelen van een spel. Dit geldt ook voor veel programmeurs. Zij zijn gefascineerd door het ontwikkelen van software en zullen dit blijven doen ongeacht of ze hiervoor beloond worden (Stallman, 1992). De beloning is intrinsiek, deze ligt besloten in *het doen*. Overigens zal plezier meer een motivatie zijn om software te ontwikkelen dan om deze ook te verspreiden (Kuwabara, 2000).

Plezier wordt vaak niet erkend als motivatie van OSS-programmeurs. De oorzaak hiervan lijkt het feit dat plezier moeilijk is te plaatsen binnen de economische theorie. Bovendien lijkt plezier vaak als paraplubegrip voor motivatieredenen te worden aangevoerd door OSS-programmeurs. Echter, plezier wordt door vele OSS-programmeurs zo uitdrukkelijk genoemd dat haar een eigen plaats dient te worden toegekend. Torvalds laat het element plezier bijvoorbeeld terugkomen in de titel van zijn autobiografie *"Just for fun, the story of an accidental revolutionary"* (Torvalds, 2001).

5.3 Zelfontplooiing

Creaties van veel mensen, ook van programmeurs, eindigen vaak een stille dood. De schaarse tijd en hersencapaciteit van een mens brengt met zich mee dat niet al zijn creaties blijvend

zijn prioriteit hebben. Bovendien beschikt niet ieder mens over voldoende externe middelen, zoals kapitaal, om zijn creatie tot een levendig succes te maken. Door software met broncode via het internet beschikbaar te stellen kan een programmeur ervoor zorgen dat zijn creatie levend blijft. Andere programmeurs kunnen zijn software namelijk verder ontwikkelen.

Behalve het feit dat zijn creatie in leven gehouden wordt, kan een programmeur ook leren van het bijdragen aan Open Source projecten. Een OSS-programmeur zal bestaande code moeten bestuderen om een bijdrage te kunnen leveren. Bovendien zal de feedback van andere programmeurs op zijn bijdrage voor hem leermomenten creëren. In praktijk blijken veel studenten en wetenschappers bij te dragen aan OSS-projecten. Bovendien vinden veel OSS-projecten, zoals Linux en Gimp, hun oorsprong direct of minder direct in de universitaire wereld.

Tevens kunnen er opbrengsten zijn voor de programmeur die tezamen kunnen worden aangemerkt als de zogenaamde *"signalling incentive"*. Deze incentive kan worden uitgesplitst naar twee vormen, namelijk de *"career concern incentive"* en de *"ego gratification incentive"*. De eerste heeft betrekking op het in de toekomst mogelijk ten gelde maken van een reputatie door bijvoorbeeld een betaalde dienstbetrekking of toegang tot venture capital. Verschillende OSS-programmeurs werken tegenwoordig voor aansprekende bedrijven of hebben een eigen bedrijf kunnen oprichten.[45] Met de tweede incentive wordt gedoeld op het verlangen om erkend te worden binnen de groep van vakgenoten. De erkenning kan bijvoorbeeld gestalte krijgen door een stijging binnen de vaak informele hiërarchie van een OSS-project of doordat andere programmeurs zich richting de desbetreffende programmeur meer coöperatief opstellen. Beide incentives zijn nauw met elkaar verbonden en veel programmeurs zullen in de regel op beide reageren. Echter, er zijn wel verschillen tussen beide. De programmeur die erkenning van vakgenoten van groot belang vindt zal zijn talent toch aan een iets andere publiek willen tonen dan degene die wordt bewogen door om carrièremotieven (Lerner, 2000, p.20; Raymond, 1999).

Binnen de Open Source beweging wordt het presenteren van de namen van programmeurs op een prominente plek in de software als zeer belangrijk ervaren. Bovendien is de prestatie van een programmeur in een Open Source Software goed zichtbaar omdat de broncode

[45] Voorbeelden zijn: Alan Cox in dienst bij Red Hat, Linus Torvalds in dienst bij Transmeta, Eric Allman oprichter van Sendmail inc., en William Joy oprichter van Sun Microsystems.

toegankelijk is (Lerner, 2000, p. 19). Het wereldomvattende internet vormt natuurlijk het ideale medium om deze signalen uit te vergroten. Onder meer op deze wijze kunnen de "*signalling incentives*" van programmeurs worden gevoed.

Een programmeur kan aldus in zelfontplooiing motivatie vinden om bij te dragen aan een OSS-project. Deze zelfontplooiing kent verschillende gradaties. Het levend houden van de eigen creatie, de leereffecten en zelfmanifestatie kunnen alle de programmeur bewegen software te schrijven en/of deze te delen.

5.4 Software als beloning

Software is een complex gebruiksgoed. Gebruikers stuiten dan ook regelmatig op onvolkomenheden in de software. Onder deze gebruikers bevinden zich ook programmeurs, die de capaciteiten en de wil hebben de software naar eigen wens te wijzigen. De onvolkomenheden bij het eigen gebruik van de software doen voor de programmeur een behoefte ontstaan. De beschikbaarheid van de broncode maakt het voor hen mogelijk deze behoefte te bevredigen. Programmeurs kunnen fouten herstellen, toevoegingen maken en de software op maat aanpassen. Raymond omschrijft dit als *"scratching an itch"* (Raymond, 1999). Het uiteindelijke doel is het verkrijgen van betere software voor eigen gebruik. De vergelijking kan worden gemaakt met de automonteur die ontevreden is met de prestatie van zijn eigen auto en daarom aanpassingen verricht aan de motor.

Als de programmeur de aangepaste code beschikbaar stelt via het internet kunnen geïnteresseerde programmeurs van over de hele wereld reageren. Door deze feedback kan de aanpassing verder geoptimaliseerd worden. De programmeur kan de vruchten plukken van het werk van andere programmeurs. Uiteindelijk kan dit proces de originele programmeur software opleveren die voor hem superieur is aan software, die of volledig door hem zelf is ontwikkeld of volledig wordt ontwikkeld en beheerst door een derde (commerciële) partij. In ruil voor de beschikbaarstelling van zijn eigen inspanning wordt de programmeur aldus beloond met (suggesties voor) aanvullende aanpassingen die de software nog meer kunnen vormen naar zijn gebruikswensen (Ghosh, 2000).

5.5 Politiek & Idealisme

Zoals reeds beschreven in Hoofdstuk 4 zijn er enorme krachten, onder meer netwerkeffecten en verspreidingsgemak, actief op de softwaremarkt. Het krachtenveld lijkt bovendien in

sterkte toe te nemen. Commerciële softwarebedrijven worstelen zichtbaar met deze krachten. Hoewel niet altijd volledig bewust, weten OSS-programmeurs dat zij in dit krachtenveld als winnaars naar voren kunnen komen. Open Source Software heeft reeds ten dele bewezen dat zij de potentie heeft de aanwezige krachten ten eigen bate in te zetten (Weber, 2000, p.27). Dit heeft een positieve invloed op de motivatie.

De commerciële software-industrie, en natuurlijk met name Microsoft, wordt soms ook opgevoerd als bron van motivatie. De monopoliepositie van Microsoft heeft bij sommigen de ogen doen openen en bij anderen voor bevestiging gezorgd. Doorgaans is de houding van vooraanstaande OSS-ontwikkelaars ten opzichte van Microsoft professioneel. Het bedrijf wordt gezien als een zeer belangrijke concurrent. Door de concurrentie worden ook OSS-ontwikkelaars scherp gehouden. Een voorbeeld diende zich aan in 2000 toen onderzoekresultaten werden gepubliceerd waaruit bleek dat MS-Windows NT bepaalde serverfuncties beter uitvoerde dan Linux. Torvalds reageerde hierop als volgt: *"The point was that it actually gave us a much better baseline to compare what we were bad at. […] Then having somebody do that comparison was very motivational."* (McMillan, 2000).

Ook bestaat er onder bepaalde OSS-programmeurs de ideële opvatting dat software vrij moet zijn *("software should be free").* Richard Stallman, de oprichter van de Free Software Foundation, is een groot voorvechter van deze visie. Voor de gehele maatschappij zou het beter zijn als alle software vrij was. Het OSI poogt de nadruk juist niet te leggen op deze idealistisch getinte opvattingen, maar heeft een meer pragmatische benadering. Om het draagvlak voor Open Source Software te vergroten wijst het OSI op de kwaliteit van veel OSS-software.

5.6 Financiële beloning

Natuurlijk kan een financiële beloning ook een reden van motivatie vormen voor OSS-programmeurs. Universiteiten of andersoortige onderzoeksinstituten kunnen uit algemene gelden (en eigen inkomsten) bijvoorbeeld programmeurs betalen om in het belang van de wetenschap bij te dragen aan softwareprojecten. Om wetenschap optimaal te bedrijven is het voor deze instellingen van belang dat de broncode gepubliceerd wordt.

Tevens dragen steeds meer programmeurs in opdracht van commerciële bedrijven bij aan OSS-projecten. Bedrijven die Open Source Software niet als primaire kernactiviteit hebben,

zoals hardwareleveranciers en veel internet providers, maken tegen beloning gebruik van de diensten van Open Source specialisten. Bij IBM werken er bijvoorbeeld ongeveer 1500 mensen aan Linux (Wilcox, 2000b). Bedrijven dragen vaak actief bij aan software-ontwikkeling om de software te optimaliseren voor hun eigen producten, ondanks het feit dat het niet hun eigen software is. Dit laatste is ook niet hun primaire belang; de bedrijven zien OSS als een middel om meer van hun eigen goederen of diensten te verkopen.

Ook de OSS-distributeurs hebben veel programmeurs in dienst. Red Hat heeft bijvoorbeeld ongeveer 600 mensen in dienst, onder wie Linux kernontwikkelaar Alan Cox (Shankland, 2001).

Ook werknemers van bedrijven, die eindgebruiker zijn, kunnen bijdragen leveren. De aanpassingen zijn primair bedoeld om de interne processen van het bedrijf te verbeteren. Toch kunnen deze aanpassingen al dan niet verplicht openbaar worden gemaakt.

5.7 Afronding

Er zijn verschillende redenen aan te wijzen die programmeurs bewegen om bij te dragen aan OSS-projecten. Plezier, zelfontplooiing, software als beloning, politiek & idealisme en financiële beloning kunnen alle drijfveren vormen voor een OSS-programmeur. De werkelijke motivatie is moeilijk te achterhalen en zeker niet kwantificeerbaar. Meestal zal een programmeur bewogen worden door een complex van motivatieredenen die nauw met elkaar verstrengeld zijn.

Het moge duidelijk zijn dat een programmeur daadwerkelijk baat kan hebben bij het deelnemen aan een OSS-project. Ook hier spelen netwerkeffecten een rol. Door het verspreiden van de software inclusief broncode kan de programmeur namelijk positieve feedback ontvangen. Deze positieve feedback bestaat bijvoorbeeld uit een leereffect, verbeterde reputatie en/of verbeterde software. Het aantal (potentiële) gebruikers zal sterk bepalend zijn voor de hoogte van deze positieve feedback en dus ook voor de motivatie.

De meeste programmeurs hebben natuurlijk een overwegend technisch achtergrond. Daarom is ook de technische aard van een taak van invloed op de motivatie van de programmeur. Een programmeur zal in de regel minder gemotiveerd zijn om de software gebruiksvriendelijk te maken of om geï ntegreerde handleidingen te schrijven (Lerner/Tirole, 2000, p.19). De

signalling incentive is voor dergelijke, minder technisch georiënteerde, taken bijvoorbeeld lager.

Middels de voorgaande categorisering is de bron van Open Source Software op micro niveau meer inzichtelijk gemaakt. In het volgende hoofdstuk zal worden bekeken hoe de individuele, borrelende bronnen gebundeld worden om te komen tot een volwaardig eindproduct.

Hoofdstuk 6: Organisatie en ontwikkelingsproces bij OSS-projecten

"The open source model of 'development at a distance' is a compelling solution to complexity management in software creation. It forces modularity, resulting in code that is generally more elegant, more secure, and more reliable than alternative techniques of software." – Varian Hall (Berger, 1999)

6.1 Inleiding

In dit hoofdstuk zullen achtereenvolgens de organisatie en het ontwikkelingsproces van OSS-projecten centraal staan. Beide invalshoeken zijn complementair en kunnen elkaar zelfs enigszins overlappen. Architectuur en modulariteit vervullen in beide invalshoeken een spilfunctie. Daarom zullen deze nauw verwante concepten voorafgaand aan de organisatie en het ontwikkelingsproces behandeld worden.

Overigens dient opgemerkt te worden dat ieder OSS-project uniek is. Er bestaat geen ideaal draaiboek voor Open Source ontwikkeling. Hoewel bij bepaalde succesvolle projecten de broncode bijvoorbeeld wel beschikbaar wordt gesteld, vindt de ontwikkeling toch in relatieve beslotenheid plaats (bijv. BIND en XFree86). Ook OSS ontkomt aldus niet aan de zogenaamde '*contingency theory*', die stelt dat er geen universele, optimale organisatievorm bestaat (Wigand, 1997, p. 207).

In dit hoofdstuk zullen met name de eerder beschreven voorbeelden, Linux en Apache, als uitgangspunt dienen. Bovendien zullen ook andere OSS-projecten aangehaald worden ter onderbouwing van de bevindingen.

6.2 Architectuur en modulariteit

Bij de ontwikkeling van software speelt architectuur een cruciale rol om de complexiteit het hoofd te bieden (Maes, p.4, 2000). Architectuur fungeert als een brug tussen de aan de software gestelde eisen en de implementatie (Arief, 2001, p.1). Het concept modulariteit wordt beschouwd als een belangrijk middel om aan architectuur invulling te geven.
"[Modularity refers to] building a complex product or process from smaller subsystems that can be designed independently yet function together as a whole." (Baldwin, 1997, p.84).

Software is in tegenstelling tot bijvoorbeeld muziek een *logisch* informatiegoed. De opbouw van software voldoet namelijk in hoge mate aan de wetten der logica. Daarom is het mogelijk om software te modulariseren in een aantal op zich zelf staande programma's. Deze laatste dienen klein en in hoge mate multifunctioneel te zijn. In de UNIX-traditie spreekt men ook wel van *"do one thing simply and well"*. De onderlinge afhankelijkheid tussen de modules dient geminimaliseerd te worden. Goed gedefinieerde interfaces, bijvoorbeeld gebaseerd op open standaarden, vormen hierbij een cruciale factor (Vreven, 1994, p.60). Door combinatie van vele kleine programma's kan toch grote, complexe functionaliteit worden verkregen. De kleine programma's vormen dan samen een groter programma (Weber, 2000, p.33-34). In het algemeen geldt dat modulariteit ten voordele werkt van de flexibiliteit, uitbreidbaarheid en robuustheid. Een goed doordachte, modulaire opbouw kan bijvoorbeeld op technisch vlak zorgen voor een hoge mate van portabiliteit en interoperabiliteit.

Ook op het vlak van organisatie en ontwikkeling brengt modulariteit voordelen als flexibiliteit, uitbreidbaarheid en robuustheid met zich mee. Doordat software een logisch informatiegoed is, kan deze niet alleen technisch gezien opgedeeld worden, maar speelt modulariteit ook een rol bij de totstandkoming van de software. De technische architectuur vindt haar spiegelbeeld in de inrichting van de organisatie (Conway, 1968). Alleen door software technisch gezien in modules te ontleden, kan de organisatie ook modulair ingericht worden. *"To keep the number of people working on Linux coordinated, we needed something like kernel modules. But from a design point of view, it was also the right thing to do."* (Torvalds, 1999).

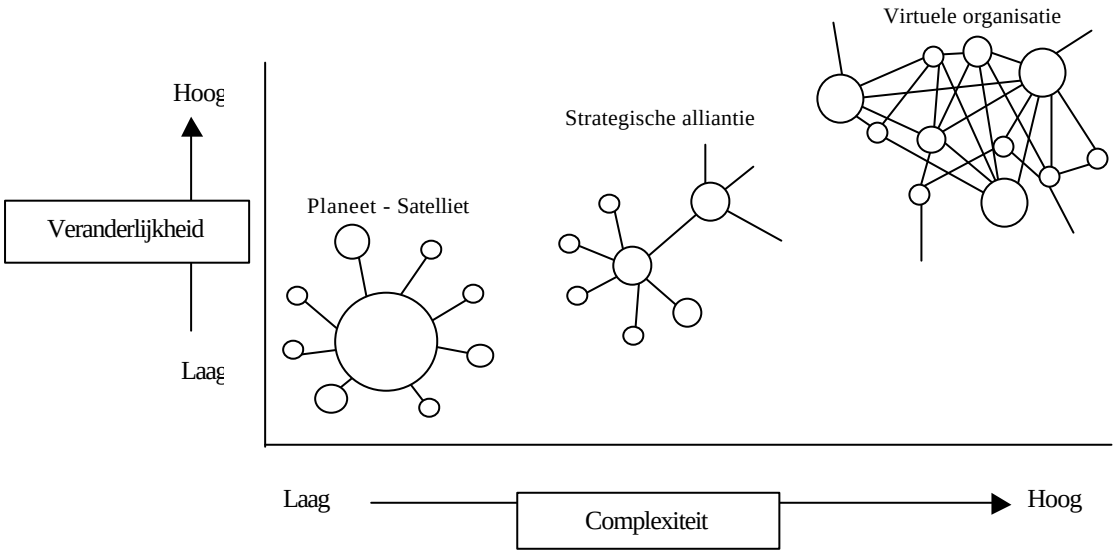
Door modulariteit zouden onder meer de transactiekosten tijdens het ontwikkelingsproces verlaagd kunnen worden (Wigand,1997, p.183). Een OSS-programmeur hoeft 'slechts' in het oog te houden dat zijn module goed kan communiceren met de andere modules. Binnen zijn module kan de programmeur idealiter aanpassingen verrichten zonder dat dit consequenties heeft voor andere modules. Er is sprake van een enorme complexiteitsreductie; de doorwerking van een verandering naar andere delen van de code wordt gereduceerd.

De commerciële software-industrie hecht sinds enige jaren in toenemende mate belang aan modulariteit (Genuchten e.a., 1993). Echter, in de OSS-wereld wordt modulariteit al decennia lang gezien als krachtig middel op zowel technisch als organisatorisch vlak. Al rond het begin van de jaren '70 werd het concept modulariteit succesvol gehanteerd in UNIX. OSS-projecten

zijn door hun aard min of meer gedwongen tot modulaire organisatie. Omdat de broncode openbaar is, kunnen mensen over de hele wereld namelijk bijdragen leveren. Om optimaal gebruik te maken van dit enorme potentieel is modulariteit een vereiste. De huidige technische middelen, zoals Internet en programmeer- en ontwikkeltools[46], hebben de mogelijkheden tot modulaire inrichting sterk doen toenemen (Wigand, 1997, p.208).

6.3 Organisatie

In recente economische literatuur (vanaf medio jaren '90) wordt veel aandacht besteed aan nieuwe organisatievormen die sterk gebruik maken van moderne technologische middelen. Met name de theorievorming over virtuele organisaties is hiervan een sprekend voorbeeld. De virtuele organisatie is de meest verregaande vorm van een netwerkorganisatie. Tegenover de virtuele organisatie kan het planeet-satelliet model worden geplaatst en tussen beide ligt de zogenaamde strategische alliantie. De indeling vindt plaats langs een glijdende schaal van centrale naar decentrale macht (zie figuur 8).



Figuur 8: Continuüm van netwerkorganisaties (Jägers, 1998; 2001).

Bij een virtuele organisatie is de macht in hoge mate gedecentraliseerd. Zoals uit de figuur blijkt, gedijt een dergelijke organisatie het best indien er sprake is van een hoge mate van complexiteit en veranderlijkheid. Door de daarmee gepaard gaande onzekerheid zou de onderlinge afhankelijkheid groot zijn en zou de behoefte aan controle juist kleiner zijn. Jägers stelt:"*[…] the virtual organization participants do not try to lower this uncertainty through regulation of forms of control (using contracts for example), but rather through the pooling of*

[46] Bijvoorbeeld objectgeoriënteerde programmeertalen, Version Control Systems en Bug Tracking Systems.

knowledge and information." In hoofdstuk 3 bleek dat softwareontwikkeling gepaard gaat met veel complexiteit en veranderlijkheid. De virtuele organisatie lijkt daarom een organisatievorm die goed is toegerust om softwareontwikkeling te coördineren. Overigens, hierbij dient natuurlijk wel bedacht te worden dat, zoals de *contingency theory* stelt, niet een bepaalde organisatievorm universeel zaligmakend is (Wigand, p. 207).

In het onderstaande zal aan de hand van de wezenlijke kenmerken van virtuele organisaties bekeken worden in hoeverre de kwalificatie van OSS-gemeenschappen als zodanig treffend is (Jägers, 1998; 2001). Bovendien zal langs deze weg meer inzicht worden verkregen in organisatiestructuur van OSS-projecten.

- ***Grensoverschrijdend:*** Nauwe samenwerking tussen eenlingen en/of bedrijven in een netwerk biedt meer flexibiliteit waardoor beter op de markt gereageerd kan worden. Veel OSS-projecten hebben producten opgeleverd die de deelnemende individuen en organisatie nooit op eigen kracht hadden kunnen voortbrengen.
- ***Geografisch verspreid:*** Programmeurs van over de hele wereld dragen bij aan OSS. Zowel de ontwikkeling als de distributie van software worden niet of nauwelijks belet door grenzen. Internet speelt hierbij een cruciale rol.
- ***Aanvullende kerncompetenties / kennisdelen:*** Middels het internet bundelen de OSS-programmeurs hun krachten. De hoge mate van modulariteit biedt mogelijkheden tot het complementair inzetten van kerncompetenties. Programmeurs kunnen actief zijn binnen hun eigen specialisme. De openheid van de broncode en de openheid van communicatie via internet waarborgt kennisdeling. Het delen van kennis zorgt voor inventiviteit en continuï teit.Baldwin zegt hierover met betrekking tot het bedrijfsleven: *"to take full advantage of modularity, companies need high skilled, independent-minded employees eager to innovate."* (Baldwin, 1997, p.92)
- ***Wisselende participanten:*** Programmeurs zijn vrijwel nooit formeel gebonden aan een bepaald OSS-project. Velen zullen slechts tijdelijk een bijdrage willen leveren. De twee initiators van The Gimp stopten bijvoorbeeld na hun studententijd volledig met het project. Toch is het project na een periode van stilstand voortgezet. Bovendien zijn OSS-gemeenschappen open organisatievormen. Voortdurend zullen daardoor nieuwe participanten bijdragen kunnen leveren.
- ***Gelijkwaardigheid van participanten:*** Hoewel ieder OSS-project een bepaalde hiërarchie kent, is deze minder vergaand dan bij traditionele organisatievormen. De hiërarchie is

meestal volstrekt functioneel gericht. Zij dient primair om de krachten te laten samenvloeien in een gemeenschappelijk, gestandaardiseerd eindproduct. Er is duidelijk sprake van een conditionerend coördinatiemechanisme. De individuele programmeur kent een zeer grote mate van vrijheid en kan bijvoorbeeld zelf kiezen aan welk onderdeel hij een bijdrage wenst te leveren.

Dynamiek binnen de hiërarchie vindt plaats op basis van prestaties. Dit is kenmerkend voor een zogenaamde meritocratie. Dit principe wordt binnen de gemeenschap meestal als volgt vertolkt: *"Judge people only on the value of what they create not who they are or what credentials they present."* (Weber, 2000, p.26). Door een stijging in de hiërarchie krijgt een programmeur bijvoorbeeld meer zeggenschap over en verantwoordelijkheid voor bepaalde modules. Bovenaan de ladder staat de *'project owner'* of te wel de eigenaar van het project. Zeker gedurende de eerste jaren is de initiator of de groep initiatoren meestal *'project owner'*. Na verloop van tijd kan dit veranderen. Hoewel Torvalds nog steeds een belangrijke contribuant en icoon van Linux is, lijkt het erop dat Cox, als hoofdonderhouder van de stabiele Linux tak, tegenwoordig een vrijwel even belangrijke stem heeft binnen het project.

Aangezien de bijdragen op vrijwillige basis plaatsvinden, wordt de hiërarchie niet (direct) bevestigd door verschillen in financiële beloning. Bovendien is een bepaalde hiërarchie niet bindend. Eenieder is vrij om met het softwareproduct een ander ontwikkelingspad in te slaan buiten de gevestigde hiërarchie om.

- ***Elektronische communicatie:*** Het internet is het voornaamste communicatiemiddel van OSS-programmeurs. Veel communicatie vindt plaats via openbare mailing lists, maar er wordt ook bilateraal gecommuniceerd via email. Via FTP sites is de code van het project te downloaden, en websites dienen ter promotie en ondersteuning van het project.
  Het internet blijkt een zeer geschikt communicatiemiddel voor softwareontwikkeling. Omdat software een informatiegoed is, kan ook het eindproduct via dit kanaal effectief en efficiënt gecommuniceerd worden. De gebruikte diensten, zoals mailing lists en FTP, zijn beproefd. En daarnaast zorgt de lagere mediarijkheid van dit zogenaamde *'Face to interface'* medium ervoor dat hiërarchie en status weinig invloed hebben en de communicatie aldus niet remmen (De Vries & Stegen, 2000). Weber over internet als communicatiemiddel voor OSS-programmeurs: *"Equally important, the Internet scales in a way that physical space does not. Put 25 people in a room, and the communication slows down -- whereas an email list can communicate with 25 people just as quickly as it communicates with 10 -- or 250."* (Weber, 2000, p.17).

- ***Gedeelde visie/droom:*** Hoewel programmeurs enigszins verschillende motivatieredenen kunnen hebben, hebben zij allen ongeveer eenzelfde doel voor ogen, namelijk *"Superieure software ontwikkelen die vrij verkrijgbaar en aanpasbaar is."*
- ***Leiderschap:*** Door de gedecentraliseerde, vrije werkwijze is het kweken van vertrouwen (trust) van enorm belang. De openheid waarin de communicatie plaatsvindt bij OSS-projecten draagt hier sterk toe bij. De regelmaat waarmee de leiders communiceren via de mailing lists is bijvoorbeeld erg bepalend. Torvalds plaatst in een week gemiddeld 100 berichten op de kernel mailing list. De hoge mate van modulariteit zorgt ervoor dat controle en uitvoering relatief eenvoudig gedelegeerd en gedeeld kunnen worden. Ook dit heeft een positief effect op het vertrouwen. Torvalds zegt over leiderschap en vertrouwen: *"It's a chaos that has some external constraints put on it. … the only entity that can really succeed in developing Linux is the entity that is trusted to do the right thing. And as it stands right now, I'm the only person/entity that has that degree of trust."* (Bootnet.com, 1999).

  De leider dient voorwaarden te scheppen en taken uit te voeren waardoor het project uiteindelijk een eindproduct voortbrengt. Het gaat om activiteiten die alleen centraal verricht kunnen worden, zoals zorgen voor een goede modulaire structuur, een snelle release cyclus en de integratie van code. Door deze taken goed uit te voeren, kan voeding worden gegeven aan de motivatieredenen van programmeurs. Reeds bijdragende programmeurs zullen het project blijven waarderen en andere programmeurs zullen worden aangetrokken tot het project. Door brede support kan grotendeels ook afsplitsing, het zogenaamde forking, worden voorkomen. Daarnaast is de mate van openheid van belang om forking te voorkomen. Deze hangt ook samen met het type licentie dat gebruikt wordt. Torvalds hierover: *"The license [the GPL] tends to keep fragmentation down. If there's a new feature, and people actually agree that it's really useful, you just import it into you source tree and you're done."* (McMillan, 2000). Het aantal deelnemers en de mate van openheid zijn bepalend voor de liquiditeit binnen het OSS-project. En des te hoger de liquiditeit, des te stabieler zal het OSS-project zijn (Weber, 2000, p. 30).
- ***Basiswaarden:*** De deelnemers hebben met betrekking tot softwareontwikkeling over het algemeen eenzelfde patroon van waarden. De UNIX-traditie is hiervoor sterk bepalend, en heeft een informele codificatie gekregen in de vorm van de zogenaamde *"Hackers Ethic"*. Principes als *"Keep It Simple, Stupid (KISS)"* en *"clean vs. ugly code"* stammen uit deze traditie, en hebben een diepere, belangrijke betekenis binnen de gemeenschap (Levy,

1984). De basiswaarden zijn vaak zeer pragmatisch van aard. Dit komt bijvoorbeeld ook terug in de metafoor *"Let the code decide"*. (Weber, 2000, p.26).

Uit het bovenstaande blijkt dat organisatievorm van OSS-projecten sterk voldoet aan de kenmerken van de virtuele organisatie. Hoewel deze organisatievorm zich met name door de vooruitgang op technologisch vlak in de loop der jaren verder heeft ontwikkeld, bestaat deze wijze van organiseren in de OSS-wereld al sinds lange tijd. Ten tijde van de eerste schreden van UNIX werd al via elektronische newsgroups samengewerkt. OSS-gemeenschappen kunnen daarom zelfs worden gekwalificeerd als virtual communities *'avant la lettre'*.

6.4 Ontwikkelingsproces

In deze paragraaf zal het ontwikkelingsmodel van OSS nader bekeken worden. Het gaat om de wijze waarop Open Source Software wordt geproduceerd. De volgende karakteristieke elementen van het OSS-ontwikkelingsproces zullen in deze paragraaf aan de orde komen.

- Initiële ontwikkeling in beslotenheid;
- Iteratieve en incrementele ontwikkeling;
- Gedistribueerde, parallelle ontwikkeling op verschillende niveaus;
- Peer-review;
- User-driven innovatie.

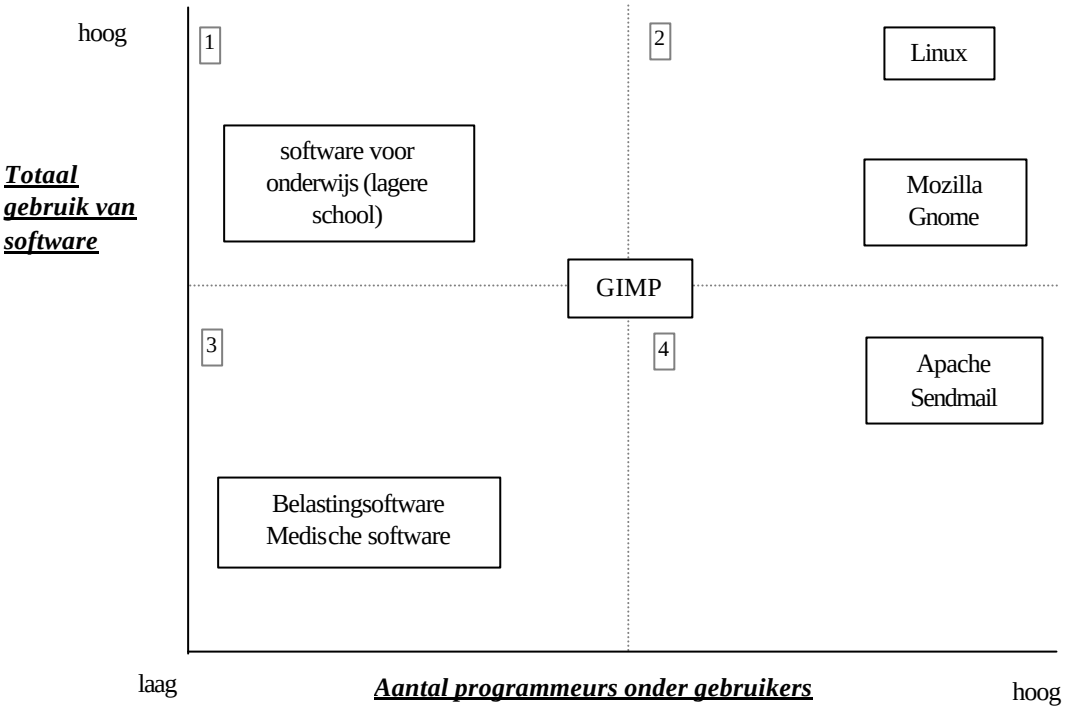
6.4.1 Initiële ontwikkeling in beslotenheid

De geboorte van een Open Source project kent verschillende vormen. Echter, vaak ontwikkelt één programmeur of een kleine groep programmeurs een substantiële hoeveelheid code. Deze hoeveelheid code dient vervolgens als een fundament voor verdere gezamenlijke ontwikkeling. Zoals reeds eerder bleek, zijn zowel Linux als Apache op deze wijze gestart. Het eerste prototype kan geheel vanaf de grond zijn opgebouwd, maar het is zeker niet ongewoon dat dit gebaseerd is op bestaande software. De wortels van Apache liggen bijvoorbeeld in de NSCA httpd server, terwijl Linux aanvankelijk sterk leunde op Minix en bovendien natuurlijk sterk gebaseerd is op UNIX (Raymond, 1998).

Om vervolgens andere programmeurs aan te trekken, oftewel een critical mass te mobiliseren, dient het project voldoende attractief te zijn. Door in te spelen op de motivatie van programmeurs, zoals beschreven in het vorige hoofdstuk, kan een project meer attractief

worden gemaakt. Hiervoor is aanvankelijk de initiële code van groot belang. Raymond merkt hierover het volgende op: *"[…] you need to be able to present a plausible promise. Your program doesn't have to work particularly well. It can be crude, buggy, incomplete, and poorly documented. What it must not fail to do is (a) run, and (b) convince potential co-developers that it can be evolved into something really neat in the foreseeable future."* Van groot belang is dus dat het prototype werkt, maar daarnaast programmeurs uitdaagt tot het leveren van bijdragen. De eerste versie van Netscape's Open Source Browser "Mozilla" werd al snel bekritiseerd door experts omdat de broncode hiaten vertoonde. Het aantal deelnemende, externe programmeurs aan het project was daardoor aanvankelijk lager dan verwacht. Jamie Zawinsky (1999), een voormalig kernontwikkelaar van het Mozilla project, zegt hierover: *"What we released was a large pile of interesting code, but it didn't much resemble something you could actually use."*

Ook de (potentiële) gebruikswaarde van de software voor programmeurs en overige eindgebruikers is van belang voor het creëren van draagvlak (of te wel critical mass) onder programmeurs. De (potentiële) gebruikswaarde echter is inherent aan het type software. Met name voor motivatieredenen als zelfontplooiing en beloning met software is het gebruik van de software een bepalende factor. Het gebruik correleert bijvoorbeeld positief met het eerder genoemde *"signalling incentive"*. Zowel het totale gebruik als het gebruik van de software onder programmeurs is van belang. Een besturingssyteem, zoals Linux, is een noodzakelijk programma voor iedere computer. Daardoor kan dergelijke software in potentie rekenen op een breed algemeen gebruik en bovendien zullen ook veel programmeurs geï nteresseerd zijn. Daarentegen wordt specialistische software, zoals medische software, minder breed gebruikt en zullen onder de gebruikers minder geï nteresseerde programmeurs zijn. In onderstaande figuur zijn een aantal OSS-projecten naar de genoemde maatstaven gepositioneerd. Projecten in het tweede kwartiel kunnen rekenen op een groot draagvlak, terwijl projecten in het derde kwartiel in het algemeen in veel mindere mate op een draagvlak kunnen rekenen (Blecherman, 1999).



Figuur 9: Gebruik van de software en draagvlak voor OSS-project.

Het eerste prototype wordt vrijwel altijd in beslotenheid ontwikkeld. Op deze wijze kan de conceptuele integriteit van het model worden gewaarborgd. Het is moeilijk consensus te bereiken binnen een gedecentraliseerde groep over de initiële architectuur. Dit ondervond men aanvankelijk ook in het Linux 8086 project. Alan Cox (1998) zegt met betrekking tot dit project het volgende: *"It ceased to be a bazaar model and turns into a core team […]. It is an inevitable defensive position in the circumstances."*

Idealiter dient het eerste prototype zodanig ontworpen te zijn dat het inzicht biedt en bovendien uitbreidbaar is. Het ontwerp moet evolutionaire verandering ondersteunen zonder zelf overmatig complex te zijn. Vaak nemen ontwikkelaars aanvankelijk slechts fundamentele beslissingen ten aanzien van het ontwerp. *"Early design should fit on a napkin. […] You need to describe the key modules and data structures – without getting too fancy, as there's a fine line between extensibilty and complexity."* (Murray, 2000). Pas nadat het programma aan gebruik is blootgesteld kunnen gaandeweg op basis van de opgedane ervaringen meer concrete ontwerpbeslissingen genomen worden.

6.4.2 Iteratieve en incrementele ontwikkeling

Software is een dermate complex product dat het vrijwel onmogelijk is vooraf een allesomvattend ontwerp vast te stellen. Het ontwikkelproces moet juist ruimte laten voor continue verandering.

Er bestaan uiteenlopende modellen voor ontwikkeling. Het Water Fall model van Royce, een welbekend klassiek ontwikkelingsmodel, gaat uit van een strikte top-down benadering. De verschillende ontwikkelingsfasen, van planning tot aan introductie, zouden sequentieel moeten worden doorlopen. Boehm lanceerde later het zogenoemde Spiral Model. Dit model omvat een evolutionaire lifecycle waarin via terugkoppeling met de gebruiker middels prototypen het product continu wordt doorontwikkeld. OSS-ontwikkeling lijkt het meest aan te sluiten bij deze evolutionaire methodiek (Wigand, 1997, p.139 e.v.). Door snelle iteraties groeit de software met kleine incrementen. Raymond (1998) omschrijft dit principe als volgt: *"Release early, Release often."*

In OSS-projecten komt een aanpassing aan de code in de regel als volgt tot stand. Een programmeur accepteert een bepaalde taak, bijvoorbeeld het oplossen van een fout of het toevoegen van een verbetering. Vervolgens kan hij een kopie van de broncode ophalen via het internet (bijv. via FTP), en deze naar eigen inzicht aanpassen. De verschillen tussen de originele en de aangepaste code zet hij in een zogenaamde patch. Vervolgens post de programmeur deze patch en een korte beschrijving op de developers mailing list. Andere programmeurs kunnen de aangereikte aanpassingen beoordelen. Indien de patch goedgekeurd wordt door de beheerder van het centale broncode-bestand, de zogenoemd source tree, dan kan deze hiermee worden geï ntegreerd. Uiteindelijk zal de code worden opgenomen in de eerstvolgende officiële versie ofwel release.

Een snelle release cyclus is van belang om de veranderingen te synchroniseren, programmeurs te motiveren en aan te moedigen tot continue feedback. De Linux-kernel is een van de meest aansprekende voorbeelden. De ontwikkelingskernel 2.1 doorging bijvoorbeeld 132 releases vanaf 30 september 1996 tot 22 december 1998. In de onderstaande tabel (Figuur 10) wordt de gehele release geschiedenis van Linux weergegeven tot aan 28 april 2001. Een regelmatige publicatiefrequentie is niet alleen kenmerkend voor Linux maar zeker ook voor veel andere Open Source projecten.

	versienummer	Periode	Aantal releases	Jaar	Aantal releases
Pre-stable tree	0.*.*	17/09/1991 – 06/03/1994	**119**	1991	5
Stable tree	1.0.*	13/03/1994 – 17/04/1994	10	1992	35
	1.2.*	07/03/1995 – 02/08/1995	14	1993	51
	2.0.*	09/06/1996 – 09/01/2001	40	1994	114
	2.2.*	26/01/1999 – 25/03/2001	20	1995	87
	2.4.*	04/01/2001 – 28/04/2001	5	1996	110
	totaal		**89**	1997	63
Development tree	1.1.*	06/04/1994 – 02/03/1995	96	1998	61
	1.3.*	12/06/1995 – 10/05/1996	101	1999	59
	2.1.*	30/09/1996 – 22/12/1998	133	2000	44
	2.3.*	11/05/1999 – 11/03/2000	52	2001 (t/m 28/04)	7
	totaal		**382**		
pre-release / test kernels (vanaf 1.0)	pre		34		
	test		12		
	totaal		**46**		
	totalen opgeteld		**636**		**636**

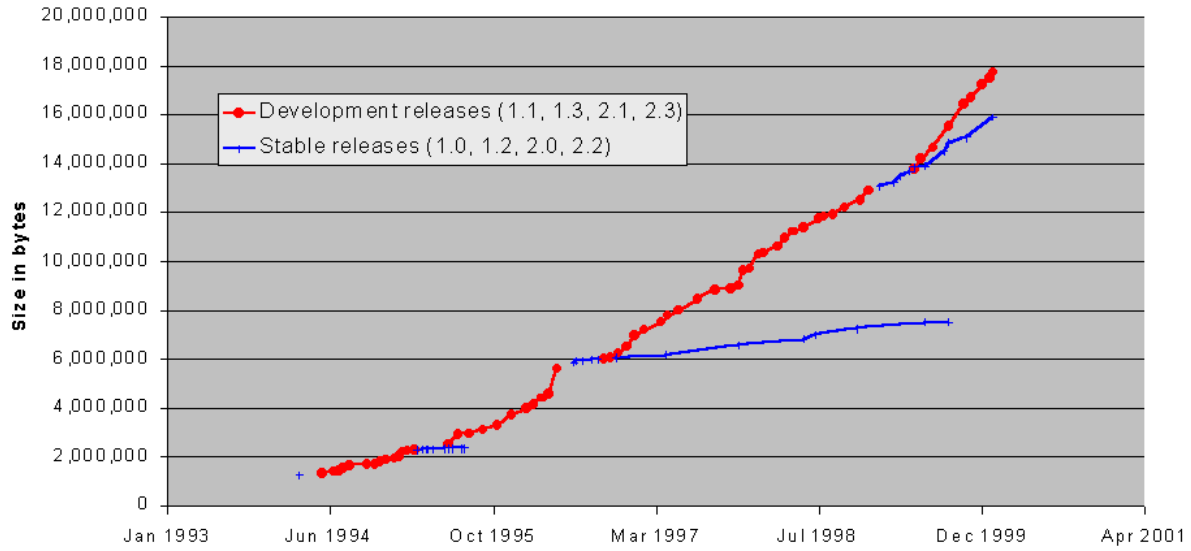
Bron: Gebaseerd op http://www.memalpha.cx/Linux/Kernel/Master.html

Figuur 10: Linux release geschiedenis t/m 28 april 2001.

Door de frequente uitgave van nieuwe versies ontstaat een nauwsluitende feedback loop. Recente toevoegingen kunnen snel geëvalueerd worden en fouten kunnen snel worden opgespoord en hersteld. Toch brengt deze methode ook risico's met zich mee. Door de snelheid en de vele bijdragen kan de code, hoewel werkend, van lage kwaliteit zijn. Dit kan ten koste gaan van de efficiëntie en uiteindelijk van de samenhangendheid en/of ontwerp, waardoor de onderhoudbaarheid verslechterd. Chris Salzburg (1999) zegt hierover met betrekking tot Perl het volgende: *"It really is hard to maintain Perl 5. Considering how many people have had their hands in it; it's not surprising that this is the situation. And you really need indoctrination in all the mysteries and magic structures and so on – before you can really hope to make significant changes to the Perl core without breaking more things than you're adding."* Mede hierdoor komen grootschalige herstructureringsacties op architectuurniveau binnen OSS-projecten voor (Murray, 2000). Larry Wall, de geestesvader van Perl, is momenteel bijvoorbeeld aan het werken aan een totale herstructurering van Perl.

Apache onderging in 1995 een belangrijke herstructurering waarbij verregaande modulariteit werd doorgevoerd. Ook Linux heeft ingrijpende herstructureringen ondergaan. Reeds in de begindagen werd Linux, dat aanvankelijk alleen geschikt was voor de Intel 386 processor, ten

faveure van de portabiliteit grotendeels herschreven (Torvalds, 1999).[47] En bij versie 2.0 werden in 1996 zogenaamde kernel-modules geï ntroduceerd. De schaalbaarheid van Linux heeft de verwachtingen van vele experts, onder wie Godfrey en Tu (2000), overtroffen. De blijvend hoge groeiratio kan onder meer worden afgeleid uit de onderstaande figuur.



Figuur 11: Groei in bytes van de gecomprimeerde (tar-) file van de volledige Linux-kernel broncode (Godfrey, 2000).[48]

Dit succes wordt met name toegeschreven aan de hoge mate van modulariteit van Linux. Met name device drivers, noodzakelijk voor de aansturing van randapparatuur, zorgen de laatste jaren voor voortzetting van de sterke groei. Echter, de groei van het hart van de Linux-kernel is grotendeels gestabiliseerd (Tuomi, 2001).

6.4.3 Gedistribueerde, parallelle ontwikkeling op verschillende niveaus

Bij Open Source projecten is er sprake van parallelle ontwikkeling op verschillende niveaus. De verschillende activiteiten, zoals ontwerpen, bouwen en testen, vinden tegelijkertijd plaats. Er bestaan geen afzonderlijke, sequentiële fases. Deelnemers kunnen op vele fronten tegelijkertijd hun bijdragen leveren. Sommigen schrijven broncode, terwijl anderen fouten corrigeren of discussiëren over het toevoegen van nieuwe onderdelen. Ook op het hoogste niveau bestaan er binnen bepaalde OSS-projecten parallelle ontwikkelmethoden. Het Linux-project kent bijvoorbeeld een ontwikkelingstak (development tree) en een stabiele tak (stable

[47] Momenteel ondersteunt Linux minstens 14 processorfamilies (http://www.kernel.org).

tree). Om de veranderingen te synchroniseren wordt op bepaalde tijdstippen een overkoepelende cyclus beëindigd en worden de patches geïntegreerd in een nieuwe release.

OSS-projecten, zoals Linux en Apache, tonen aan dat parallelle, gedistribueerde ontwikkeling zeer succesvol kan zijn. De kracht lijkt het aanboren van een wereldwijde bron met geïnteresseerde programmeurs. Echter, Brooks stelt juist in zijn klassieke boek "The Mythical Man Month" dat het toevoegen van extra programmeurs aan een project niet perse leidt tot snellere afronding (Brooks, 1995). Aan de ene kant kunnen twee programmeurs twee keer zoveel software produceren als één programmeur, maar het toevoegen van programmeurs zou ook leiden tot meer koppelingen binnen de software en tussen de programmeurs. De extra koppelingen, ofwel interfaces, zorgen op het vlak van zowel techniek als organisatie voor een extra communicatielast. Deze communicatielast zou exponentieel toenemen met het aantal onderlinge relaties.

Brooks' stellingen lijken aldus in strijd met OSS-ontwikkeling. Echter, twee factoren laten zien dat het dilemma slechts schijnbaar is. Allereerst geldt Brooks' stelling alleen voor het ontwikkelen van nieuwe softwarecode, en niet voor het vinden van fouten cq. bugs. Het opsporen van fouten tijdens dagelijks gebruik vraagt namelijk weinig afstemming op zowel technisch als organisatorisch vlak. Terwijl het opsporen (en vervolgens oplossen) van fouten juist een zeer belangrijke en kostbaar deel vormt van de ontwikkelingscyclus van een softwareprogramma (Koning, 1999).

Ten tweede kan modulariteit via open gedefinieerde koppelingen zorgen voor opdeling van een softwareprogramma in relatief onafhankelijke deelprojecten. Daardoor kunnen kleine programmeerteams in hoge mate autonoom werkzaam zijn. Zawinsky, voormalig Mozilla kernprogrammeur, hierover: *"Most of the larger open source projects are also fairly modular, meaning that they are really dozens of different, smaller projects. So when you claim that there are ten zillion people working on the Gnome project, you're lumping together a lot of people who never need to talk to each other, and thus, aren't getting in each other's way."* (Jones, 2000). Succesvolle OSS-projecten voldoen aan beide kenmerken; een groot aantal mensen identificeert fouten, terwijl de structuur van de code in hoge mate modulair is, zodat kleine programmeerteams relatief onafhankelijk kunnen opereren.

[48] Opgemerkt dient te worden dat er verschillende maatstaven voor groei worden gehanteerd. Echter, een strikt neutrale maatstaf voor groei bestaat niet.

Modulariteit kent echter grenzen. Aan modulariteit zijn namelijk (opportuniteits-)kosten verbonden. Allereerst zijn er kosten voor het ontwerpen van een geschikte modulaire opbouw met bijbehorende koppelingen. De kosten hiervan zullen toenemen naarmate het niveau van beoogde modularisering meer detaillistisch en/of meer fundamenteel wordt. Met andere woorden: een project is niet tot in het oneindige op te delen in modules, en bovendien zullen bepaalde wezenlijke bestanddelen vrijwel onmogelijk ontleed kunnen worden. Daardoor zal slechts een beperkt aantal kleine teams kunnen werken aan de modules en zal een klein "chirurgisch" team zorg dragen voor het fundament van de software. Zawinsky erkent dit: *"In most open source projects there is a small group who do the majority of the work, and the other contributors are definitely at a secondary level, meaning that they don't behave as bottlenecks."* (Jones, 2000). Uit onderzoek blijkt inderdaad dat bij veel OSS-projecten slechts een klein deel van het aantal deelnemende programmeurs verantwoordelijk is voor het overgrote deel van de broncode (Mockus, 2000). Ghosh becijferde in zijn onderzoek bijvoorbeeld dat 10% van de programmeurs ongeveer 72% van de totale hoeveelheid broncode had geschreven (Ghosh, 2000). Evenwel wil dat niet zeggen dat het werk dat door anderen is gedaan onbelangrijk is. Dave Hitz, een van de programmeurs die hielp UNIX te herschrijven zodat het bevrijd was van AT&T auteursrechten, zegt hierover: *"It think it's rarely the case that you get people who make a broad base of source code their life. There are just a whole bunch of people who are dilettantes. The message is, 'Don't underestimate the dilettantes.'"* (Wayner, 2000, p.112).

Toch, moet worden gezegd dat op het terrein van modulariteit mede door nieuwe technologie nog steeds progressie wordt geboekt, waardoor de hoogte van de genoemde kosten in toekomst eventueel omlaag gebracht kan worden. Daardoor zal parallelle, gedistribueerde ontwikkeling nog eerder rendabel zijn en nog verder doorgevoerd kunnen worden.

Daarnaast zijn er kosten, omdat door modulariteit op het gebied van koppelingen sprake is van een technologische status quo voor een bepaalde tijd. Deze pre-gedefinieerde koppelingen, oftewel (interne) standaarden, kunnen een beperking vormen bij het doorontwikkelen. De koppelingen verlagen aldus de technische en organisatorische complexiteit ten koste van de prestatie en vooruitgang. Beide bovengenoemde kosten dienen idealiter afgewogen te worden tegen de baten van de complexiteitsreductie alvorens over te gaan tot modularisatie.

De uitkomsten van de parallel ontwikkelingsprocessen dienen uiteindelijk geïntegreerd te worden. Deze integratie van de code kan worden beschouwd als *'overhead'* en wordt uitgevoerd door de projectleiding. Paul Vixie: *"Integration of an open-source project usually involves writing some manpages, making sure that it builds on every kind of system the developer has acces to, cleaning up the Makefile to remove the random hair that creeps in during the implementation phase, writing a README, making a tarball, putting it up for anonymous FTP somewhere, and posting a note to some mailinglist or newsgroup where interested users can find it."*

Bepaalde procedures worden gevolgd om de integratie van de toevoegingen aan de code en de aanpassingen daarvan te vergemakkelijken. Voorafgaand aan een release worden bijvoorbeeld zogenaamde feature freezes en code freezes afgekondigd. Gedurende een feature freeze wordt geen nieuwe functionaliteit meer toegevoegd aan de code, maar zijn bugfixes nog wel toegestaan. Een code freeze is verdergaand dan een feature freeze. Er worden geen aanpassingen meer verricht aan de code, behalve indien er sprake is van zeer ernstige fouten. De ratio achter een freeze is het creëren van een groeiende buffer rond de integratie welke uiteindelijk uitmondt in een release.

Veel Open Source projecten maken gebruik van een version control system om de integratiekosten te verminderen. Een dergelijk systeem zorgt ervoor dat het in kaart houden van veranderingen en het samenstellen van een release eenvoudiger wordt. Veel OSS-projecten, zoals Apache, GIMP en Samba, maken gebruik van CVS, dat zelf ook Open Source Software is.[49] Met dit systeem kan over het internet gewerkt worden. Overigens wordt er bij Linux opvallenderwijs geen gebruik gemaakt van een version control system. Torvalds acht de functionaliteit van dergelijke systemen (nu nog) te beperkt, en de integratie van code vindt daarom grotendeels handmatig plaats.

6.4.4 Peer review

Net als in de wetenschappelijke wereld wordt in de Open Source wereld de zogenaamde peer review, oftewel wederzijdse inspectie door collega's, beschouwd als een wezenlijk methode om kennis naar een hoger niveau te tillen.

[49] Zie http://www.cvshome.org

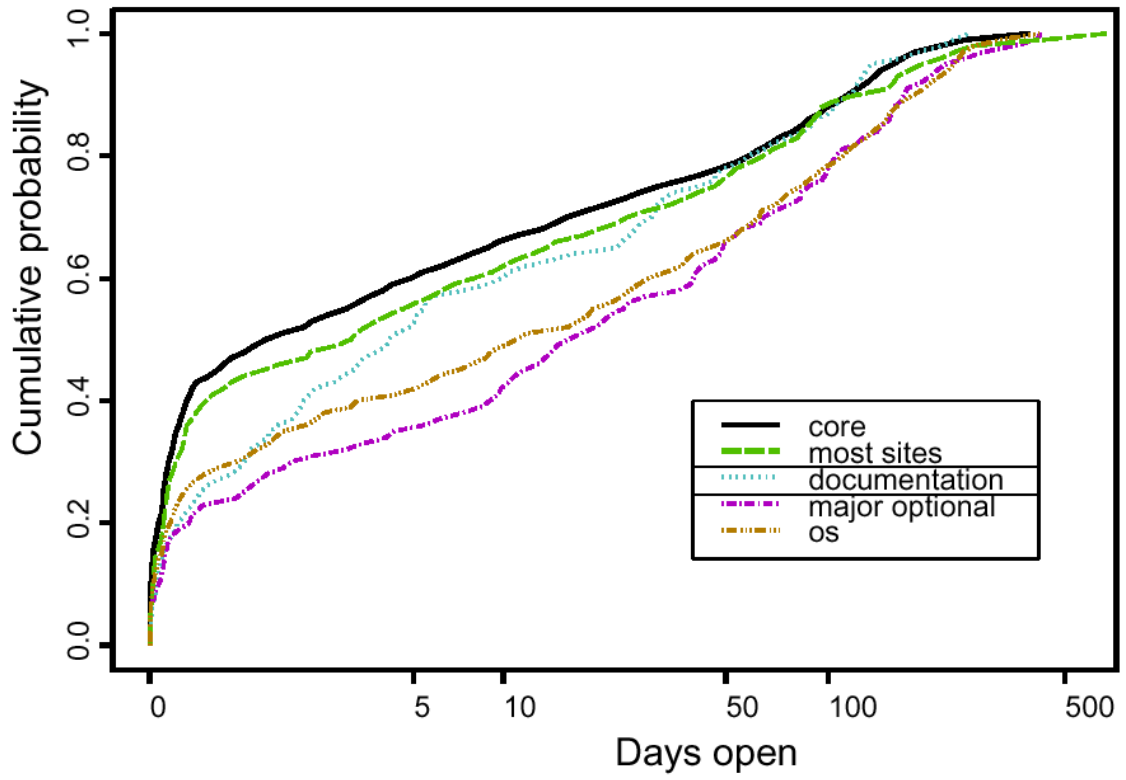
Peer review is impliciet aan OSS-projecten. De broncode is namelijk beschikbaar voor iedereen, en bovendien vindt de communicatie vaak publiekelijk plaats. OSS-projecten leunen vaak volledig op peer review bij het verwijderen van fouten. De meeste OSS-projecten kennen geen vastomlijnde testprogramma's. Toch staat veel (succesvolle) OSS-software bekend om haar betrouwbaarheid. Uit een vergelijking tussen Apache webserversoftware en commerciële software kwam bijvoorbeeld de hypothese naar voren dat OSS in het algemeen een lagere dichtheid aan fouten kent dan commerciële software (Mockus, 2000, p.10).

Raymond (1998a) noemt de kracht achter Open Source Software "Linus's law" welke zegt dat *"Given enough eyeballs, all bugs are shallow."* De gedachte hierachter is dat meer gebruikers de software op meer uiteenlopende wijzen op de proef zullen stellen, waardoor fouten eerder aan het licht komen. Bovendien hebben OSS-gebruikers ook nog eens directe toegang tot de broncode, waardoor ook langs deze weg fouten opgespeurd kunnen worden. Paul Vixie (1999) hierover: *"The essence of testing is lack of rigor. What software engineering is looking for from its field testers is patterns of use which are inherently unpredictable at the time the system is being designed and built – in other words, real world experiences of real users. Unfunded open-source projects are simply unbeatable in this area."*

Bovendien is de deelnemende groep zelfselectief, waardoor de reviewers in hoge mate gemotiveerd en getalenteerd zijn. Singleton zegt hierover het volgende: *"The talent pool is very deep when you can draw from developers around the world, not just from a single city or a single company. A globally distributed team can include developers that are not only talented, but also interested in and knowledgeable about the task at hand."* (Murray, 2000).

Dat het bovenbeschreven principe in de informatiemanagement literatuur niet onbekend is, blijkt wel uit het volgende citaat van Corbató (1995): *"Bij een groot project is één van de beste investeringen die je kunt doen, een investering in de 'onderlinge educatie' van het team, zodat vrijwel iedereen meer weet dan hij moet weten. Het is duidelijk dat met educationele redundantie het team veel veerkrachtiger is bij onverwachte tragedie of het onverwachte vertrek van personen. Maar bovendien kan het toegenomen bewustzijn van de teamleden helpen bij een herkenning in een vroege fase van globale fouten of systeem fouten. Het is in dit geval werkelijk zo dat 'twee meer weten dan één'."* De schaal waarop het principe wordt toegepast in de OSS-wereld kent echter in de bedrijfswereld geen gelijke.

Door de continue peer-review worden fouten in de software vaak snel verholpen, nadat deze aan het licht zijn gekomen. Torvalds hierover: *"Bugs in Linux are minimal and fixed very quickly. As an example, a networking bug that allowed you to send illegal packets was corrected within four hours. Most commercial UNIXES had fixes within a week or two."* (bootNet.com, 1999). Natuurlijk is de snelheid waarmee een bugfix plaatsvindt afhankelijk van de ernst van de fout. De volgende figuur laat de proportie foutoplossingen zien binnen een gegeven aantal dagen voor het Apache webserver project. Vijftig procent van de fouten worden binnen een dag opgelost. Fouten die gerelateerd zijn aan de kern (core) worden relatief het snelst opgelost.



Figuur 12: Snelheid van foutoplossing bij het Apache-project (Mockus e.a., 2000, p.271).

De peer-review dwingt de programmeur ook tot simpliciteit en elegantie. De code dient namelijk inzichtelijk te zijn voor collega-programmeurs. *"[Arcangnelli] made substantial improvements, then branched out, but tended to do some pretty sloppy things – to the point where Linus [Torvalds] said "go away." [Arcangnelli] refused to go away, and eventually had major changes to the kerenel accepted. All Linus did was enforce coding standards."* (Kuwabara, 2000)

6.4.5 User driven ontwikkeling

Software is een gebruiksgoed. Eén van de meest uitdagende aspecten voor ontwikkelaars is de software optimaal aan te laten sluiten op de eisen van de gebruiker. Deze eisen kunnen van gebruiker tot gebruiker verschillen en bovendien kunnen de eisen aan verandering onderhevig zijn. Participatie van de gebruiker is een van de pijlers van OSS-ontwikkeling. De programmeurs zijn namelijk tevens gebruikers van de software. Kelly vergelijkt the bottom-up benadering van OSS-ontwikkeling met de commerciële ontwikkelingsmethoden als volgt: *"If top-down management asked for a stupid feature that would make the system worse, developers would think "If they want to pay me…. The developer gets paid to do what is asked, not what is best."* (Kuwabara, 2000).

De keerzijde van de bovenbeschreven empowerment is dat het de programmeurs kan verleiden tot het maken van slecht doordachte toevoegingen waardoor de conceptuele integriteit van de code op het spel staat. In OSS-projecten vervult de gelaagde peer-review een filterfunctie om een dergelijke aftakeling te voorkomen. Torvalds hierover met betrekking tot de Linux-kernel: *"It's just a basic fact of software engineering that it's a lot easier to add features than it is to remove them! Adding new features never breaks old programs, unless you have a bug – and then the bug may be so subtle that you never notice. Eventually it will happen to Linux too. The only thing you can do is to make that process as slow as you can make it."* (Goodman, 1999).

Doordat veel deelnemers met name technisch georiënteerd zijn, is Open Source Software in de regel minder gebruiksvriendelijk dan commerciële software. De nadruk ligt meestal op de technische efficiëntie. Open Source Software is daardoor vaak meer gericht op infrastructuur. Traditioneel gezien houden relatief veel OSS-projecten zich bezig met besturingssystemen en netwerkapplicaties. Echter, met name de laatste jaren vindt er op dit vlak een verschuiving plaats. Steeds meer OSS-projecten richten zich op gebruiksvriendelijke applicaties. De modulaire opbouw van veel Open Source Software biedt natuurlijk ook uitstekende mogelijkheden voor derden om de software te stroomlijnen voor eindgebruikers. De grafische schillen voor besturingssystemen, die GNOME en KDE ontwikkelen, zijn hiervan een voorbeeld.

Hoewel de gebruiksvriendelijkheid toeneemt, speelt Open Source Software nauwelijks een rol op de PC-markt. Deze markt wordt vrijwel volledig beheerst door Microsoft. Torvalds

hierover: *"People get Windows because it's still the no-brainer. […] What Linux needs to do is become a no-brainer decision. [..] The no-brainer point is five or ten years in the future."* (McMillan, 2000).

6.5 Afronding

Modulariteit kan worden beschouwd als een belangrijk middel om invulling te geven aan een technische architectuur van software. Open Source Software is van oudsher in hoge mate modulair gestructureerd. Dit biedt niet alleen voordelen op technisch vlak, maar ook op het vlak van de organisatorische inrichting en ontwikkelingsmethode van OSS-projecten. Hoewel het internet en de modulariteit de coördinatiekosten enorm reduceert, blijft een bepaalde mate van hiërarchie en organisatie wel noodzakelijk. Bepaalde taken kunnen namelijk alleen centraal worden uitgevoerd. De vorm waarin de individuele krachten worden gebundeld kan worden gekarakteriseerd als een virtuele organisatie. Vertrouwen en gemeenschappelijk uitgangspunten en doelstellingen vormen belangrijke bindingsfactoren. Het ontwikkelingsproces vindt in hoge mate parallel en gedistribueerd plaats. Via frequente iteraties groeit de software met kleine incrementen. Door openheid in combinatie met modulariteit zijn zowel de organisatie als het ontwikkelingsproces zeer flexibel.

Het internet en aanverwante technologie spelen bij de organisatie en het ontwikkelingsproces van OSS een wezenlijke faciliterende rol. Brooks stelde in 1995 terugkijkend op zijn boek "The Mythical Man Month" dat *"the microcomputer revolution has changed how everybody builds software."* (Brooks, 1995, p.281). De afgelopen jaren lijkt het internet belangrijke veranderingen met zich mee te brengen voor softwareontwikkeling. OSS-ontwikkeling lijkt als een van de eerste het potentieel dat geboden wordt door internet succesvol aan te grijpen.

Overigens kunnen bepaalde methoden die kenmerkend zijn voor OSS-ontwikkeling ook (ten dele) worden ingezet door het bedrijfsleven. Tim O'Reilly, uitgever van boeken over OSS, stelt dat *"[…] it is possible to apply open source collaborative principles inside a large company, even without the intention to release the resulting software to the outside world."* (O'Reilly, 2000). Echter, de meeste voordelen zullen door de volstrekte openheid naar hun aard altijd meer naar voren komen bij OSS-ontwikkeling. Een commercieel bedrijf kan bijvoorbeeld nooit in de mate waarin OSS-projecten dat doen, putten uit het wereldwijde potentieel van *vrijwillige* programmeurs.

In het volgende hoofdstuk zal bekeken worden welke voor- en nadelen de beschreven organisatie en ontwikkelingsmethode voor het eindproduct, namelijk Open Source Software, met zich mee brengen.

Hoofdstuk 7: Sterkten en zwaktes van Open Source Software

"People think just because it is open-source, the result is going to be automatically better. Not true. You have to lead it in the right directions to succeed. Open source is not the answer to world hunger." – Linus Torvalds (Scannel, 1999).

7.1 Inleiding

In dit hoofdstuk zullen de sterke en zwakke punten van Open Source Software aan bod komen. Deze punten kunnen per softwareproduct verschillen, en er dient dan ook gewaakt te worden voor generalisatie. Bovendien heeft de gebruiker een bepalende invloed op de realisatie van de voor- en nadelen in de praktijk. Echter, Open Source Software lijkt (op zijn minst in potentie) op bepaalde vlakken te verschillen van commerciële software. De plus- en minpunten hangen nauw samen met de achtergrond, de licentie, de organisatie en het ontwikkelingsproces, zoals besproken in de voorgaande hoofdstukken.

7.2 Kwaliteit

De technische kwaliteit van software omvat meerdere aspecten, maar met name stabiliteit en efficiëntie. Door experts wordt vaak aangedragen dat OSS meer kwaliteit biedt dan vergelijkbare commerciële software (Breedveld e.a., 1999, p.88). Ook uit vergelijkingstests komen populaire OSS-pakketten, zoals Linux en Apache, vaak als kwalitatief hoogwaardig naar voren. Het volgende citaat uit een Duitse test bevestigt dit:*"Das Duel [Windows NT +] IIS kontra [Linux +] Apache geht ganz klar zu Günsten des Open-Source-Projektes aus: Die getesteten IIS-server hatten etwa fünfmal so lange Ausfallzeiten wie Server mit Apache."* (Bundesministerium für Wirtschaft und Technologie, 2001, p.21). Ook een vergelijkend onderzoek tussen Linux en Windows NT door Bloor Research in 1999 toonde aan dat Open Source Software van hoge technische kwaliteit kan zijn. *"In the space of one year, Linux crashed once because of a hardware fault (disk problems), which took 4 hours to fix, giving it a measured availability of 99.95 percent. Windows NT crashed 68 times, caused by hardware problems (disk), memory (26 times), file management (8 times), and a number of odd problems (33 times). All this took 65 hours to fix, giving an availability of 99.26 percent. The winner here is clearly Linux."* (Godden, 2000).

De kwaliteit van software kan in theorie bij OSS door een grotere groep programmeurs worden gewaarborgd dan bij gesloten broncode software. In het geval van OSS kan namelijk

de hele wereld de broncode bekijken, waardoor fouten in de code eerder ontdekt en gerepareerd kunnen worden. De OSS zou daardoor kwalitatief hoogwaardiger zijn. Echter, het probleemoplossend vermogen valt of staat natuurlijk met de hoeveelheid beschikbare middelen. Alleen wanneer voldoende, bekwame programmeurs de software intensief bekijken, zal de kwaliteit van hoog niveau zijn.

Zoals eerder opgemerkt, is een verregaande modulariteit van de software een belangrijk vereiste voor de coördinatie van een OSS-project. Deze modulariteit kan ook voordelen bieden met betrekking tot de kwaliteit van de software. Als de processen namelijk voor een groot deel afzonderlijk worden uitgevoerd, zal een fout in een bepaald gedeelte meestal niet direct leiden tot problemen in andere delen van het programma (Bundesministerium für Wirtschaft und Technologie, 2001, p.20).

Ook de relatieve volwassenheid van bepaalde OSS levert kwaliteitsvoordelen op. Veel van de huidige OSS kent al een lang verleden of is gebaseerd op oudere software. LATEX bestaat bijvoorbeeld sinds 1985 en Linux is gebaseerd op UNIX, dat al in 1969 het levenslicht zag. Ook het hergebruik van bestaande software middels het iteratieve ontwikkelingsproces zorgt ervoor dat OSS in ieder geval vaak zeer beproefde delen bevat.

7.3 Veiligheid

Veiligheid is nauw verwant aan kwaliteit. Ook de veiligheid van OSS wordt door experts vaak als hoog ervaren. Voor commerciële software geldt zogenaamde *"safety through obscurity"* oftewel veiligheid door onbekendheid. Doordat de achterliggende code geheim wordt gehouden, zou de software veiliger zijn. Dat betekent wel dat een gebruiker voor de veiligheid van de software volledig dient te vertrouwen op de producent. Evenwel, deze laatste kan natuurlijk veiligheidsfouten over het hoofd zien. De webserversoftware van Microsoft staat bijvoorbeeld bekend om haar vele veiligheidsfouten, waardoor bijvoorbeeld onlangs het zogenaamde Code Red virus kon toeslaan. Bovendien zijn er gevallen bekend van producenten die zelfs bewust 'achterdeuren' in software open lieten. In 2000 werd er bijvoorbeeld in de "FrontPage 98 extensions" van Microsoft's webserversoftware een beveiligingslek aangetroffen dat programmeurs moedwillig hadden geïmplementeerd (Wilcox, 2000a). De geslotenheid van de broncode kan aldus zorgen voor een vals gevoel van veiligheid. Volgens Raymond (1999) biedt de gesloten broncode eerder nadelen dan voordelen: *"Closed source leads not to true security but to a false sense of security. You don't*

know what's in there, you can't verify it, you can't check the assumptions or honesty of the people who wrote it."

In crypthografische kringen geldt het volgende adagium*: "The security of an algorithm should not depend on its secrecy"* (Walker Whitlock, 2001). In de OSS-wereld trekt men deze lijn door naar software in het algemeen. Bij OSS is de broncode voor eenieder beschikbaar. Daardoor kan in potentie iedereen veiligheidslekken opsporen en oplossen.

OSS biedt over het algemeen ook een hoger niveau van dynamische veiligheid dan commerciële software. Fouten worden relatief snel hersteld en patches worden snel beschikbaar gesteld via het internet. Bij commerciële software duurt dit vaak langer. Volgens een rapport uit 2000 duurde het bijvoorbeeld gemiddeld 11 dagen voordat Red Hat een fout in haar software had opgelost, terwijl Microsoft en Sun de fouten in hun software in respectievelijk gemiddeld 16 en 89 dagen hadden verholpen (Beale e.a., 2000).

Softwarebedrijven bieden de patches bovendien vaak gebundeld aan, namelijk in de vorm van "Service Packs". Het kan daardoor langer duren voordat een oplossing beschikbaar is. Daarnaast zijn softwarebedrijven niet gebaat bij veel publiciteit rondom een softwarefout. Dit geldt zeker op het moment dat het bedrijf nog geen oplossing voorhanden heeft. Microsoft heeft onlangs bijvoorbeeld een beroep gedaan op veiligheidsdeskundigen en hackers om gevonden veiligheidsfouten niet publiekelijk te onthullen (Webwereld, 2001c). Een en ander kan leiden tot onopgeloste fouten waarvan kwaadwillenden wel op de hoogte kunnen zijn en de gebruiker niet.

De openheid van de broncode *perse* garandeert echter niet dat de software een hoog veiligheidsniveau heeft. Er kan ook sprake zijn van grootschalig free-rider gedrag. Bij de GNU mailing list manager Mailman werden onlangs verschillende ernstige beveiligingsrisico's ontdekt. Volgens de oorspronkelijke auteur van het programma, John Viega, werd dit veroorzaakt door het feit dat veel gebruikers er vanuit waren gegaan dat anderen de code reeds grondig hadden doorgelicht (Walker Whitlock, 2001). De relatie tussen veiligheid en Open Source Software omschrijft hij als volgt: *"Open-sourcing your software can help improve your security, but there are associated risks to understand and consider if you're going to see real benefits. Open-sourcing your software is not a cure-all; it's an*

effective supplement to solid development practices and pre-release source-level audits." (Viega, 1999)

7.4 Flexibiliteit

De compatibiliteit en portabiliteit van Open Source Software is in het algemeen zeer hoog. Dit is een gevolg van de hoge mate van modulariteit en openheid. Apache is bijvoorbeeld beschikbaar voor uiteenlopende besturingssystemen[50] en Linux ondersteunt bijvoorbeeld 14 verschillende processorfamilies. De schaalbaarheid van Linux is daardoor enorm; "[Linux] could run on a high-end server as well as on a small machine, even down to a handheld." (Godden, 2000). Daarentegen ondersteunt MS-Windows bijvoorbeeld vrijwel alleen Intel-architecturen en zijn Microsoft programma's vrijwel alleen geschikt voor Windows.

De uiteinden van Open Source Software zijn altijd open, waardoor uitwisseling van data tussen verschillende applicaties relatief eenvoudig tot stand kan worden gebracht. Bovendien ondersteunt Open Source Software voor vrijwel altijd *de jure* standaarden en,voor zover mogelijk, ook *de facto* 'proprietary' standaarden. Linux ondersteunt bijvoorbeeld ook Microsoft-bestandssystemen, zoals FAT16/32 en NTFS. Het risico van lock-in is bij Open Source Software door de enorme flexibiliteit vrijwel afwezig.

Bovendien kan een OSS-programma aangepast worden aan de veranderende wensen van eindgebruikers. De levensduur van de software kan hierdoor verlengd worden. Commerciële software heeft vaak een relatief korte levensduur, doordat deze door de voortschrijdende technologie achterhaald wordt. Softwarefabrikanten proberen dit proces middels *'planned obsolence'* vaak te versnellen door bijvoorbeeld doelbewust incompatibiliteiten te veroorzaken (Katz/Shapiro, p.7-8). Daarentegen heeft Open Source Software in beginsel een oneindige levensduur. Door de openheid van de broncode kan de software altijd doorontwikkeld blijven worden.

7.5 Ondersteuning

Het gebruik van software gaat altijd gepaard met problemen waardoor ondersteuning van groot belang is. Verschillende bronnen van ondersteuning kunnen worden onderscheiden.

[50] De Apache webserver is beschikbaar voor onder meer Windows NT/9x, Netware 5.x, OS/2, en de meeste versies van Unix.

- Documentatie die bij het softwarepakket hoort, zoals bijvoorbeeld manuals;
  Hoewel deze documentatie voor Open Source Software in de regel wat minder gepolijst is, zijn verbeteringen reeds duidelijk merkbaar. Met name distributeurs, zoals Red Hat en SuSE, laten de software vergezeld gaan van prima documentatie.
- Documentatie van derde partijen, zoals bijvoorbeeld boeken en tijdschriften;
  Er zijn minder boeken en tijdschriften beschikbaar over Open Source Software dan over commerciële software. Een uitzondering vormt de uitgever O'Reilly, wiens boeken veelal op OSS gericht zijn.[51] Ook is er duidelijk een tendens dat er steeds meer boeken en tijdschriften aandacht besteden aan OSS. Ook tijdschriften, die zich traditioneel op de Windows-thuisgebruiker richten, ruimen bijvoorbeeld meer plaats in voor Open Source Software.
- Internet, bijvoorbeeld nieuwsgroepen en websites met handleidingen, tips en Frequently Asked Questions (FAQs);
  Op dit vlak is de ondersteuning van OSS traditioneel sterk. Het internet vormt het centrale medium voor communicatie-uitwisseling door OSS-progammeurs en gebruikers. Een zoekopdracht met de term "Linux" levert bij de zoekmachine Google bijvoorbeeld circa 38,300,000 treffers op. Dit overtreft zelfs een zoekopdracht met de term "Windows", welke 'slechts' circa 33,200,000 treffers oplevert.[52] Het Linux Documentation Project is bijvoorbeeld een open project met als doel het creëren van vrij beschikbare, kwalitatief hoogwaardige documentatie over Linux.
- Ondersteunende bedrijven, zoals de softwareverkoper of een consultant.
  Steeds meer bedrijven bieden ondersteuning aan voor OSS. De distributeurs zijn zeer actief op deze markt. Echter, in toenemende mate ondersteunt het traditionele computerbedrijfsleven OSS. Men richt zich natuurlijk met name op de meer succesvolle applicaties, waardoor bepaalde Open Source Software geen ondersteuning kent vanuit het bedrijfsleven. Doordat de software niet toebehoort aan een partij, is er geen risico voor lock-in door het ondersteunende bedrijf.
  Certificatie van dienstenleveranciers op het gebied van OSS is nog volop in ontwikkeling. Doordat de dienstverleners aan bepaalde eisen moeten voldoen, kunnen garanties worden gegeven aan de gebruikers ten aanzien van de vaardigheden van de dienstverleners. In de commerciële softwarewereld is certificering zeer gebruikelijk. Een voorbeeld hiervan is het Microsoft Certified Systems Engineer (MCSE) programma. Op internationaal niveau

[51] http://www.oreilly.com/
[52] http://www.google.com, datum: 19 oktober 2001.

poogt het Linux Professional Institute (LPI) een standaardisatie voor Linux certificering te creëren.[53] Dit initiatief wordt mede ondersteund door IBM. Daarnaast bieden de verschillende distributeurs, zoals Red Hat en SuSE, certificeringsprogramma's aan (Breeveld, 1999, p.74/75; p.91).

Naast bovengenoemde bronnen, heeft Open Source Software uiteraard nog een bron van ondersteuning, namelijk de broncode zelf. Toch zal deze door veel gebruikers niet direct worden gebruikt. Desalniettemin kan de broncode indirect, bijvoorbeeld als een consultant wordt ingeschakeld, van grote ondersteunende waarde zijn.

7.6 Vooruitgang en planning

OSS-projecten kunnen niet garanderen dat bepaalde stappen worden gezet in een bepaalde tijdspanne. Deelname aan het project is in beginsel vrijblijvend. Het vertrek van de initiators van the Gimp, zorgde er bijvoorbeeld voor dat het project gedurende ongeveer een jaar stil kwam te liggen. Echter, ook commerciële softwareproducenten kunnen niet altijd voldoen aan de verwachtingen. Productlanceringen worden geregeld uitgesteld. Bovendien bieden arbeidscontracten ook maar een beperkte mate van zekerheid. Softwareprogrammeurs kunnen namelijk overstappen naar een concurrerend bedrijf. In 1997 spande Borland bijvoorbeeld een rechtszaak aan tegen Microsoft, nadat deze laatste 34 werknemers had overgenomen van Borland (Ricciuti, 1997).

Het gebrek aan planning bij OSS kan ook als voordeel uitgelegd worden. Commerciële software wordt door de druk van deadlines soms, voordat deze grondig getest is, op de markt gebracht. Daardoor kennen deze vroege versies vaak veel fouten. Bij OSS-ontwikkeling wordt vaak uitgegaan van adagium *"it will be ready when it's ready"*. Alan Cox hierover: *"It's a target. If you don't have a target you never finish. If you do have a target you miss it sometimes. The good thing is we are missing by a smaller margin each time."* (Cox, 2000).

7.7 Gebruiksvriendelijkheid

Door haar afkomst is Open Source Software traditioneel gezien minder gebruiksvriendelijk dan commerciële software. Met name de laatste jaren wordt op dit terrein veel progressie geboekt. Distributies gaan vaak vergezeld van visuele installatiescripts en grafische interfaces,

[53] http://ww.lpi.org

zoals KDE en Gnome, die het gebruik van de software eenvoudiger maken. Een onlangs gepubliceerde test van drie Linux-distributies eindigde als volgt: *"Tot slot kunnen we stellen dat alle drie de distributies een gemakkelijk te installeren en goed voorgeconfigureerd systeem met fatsoenlijke hardwareondersteuning en een rijke software-uitrusting bieden. [...] alle Linux-varianten kunnen als thuis-pc, Office-computer, ontwikkel-workstation of server worden gebruikt."* (Diedrich, 2001).

Ook bieden verschillende partijen, onder wie IBM en HP, hardware voorgeïnstalleerd met Linux aan. Bovendien kan de software door de openheid van de broncode volledig op maat gemaakt worden voor de eindgebruiker. Dit kan de gebruiksvriendelijkheid ten goede komen. Ook Torvalds, de geestesvader en hoofdontwikkelaar van de Linux-kernel, hecht veel waarde aan de toenemende gebruiksvriendelijkheid (Shankland, 2001b).

7.8 Focus op bepaalde typen software en lage gebruikscijfers

Nog niet in alle segmenten is Open Source Software goed vertegenwoordigd. Vooral op het gebied van applicaties is relatief weinig Open Source Software beschikbaar. Bepaalde projecten verkeren bovendien nog niet in een volwassen stadium. Ook is lang niet alle commerciële software geschikt voor OSS-platformen. Toch wordt op dit gebied vooruitgang geboekt. Producenten, zoals Sun en Netscape, bieden belangrijke applicaties tegenwoordig als Open Source Software aan. Daarnaast maken andere bedrijven, zoals Oracle, SAP en Corel, hun producten compatibel met Open Souce Software.[54]

Een nadeel van het inzetten van Open Source Software kan zijn dat informatie-uitwisseling met derden moeilijker wordt. Open Source Software wordt bijvoorbeeld op desktopniveau nog niet veel gebruikt. Hoewel vrijwel alle OSS compatibel probeert te zijn met commerciële software, blijkt dit in de praktijk moeilijk realiseerbaar. Commerciële producenten proberen met name als ze een dominante marktpositie innemen de compatibiliteit te frustreren, zodat ze optimaal kunnen profiteren van de netwerkeffecten. Microsoft biedt bijvoorbeeld geen openheid over het opslagformaat van Office-bestanden, waardoor de realisatie van volledige compatibiliteit door derden bemoeilijkt wordt. Dit brengt met zich mee dat een overstap naar Open Source Software vrijwel altijd gepaard met aanzienlijke switching costs voor de

[54] Zie Appendix 3C voor website-adressen van verschillende bedrijven die actief zijn op het gebied van OSS.

eindgebruiker. Wellicht kan de open standaard XML voor data-uitwisseling in de toekomst voor meer compatibiliteit zorgen.[55]

7.9 Juridisch aanspreekpunt

Omdat Open Source Software niet exclusief door een bedrijf wordt vertegenwoordigd, wordt vaak als nadeel beschouwd dat er geen juridisch aanspreekpunt bestaat. In beginsel wordt Open Source Software inderdaad "*as is*" geleverd, dat wil zeggen zonder enige garanties. Door de toenemende ondersteuning van OSS door computerbedrijven, zoals Red Hat en IBM, nemen ook de garanties toe voor eindgebruikers. Bovendien sluiten commerciële softwareproducenten via zogenaamde shrink-wrap licenties iedere vorm van aansprakelijkheid uit. Ter illustratie hiervan een citaat uit de gebruiksrechtenovereenkomst van Microsoft: *"Voor zover is toegestaan onder toepasselijk recht, is de PC-Fabrikant noch diens leveranciers aansprakelijk voor enige schade (daaronder begrepen maar niet beperkt tot directe of indirecte schade door lichamelijk letsel, winstderving, bedrijfsonderbreking, verlies van bedrijfsinformatie of enig ander geldelijk verlies) die is ontstaan door het gebruik of verhindering tot gebruik van dit produkt, ook indien de PC-Fabrikant op de hoogte gesteld is van het risico van dergelijke schade. In ieder geval zal de totale aansprakelijkheid van de PC-Fabrikant en diens leveranciers onder welke voorwaarde van deze overeenkomst dan ook, beperkt zijn tot het daadwerkelijk door u voor de SOFTWARE en/of de Microsoft hardware betaalde bedrag."*[56] Het blijkt hier dus grotendeels te gaan om een *verondersteld* nadeel, dat in de praktijk niet blijkt te bestaan. Open Source Software biedt niet per definitie minder juridische zekerheid dan commerciële software.

7.10 Kosten

Uiteindelijk zullen de genoemde voor- en nadelen hun uitwerking hebben op de kosten van de software. Om een volledig beeld te krijgen dient uitgegaan te worden van een integrale kostenbenadering, de zogenaamde Total Cost of Ownership (TCO). Voor de berekening van de TCO bestaat echter geen eenduidige methode (Breeveld e.a., 1999, p80). In het onderstaande zal aan de hand van verschillende kostendragers meer inzicht worden geboden in de kosten waarmee bij de inzet van Open Source Software rekening dient te worden gehouden.

[55] XML - Extensible Markup Language, http://www.w3.org/XML.
[56] Zie appendix C voor verschillende licenties.

- Zoekkosten: De kosten om geschikte OSS te vinden kunnen hoger zijn. In vergelijking met commerciële software wordt OSS in veel mindere mate via reclame onder de aandacht gebracht. Toch wordt er in toenemende mate reclame voor Open Source Software gemaakt. IBM voert bijvoorbeeld sinds maart 2001 een *"Peace, Love & Linux"* reclamecampagne, welke van start ging met een enorme billboard op Times Square in New York.[57] Distributeurs bieden Open Source Software vaak gebundeld aan, waardoor de software ook direct onder de aandacht wordt gebracht van de eindgebruiker. De distributie van SuSE bevat bijvoorbeeld ongeveer 2000 programma's, waarvan het merendeel Open Source Software is.
- Aanschafkosten: Open Source Software is gratis verkrijgbaar, omdat deze vrij kopieerbaar en verspreidbaar is. Toch zal OSS meestal via een distributeur worden betrokken. De software is ook dan relatief goedkoop en bovendien mag een pakket op een onbeperkt aantal machines worden geïnstalleerd. De lage aanschafkosten maken het gemakkelijker de software te testen, voordat men een definitieve keuze maakt. Uit Duits onderzoek kwam naar voren dat de initiële aanschafkosten ongeveer vijf keer lager liggen indien Open Source Software in plaats van commerciële software wordt gebruikt.[58]
- Scholingskosten: Deze hoeven niet hoger te liggen, maar omdat OSS afwijkt van de veelgebruikte Windows-omgeving zullen deze bij een overstap toch hoger zijn. Linux is bijvoorbeeld een Unix-variant en heeft geheel andere eigenschappen dan Windows. Daardoor zal de leercurve voor Windows-gebruikers zeker voor systeembeheerders behoorlijk stijl zijn als zij overschakelen naar Linux. Er worden wel steeds meer cursussen over Linux aangeboden. Op desktopniveau hoeft de overstap van Windows naar Linux niet voor grote problemen te zorgen. Distributeurs leveren Linux altijd met een grafische schil, die zeer veel gelijkenis kent met de Windows-interface. Ook veel andere OSS-programma's voor de desktop, zoals the Gimp en Mozilla, kennen een gebruiksvriendelijke interface.
- Operationele kosten: Zoals eerder beschouwd, is de kwaliteit van OSS in potentie hoog. Het blijkt ook dat professionals de kwaliteit van Open Source Software hoog aanslaan (Breedveld e.a., 1999, p.88; Godden, 2000). Door de stabiliteit kunnen kostenvoordelen worden behaald op het gebied van onderhoud. Bovendien kan de software door de open broncode afgestemd worden op de eisen en kunnen eventuele onvolkomenheden zelf

[57] http://www-1.ibm.com/servers/eserver/linux/passport.swf
[58] In casu werden vergelijkingen gemaakt op basis van een webserversysteem en een routersysteem met internettoegang.

worden opgelost. Dit laatste is natuurlijk uiteraard alleen een voordeel voor organisaties met toegang tot voldoende technische kennis.

- Ondersteuningskosten: Ondersteuning van Open Source Software, en zeker van Linux en Apache, is in de regel van aanvaardbaar niveau. OSS zal op dit gebied geen kostenvoordelen met zich meebrengen. Hoewel de ruime ondersteuning via internet wel veel mogelijkheden biedt om zelf meer inzicht te krijgen in problemen en deze eventueel ook zelf op te lossen.
- Switching costs: Open Source Software is vrijwel altijd gebaseerd op standaarden. Daardoor kunnen bijvoorbeeld data eenvoudig gedeeld worden met andere applicaties. Bovendien is Open Source Software in hoge mate portabel. De software kan daardoor relatief eenvoudig op een ander platform worden geïnstalleerd. De hoge mate van flexibiliteit, dat wil de combinatie van portabiliteit, compatibiliteit en aanpasbaarheid, zorgt ervoor dat Open Source Software een oplossing voor de lange termijn kan bieden. Als eenmaal gekozen is voor OSS zullen switching costs niet of nauwelijks meer gemaakt hoeven te worden. Echter, door een huidige lock-in situatie kan de overschakeling van commerciële software naar OSS wel aanzienlijke switching costs voor een bedrijf met zich meebrengen.

De Total Cost of Ownership kent vele facetten en is in hoge mate afhankelijk van het softwareproduct. Met name op het vlak van de aanschafkosten is OSS duidelijk in het voordeel, maar dit vormt slechts een klein deel van de TCO. Ook op het gebied van operationele kosten lijkt OSS (op zijn minst in potentie) kostenvoordelen met zich mee te kunnen brengen. Bovendien lijken de kosten op lange termijn lager, omdat switching costs zich nauwelijks meer zullen voordoen. Echter, met name op de korte termijn kan er voor Windows-georiënteerde gebruikers sprake zijn van relatief hoge scholingskosten en switching costs.

7.11 Afronding

Concluderend kan worden gesteld dat Open Source Software in potentie een hoger niveau van technische kwaliteit en veiligheid kent dan commerciële, gesloten software. Ook in de praktijk is gebleken dat de open ontwikkelingsmethode kwalitatief hoogwaardige software kan opleveren. Linux en Apache tonen dit aan. Daarnaast is Open Source Software duidelijk in het voordeel op het vlak van aanschafkosten. Echter, over het algemeen zullen deze slechts een klein deel uitmaken van de TCO.

Commerciële software lijkt in de regel voordelen te bieden qua gebruiksvriendelijkheid ten opzichte van Open Source Software. Ook het feit dat op bepaalde vlakken niet of nauwelijks Open Source Software bestaat, spreekt in het voordeel van commerciële software. Bovendien kan een huidige "lock-in" situatie een overstap naar Open Source Software bemoeilijken.

Op het vlak van ondersteuning doen zich geen wezenlijke verschillen voor. Echter, net als bij commerciële software geldt voor OSS in de regel dat des te populairder de software is des te breder en beter de ondersteuning. Omdat certificatie nog grotendeels in kinderschoenen staat, dient een ondersteunende derdepartij zorgvuldig te worden uitgekozen. Qua planning en juridische zekerheid doet Open Source Software zeker niet onder voor zijn commerciële tegenhanger.

Open Source Software kan door zijn eigenschappen in ieder geval prima ingezet worden op serverniveau, bijvoorbeeld als webserver, file-/printserver of als databankserver. Op dat niveau is zeer beproefde Open Source Software beschikbaar. Bovendien zal op dit niveau gebruiksvriendelijkheid in de zin van grafische interfaces een kleinere rol spelen en wordt in de regel veel gebruik gemaakt van open standaarden. Op desktopniveau is het eveneens goed mogelijk om Open Source Software als alternatief in te zetten voor commerciële software. Echter, op dit vlak is OSS momenteel minder volwassen en zijn gesloten standaarden meer regel dan uitzondering. Deze laatste kunnen zorgen voor hoge switching costs bij een overstap van 'gesloten' naar 'open'. Toch biedt Open Source Software voor de lange termijn ook hier duidelijk voordelen. De flexibiliteit zorgt er namelijk voor dat switching costs zich, nadat de overstap is gemaakt, niet of nauwelijks meer voor zullen doen.

Met betrekking tot Linux kwam een onderzoeker van Gartner onlangs tot vergelijkbare conclusies: *"Linux is at a crossroads. Gartner has detected serious interest from large enterprises. Yet, most IT executives also recognize that Linux works on a different model and that whatever gains they can achieve will be dependent on the support relationships with vendors and the breadth of application software--not just system infrastructure. Linux is being viewed as an opportunity to enable users to get out from under the yoke of proprietary platforms and high software license fees and into a much more flexible and evenhanded negotiating position. But vendors will always seek new opportunities to wedge users into*

proprietary solutions, so users must remain vigilant to avoid past mistakes that led to lock-in." (Weiss, 2001).

Hoofdstuk 8: Conclusie

"Open source software is important because it represents the first stage of something that will eventually become more widespread: shared projects carried out over the Net by people who are geographically and organizationally independent." – Tim O'Reilly (Booch, 2001, p.121).

Open Source Software is geen nieuw fenomeen. In de begindagen van de softwarehistorie was het zeer gebruikelijk de broncode mee te leveren met de software. De software was aan de hardware gebonden en werd daarom in vaste combinatie met de bijbehorende hardware geleverd. Met name in wetenschappelijke kring werd deling van broncode als een krachtig middel tot innovatie beschouwd. Software verloor na verloop van tijd echter aan aantrekkingskracht voor wetenschappers en de overheid verminderde haar steun voor onderzoek op het gebied van software. Bovendien nam de portabiliteit van software toe waardoor deze in toenemende mate minder gebonden werd aan bepaalde hardware. Dit bood mogelijkheden voor commerciële bedrijven om met commerciële 'gesloten broncode' software geld te verdienen. Commerciële software werd meer en meer van uitzondering tot regel, echter Open Source Software bleef op de achtergrond bestaan.
De eigenschappen van software, met name verspreidingsgemak en netwerkeffecten (oftewel anti-rivaliteit), leiden ertoe dat de commerciële softwaremarkt neigt naar een sterke concentratie onder de aanbieders. Het voorbeeld bij uitstek is Microsoft, dat de voornoemde schaalvoordelen aan aanbod- en vraagzijde uitstekend heeft benut en daarmee een overheersende positie op de softwaremarkt heeft weten te verwerven.

Het internet, dat in 1993 het levenslicht zag, is gebaseerd op open standaarden en is misschien wel het krachtigste product dat vrije, open innovatie heeft voortgebracht. Hoewel programmeurs reeds eerder via het ARPAnet samenwerkten, gaf het 'nieuwe' wereldwijde netwerk OSS-ontwikkeling een enorme impuls. Op dit moment bestaat er een grote variëteit aan levendige OSS-projecten op uiteenlopende gebieden, maar met name op het gebied van infrastructuur en tools is de software volwassen. Linux en Apache zijn succesvolle voorbeelden, die beide onder meer laten zien dat modulariteit een kritieke factor vormt. Het internet bundelde de krachten van OSS-programmeurs over de hele wereld. De volstrekte openheid, die kenmerkend is voor OSS-ontwikkeling, zorgt ervoor dat door het verspreidingsgemak in combinatie met de anti-rivaliteit van software via het internet veel potentiële programmeurs kunnen worden bereikt. Er zijn verschillende redenen van motivatie

te onderscheiden die verklaren waarom programmeurs daadwerkelijk vrijwillig bijdragen leveren aan OSS-projecten. Meestal zal een programmeur gemotiveerd worden door een combinatie van redenen. Ook hier spelen netwerkeffecten een rol. Door het verspreiden van de software inclusief broncode kan de programmeur namelijk positieve feedback ontvangen. Deze positieve feedback bestaat bijvoorbeeld uit een leereffect, verbeterde reputatie en/of verbeterde software. Het aantal (potentiële) gebruikers is bepalend voor de hoogte van deze positieve feedback en dus ook voor de motivatie. Tegenwoordig zijn er ook steeds meer OSS-programmeurs die in betaalde dienst zijn bij een wetenschappelijke instelling of een commercieel bedrijf.

Software is een logisch informatiegoed, waardoor parallelle, gedistribueerde samenwerking op afstand aan software zeer goed mogelijk is. Echter, software is zeer complex. Om deze complexiteit te reduceren en om het arbeidspotentieel dat ter beschikking staat optimaal te benutten dient de software in hoge mate modulair te worden opgebouwd. De technische structuur kent aldus een evenbeeld in de vorm van de organisatie. Samenwerkingsverbanden binnen OSS-projecten voldoen sterk aan de kenmerken van virtuele organisaties, waarbij vertrouwen (trust) en gemeenschappelijke uitgangspunten en doelen de bindingsfactoren zijn. De open context, waarbinnen OSS ontwikkeld wordt, werkt stimulerend voor de binding. De ontwikkeling vindt in hoge mate parallel en gedistribueerd plaats. Door frequente iteraties groeit de software incrementeel. Er is wel sprake van een functionele hiërarchie op basis van merites. Samenvoeging van code, (informele) planning en ontwerp vormen overheadtaken, die centraal door de projectleiding worden uitgevoerd. Deze taken zijn niet of nauwelijks parallel en gedistribueerd te coördineren. Met name voor foutopsporing en foutoplossing is de parallelle, gedistribueerde werkwijze zeer waardevol.

Het verschil tussen commerciële softwareontwikkeling en OSS-ontwikkeling kan uiteindelijk als volgt worden omschreven. De commerciële producent blijkt de netwerkeffecten en het verspreidingsgemak op basis van geslotenheid in te zetten op verkoopniveau om op zijn minst de hoge ontwikkelingskosten veroorzaakt door de complexiteit van software terug te verdienen. Daarentegen lijkt OSS de complexiteit van software juist te reduceren door verspreidingsgemak en netwerkeffecten ook op ontwikkelingsniveau op basis van openheid in te zetten. Modulariteit zorgt er hierbij als katalysator voor dat de krachten gebald kunnen worden. In de kern gaat het aldus om de inzet van de krachten respectievelijk *ex post* of *a priori* ter afhandeling van de complexiteit.

Ten aanzien van de sterken en zwaktes dient te allen tijd bedacht te worden dat OSS-ontwikkeling niet zaligmakend is, en dat ieder programma daarom uiteindelijk op zichzelf beoordeeld dient te worden. Echter, de praktijk heeft geleerd dat OSS zich in het algemeen vaak op bepaalde punten onderscheidt van commerciële software. De OSS-ontwikkelingsmethode levert uiteindelijk software op die bijvoorbeeld voordelen kan bieden op het vlak van aanschafkosten, kwaliteit, veiligheid en flexibiliteit. Deze sterke kanten kunnen duidelijk een drukkende werking hebben op de Total Cost of Ownership. Hoewel OSS-ontwikkeling aldus een aantal zeer sterke kanten heeft, zijn er ook tekortkomingen. Deelnemers aan OSS-projecten hebben veelal een sterk technische achtergrond. Voor bepaalde software zal het moeilijk zijn om een draagvlak te creëren, omdat door de aard van de software minder kan worden ingespeeld op de motivatie van potentiële programmeurs. Daarom is OSS niet in alle segmenten sterk vertegenwoordigd, en bovendien is de software vaak minder gepolijst. Open Source Software is momenteel om die reden nog minder geschikt voor desktop niveau waar de gebruiker niet of nauwelijks technisch onderlegd is. Echter, Open Source Software kan zeer goed worden ingezet op serverniveau waar de gebruikers een technische achtergrond hebben. Op desktopniveau maakt OSS wel snelle vorderingen qua gebruiksvriendelijkheid en qua hoeveelheid volwassen applicaties, maar bestaat er nog een achterstand op de commerciële marktleiders. Bovendien zijn de eenmalige switching costs bij een overschakeling van 'gesloten' naar 'open' op desktop niveau relatief hoog.

Ook het bedrijfsleven erkent de kracht van OSS-ontwikkeling. Commerciële producenten pogen de methoden van OSS-ontwikkeling over te nemen om ook de vruchten te kunnen plukken. De volstrekte openheid van OSS brengt echter unieke voordelen met zich mee. Daarom zijn er ook veel producenten die onder het motto *"Join them if you can't beat them"* Open Source Software actief ondersteunen en/of bijdragen leveren. OSS-ontwikkeling neemt waarschijnlijk nog aan kracht toe door de verdere ontwikkeling van het internet dat goedkoper, sneller en uitgebreider zal worden. Bovendien zal het alumni-effect, een netwerkeffect waardoor OSS-programmeurs een kennisvoorsprong hebben ten opzichte van hun commerciële collega's, de kracht van Open Source Software in de toekomst verder doen toenemen.

Hoewel de kracht van OSS-ontwikkeling nog zal toenemen, kan dit de eerder geschetste nadelen nooit geheel wegnemen, omdat deze grotendeels inherent zijn. Er zal daarom altijd

een voorname plaats voor commerciële software op de softwaremarkt zijn weggelegd. Beide typen softwareontwikkeling, gesloten en open, zijn wel meer dan ooit aan elkaar gewaagd. In mijn ogen ontstaat hierdoor de meest ultieme vorm van concurrentie op de software markt. Door OSS zal het corrigerend vermogen van de markt groter worden, omdat netwerkeffecten als concurrentiemiddel grotendeels kunnen worden uitgeschakeld. Bovendien biedt de open, modulaire structuur van OSS mogelijkheden voor commerciële bedrijven tot het aanbieden van complementaire producten. De eindgebruiker zal uiteindelijk vrijer zijn in zijn keuze voor software. Daardoor zal de concurrentie tussen commerciële softwareproducenten toenemen, en dit komt de innovatie ten goede. *Kortom, de bron zal niet heersen, maar haar kracht zal de 'zieke' markt wel gezonder maken. En eigenlijk ligt dit voor de hand, want staan bronnen immers niet bekend om hun helende kracht?*

Literatuur

Arief, B., Gacek, C., Lawrie, T. (2001), *Software Architectures and Open Source Software – Where can Research Leverage the Most?*, Centre for Software Reliability, Department of Computing Science, University of Newcastle, http://opensource.ucc.ie/icse2001/ariefgaceklawrie.pdf.

Baldwin, C.Y., Clark, K.B. (1997), *Managing in an age of modularity*, Harvard Business Review, September-October, pp.84-93.

Beale, J., Seifried, K. (2000), *Open Source – Why it's good for security*, April 17, http://securityportal.com/topnews/os20000417.html.

Behlendorf, B. (1999), *Open Source as a Business Strategy*, in: Dibona, C., Ockman, S., en Stone, M. red., *Open Sources: Voices from the Open Source Revolution*, O'Reilly and Associates, pp.149-170.

Berger, R. (1999), *Microscared - The Challenge of Open Source Software*, Linux Magazine, http://www.linux-mag.com/1999-05/microscared_01.html.

Bezroukov, N. (1999), *A second look at the Cathedral and the Bazaar*, First Monday, volume 4 (12), http://firstmonday.org/issues/issue4_12/bezroukov/index.html.

Blechermann, B. (1999), *The cathedral versus the Bazaar (with apologies to Eric S. Raymond), An economic analysis and strategic look at Open-Source Software*, http://www.ite.poly.edu/htmls/chapel_printable.htm.

Bluescope (2001), *Apache webserver ook in Nederland marktleider*, http://www.bluescope.nl/statistieken.php.

Booch, G. (2001) *Software Solutions, Developing the Future, Communications of the ACM, 44 (3).*

bootNet.com (1999), *Linux Manifesto*, http://www.bootnet.com/youaskforit/lip_linux_manifesto.html.

Breedveld, P., A. Koldewe, R. Scharis, R. Westerhof (1999), *De economische betekenis van Open Source Software in Nederland*, Onderzoeksrapport, International Data Corporation (IDC) i.o.v. Ministerie van Economische Zaken.

Brooks, F.P. jr. (1995), *The Mythical Man Month: Essays on Software Engineering, 20th Anniversary edition*, Addison-Wesley.

Bundesministerium für Wirtschaft und Technologie (2001), *Alternative Betriebssysteme, Open-Source-Software - Ein Leitfaden für kleine und mittlere Unternehmen*,

http://www.bmwi.de/textonly/Homepage/download/infogesellschaft/Open-Source-Software.pdf.

Chicago Sun-Times (2001), *Microsoft CEO takes launch break with the Sun-Times*, 1 juni, http://www.suntimes.com/output/tech/cst-fin-micro01.html

Conway, M.E. (1968), *How do committees invent? Datamation*, 14(4), pp.28-31.

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

Cox, A. (2000), *Re: Linux 2.4 before 2001?*, Linux Kernel mailing List, 5 januari, archief: http://www.uwsg.indiana.edu/hypermail/linux/kernel/index.html

Corbató, F.J. (1995), *Het bouwen van systemen die zullen falen*, Informatie nr. 2, pp.140-149.

Dellio, M. (2001), *The Story Behind Tux the Penguin*, http://www.wired.com/news/culture/0,1284,42209,00.html.

Diedrich, O. (2001), *Haute Couture, Actuele Linux-distributies vergeleken*, c't, nr. 10, oktober.

Evers, S. (2000), *An introduction to Open Source Software Development*, Technische Universität Berlin.

Genuchten, M.J.I.M., Heemstra, F.J., Bemelmans, T.M.A. (1993), *De software-fabriek: beheersing van ontwikkeling, onderhoud en hergebruik*, Informatie, nr. 4.

Ghosh, R.A., V. Ved Prakash (2000), *The Orbiten Free Software Survey*, http://www.fistmonday.org/issues/issue5_7/ghosh/index.html.

Godden, F. (2000), How do Linux and Windows NT measure up in real life?, http://gnet.dhs.org/stories/bloor.php3, januari.

Free Software Foundation (FSF) (1998), *Categories of Free and Non-Free Software*, http://www.fsf.org/philosophy/categories.html.

Harmon, A. (1998), *For Sale: Free Operating System*, The New York Times, September 28, http://www.nytimes.com/library/tech/98/09/biztech/articles/28linux.html.

Jägers, H.P.M., Jansen, W., Steenbakkers, G.C.A. (1998), *Characteristics of virtual organizations*, PrimaVera Working Paper Series, http://primavera.fee.uva.nl.

Jägers, H.P.M. (2001), *Sheets Hoorcollege*, Innovatie en Informatiesystemen, UvA.

Johnson, K. (2001), *A Descriptive Process Model for Open-Source Software Development [DRAFT]*, http://www.cpsc.ucalgary.ca/~johnsonk/thesis/final_draft.pdf.

Johnson, P. (1999), *Liberal Source Software*, http://www.elj.com/lss/.

Jones, P. (2000), *Brooks' law and open source: The more the merrier?, Does he open source development method defy the adage about cooks in the kitchen*, http://www-106.ibm.com/developerworks/linux/library/merrier.html.

Kaminsky, D. (1999), *Core competcies: Why Open Source is The Opetimum Economic Paradigm for Software*, http://www.doxpara.com/read.php/core.html.

Katz, M.L., Shapiro, C. (1998), *Antitrust in software markets*, Haas School of Business, University of California at Berkeley, http://www.haas.berkeley.edu/~shapiro/software.pdf.

Koelman, K.J. (2000), *Terug naar de bron: open source en copyleft*, Informatierecht/AMI 2000-8, pp. 149-155.

Koning, C. de, Aalst, L. van der (1999), *Testen duur? Niet testen is duurder! Bepaal het testkostenoptimum met eenvoudige kengetallen*, Informatie nr.10, http://www.informatie.nl/artikelen/1999/10/TestenduurNiettestenisdu_1.html.

Kuwabara, K. (2000), *Linux: A bazaar at the edge of Chaos*, http://www.firstmonday.org/issues/issue5_3/kuwabara.

Lerner, J., Tirole, J. (2000), *The Simple Economics of Open Source*, Working Paper 7600, National Bureau Of Economic Research.

Lessig, L. (2000), *Open Code and Open Societies*, http://cyberlaw.stanford.edu/lessig/content/index.html.

Levy, S. (1984), *Hackers: Heroes of the Computer Revolution*, Anchor Press/Doubleday.

Meij, van der M. (2000), *The Economics of Using Open-Source Software*, Vrije Universiteit Amsterdam.

Mockus, A., Fielding, R.T., Herbsleb, J. (2000), *Case Study of Open Source Software Development: The Apache Server*, In Proceedings of the International Conference on Software Engineering, Limerick Ireland, June 5-7, pp. 263-272, http://www.bell-labs.com/org/11359/colab_prod/apachefinal3.pdf.

Moglen, E. (1999), *Anarchistic Triumphant: Free Software and the death of copyright*, http://emoglen.law.columbia.edu/publications/anarchism.html.

Moon, J.Y., Sproul, L. (2000), *Essence of Distributed Work: The Case of the Linux kernel*, First Monday, volume 5, number 11, http://firstmonday.org/issues/issue5_11/moon/index.html.

Murray, J. (2000), *Inspired by open source, Gaining the benefits of open source in a commercial development environment*, http://oss.software.ibm.com/developer/opensource/features/inspired.html.

Mundie, C. (2001), *Commercial software, sustainable innovation*, 17 mei, http://news.cnet.com/news/01-1276-210-5952851-1.html.

Netcraft (2001), *Web Server Survey*, 10 juni 2001, http://www.netcraft.com/survey/.

Ouwersloot, J. (1994), *Information and communication from an economic perspective*, diss., Vrije Universiteit Amsterdam.

Raymond, E.S. (1999), *The Cathedral and the Bazaar*, http://www.tuxedo.org/~esr/writings.

Ricciuti, M. (1997), *Borland sues Microsoft over brain drain*, 7 mei, http://news.cnet.com/news/0,10000,0-1005-200-318765,00.html.

Rispens, S.I. (2001), *Apple bakt een nieuwe taart*, Intermediair 14, 5 april.

Scannel, E. (1999), *Linus Torvalds says open source not guarantee of success*, 6 oktober, http://www.infoworld.com/cgi-bin/displayStory.pl?99106.pitorvalds.htm.

Shankland, S. (2001a), *Red Hat meets estimates*, 19 juni, http://news.cnet.com/news/0-1003-200-6323556.html.

Shankland, S. (2001b), *Making Linux usable tops Torvalds' list*, 31 augustus http://www.zdnet.com.au/biztech/enterprise/story/0,2000010343,20258097,00.htm.

Shrankland, S. (2001c), *Linux growth underscores threat to Microsoft*, 28 februari, http://news.cnet.com/news/0-1003-202-4979275.html.

Shapiro, C., Varian, H.R. (1999), *Information Rules: A Strategic Guide to the Network Economy*, Harvard Business School Press.

Shapiro, C. (1999), *Exclusivity in Network Industries*, Haas School of Business, University of California at Berkeley, http://www.haas.berkeley.edu/~shapiro/exclusivity.pdf.

Shapiro, C. (2000), *Setting Compatibility Standards: Cooperation or Collusion?*, Haas School of Business, University of California at Berkeley, http://www.haas.berkeley.edu/~shapiro/standards.pdf.

Stallman, R. (1992), *Why software should be free*, Paper, GNU Project Free Software Foundation, http://www.gnu.org/philosophy/shouldbefree.html.

Stone, M. (2001), *Understanding the Linux Kernel*, http://www.osdn.com/osdocs/01/01/15/2224213.shtml.

Torvalds, L. B. (1999), *The Linux Edge*, in: Dibona, C., Ockman, S., en Stone, M. red., *Open Sources: Voices from the Open Source Revolution*, O'Reilly and Associates, pp.101-111.

Torvalds, L. B. (2001), Diamond, D., *Just for fun, the story of an accidental revolutionary*, Harper Collins Publishers, New York.

Tuomi, I. (2001), *Internet, Innovation, and Open Source; Actors in the Network*, First Monday, volume 6, number 1, http://firstmonday.org/issues/issue6_1/tuomi/index.html.

Uytterhoeven, G. (1999), Open Source Software (OSS) Ontwikkeling, http://home.tvd.be/cr26864/.

Varian, H.R. (1998), *Markets for Information Goods*, Paper, University of Califoria at Berkely, http://www.sims.berkeley.edu/~hal/Papers/japan/index.html.

Viega, J. (1999), *Open source software: Will it make me secure?, Alleviating the risks of opening up your source can pay off in improved security,* September, http://www-106.ibm.com/developerworks/security/library/oss-security.html.

Vreven, A.A. (1994), *Open systemen,* Informatie, nr.1, pp. 60-70.

Vries, E.J. de, Stegen, F.S. (2000), *Multichanneling in de dienstverlening, het (PMC)^2-raamwerk voor kanaalcoördinatie,* Management & Informatie, nr. 1, jg. 8, pp. 4-15.

Walker Whitlock, N. (2001), *The security implications of open source software, does open source mean an open door?,* http://www-106.ibm.com/developerworks/linux/library/l-oss.html.

Wall, L. (1999), Diligence, Patience, and Humility, in: Dibona, C., Ockman, S., en Stone, M. red., *Open Sources: Voices from the Open Source Revolution,* O'Reilly and Associates, p.127-147.

Wayner, P. (2000), *Free for All, How Linux and the Free Software Movement Undercut the High-Tech Titans,* Harper Business.

Weiss, G. (2001), *What's the future of Linux?,* 23 Oktober, http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2819787,00.html.

Weber, S. (2000), *The Political Economy of Open Source Software,* BRIE Working Paper 140, E-conomy Project Working paper 15.

Webwereld (2001a), *Zowel Linux als Windows groeien sterk in OS-markt,* 1 maart, http://www.webwereld.nl/nieuws/6851.phtml.

Webwereld (2001b), *Linux wint steeds meer terrein op servermarkt,* 12 april, http://www.webwereld.nl/nieuws/7275.phtml.

Webwereld (2001c), *Microsoft wil einde aan openheid over lekken,* 18 oktober, http://www.webwereld.nl/nieuws/8859.phtml.

Wegberg, M., Berens, P. (2000) *Competing communities of users and developers of computer software: competition between open source software and commercial software,* NIBOR working paper, NIBOR/RM/00/001, http://www.unimaas.nl/~mwegberg/index.htm.

Wigand, R., Picot, A., Reichwald, R. (1997), *Information, organization and management, Expanding Markets and Corporate Boundaries,* John Wiley & Sons Ltd.

Wilcox, J. (2000a), *Microsoft secret file could allow access to Web sites*, 14 april, http://news.cnet.com/news/0-1003-200-1696137.html.

Wilcox, J. (2000b), *IBM to spend $1 billion on Linux in 2001*, http://news.cnet.com/news/0-1003-200-4111945.html, 12 december.

Zawinski, J. (1999), *Resignation and postmortem*, http://ww.jwz.org/gruntle.nomo.html.

Geraadpleegde websites

CNet - http://linux.cnet.com

Dagboek Alan Cox: http://www.linux.org.uk/diary/

Homepage Linus Torvalds: http://www.cs.helsinki.fi/u/torvalds/

Linux Documenation Project - http://www.linuxdoc.org/

Linux on handhelds - http://www.handhelds.org/

Linux Today - http://linuxtoday.com

Linux-kernel mailing list samenvatting en archief - http://kt.zork.net/kernel-traffic/latest.html / http://www.uwsg.indiana.edu/hypermail/linux/kernel/index.html

LWN.net - http://lwn.net/

Open Source Software in der Bundesverwaltung, Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung (KBSt) - http://linux.kbst.bund.de

Slashdot – http://www.slashdot.org

The Linux-kernel mailing list FAQ: http://www.tux.org/lkml

Tienjarig jubileum van Linux - http://www.linux10.org

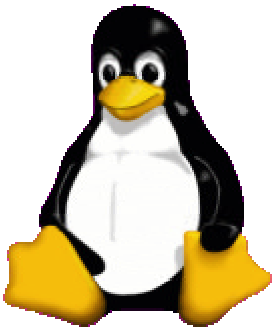
Tweakers - http://www.tweakers.net

Webwereld - http://www.webwereld.nl

ZDNet – http://www.zdnet.com

Appendices

Appendix A: Logo's van enkele OSS-projecten



Linux logo genaamd Tux ontworpen door Larry Ewing met GIMP, zie http://www.isc.tamu.edu/~lewing/linux

Logo van de Apache HTTP webserver, zie http://www.apache.org

Logo van GNU-project (Gnu's Not UNIX) / Free Software Foundation (FSF), zie http://www.fsf.org

Appendix B: Bedrijven en instellingen die actief zijn op het gebied van OSS

B1. OSS-samenwerkingsfora

CollabNet - http://www.collab.net

Freshmeet – http://www.freshmeet.com

Open Source Developers network – http://www.osdn.com

Sourceforge.net - http://www.sourceforge.net

B2. Linux distributeurs

Caldera (VS) – http://www.caldera.com

Connectiva (Brazilië) – http://www.connectiva.com

Debian (VS, non-profit) – http://www.debian.org

Mandrake (Frankrijk) – http://www.mandrake.com

Red Hat (VS) – http://www.redhat.com

SuSE (Duitsland) – http://www.suse.com

TurboLinux (VS, gericht op Azië) – http://www.turbolinux.com

Lijst met Linux-distributeurs: http://www.linux.org/dist

B3. OSS-supporters / -ontwikkelaars

AMD – http://www.amd.com / http://www.x86-64.org

Compaq – http://www.compaq.com/linux

Corel (WordPerfect) - http://linux.corel.com

DELL - http://www.dell.com/linux

HP - http://www.hp.com/linux

IBM – http://www.ibm.com/linux/

Intel – http://www.intel.com / http://www.linuxia64.org

Nokia - https://www.ostdev.net

Open Source Development Lab - http://www.osdl.org

Oracle - http://technet.oracle.com/tech/linux/content.html

Philips (programmeertaal) -

http://www.research.philips.com/generalinfo/special/elegant/elegant.html

SAP - http://www.sap.com/linux

Sony (Playstation 2 & Linux) - http://www.ps2linux.scea.com

SGI - http://oss.sgi.com

Sharp Handhelds- http://developer.sharpsec.com

Sun - http://www.sun.com/linux / http://www.sunsource.net

Appendix C: Licenties

C1: Open Source Definition *(Open Source Initiative (OSI) – http://www.opensource.org)*

Version 1.8

Introduction

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost–preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Contaminate Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

C2: GNU General Public License (GPL) *(Free Software Foundation (FSF) – http://www.fsf.org)*

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original
authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without

limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

   In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library GeneralPublic License instead of this License.

C3: The BSD License

The following is a BSD license template. To generate your own license, change the values of OWNER, ORGANIZATION and YEAR from their original values as given here, and substitute your own. Note: The advertising clause in the license appearing on BSD Unix files was officially rescinded by the Director of the Office of Technology Licensing of the University of California on July 22 1999. He states that clause 3 is "hereby deleted in its entirety." Note the new BSD license is thus equivalent to the MIT License, except for the no-endorsement final clause.

<OWNER> = Regents of the University of California

<ORGANIZATION> = University of California, Berkeley

<YEAR> = 1998

In the original BSD license, the first occurrence of the phrase "COPYRIGHT HOLDERS AND CONTRIBUTORS" in the disclaimer read "REGENTS AND CONTRIBUTORS".

Here is the license template:

Copyright (c) <YEAR>, <OWNER>

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Appendix D: Eerste aankondigingen van Linux door Linus Torvalds

From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki

Hello everybody out there using minix - I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same hysical layout of the file-system (due to practical reasons) among other things). I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torvalds@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: Free minix-like kernel sources for 386-AT
Message-ID: <1991Oct5.054106.4647@klaava.Helsinki.FI>
Date: 5 Oct 91 05:41:06 GMT
Organization: University of Helsinki

Do you pine for the nice days of minix-1.1, when men were men and wrote their own device drivers? Are you without a nice project and just dying to cut your teeth on a OS you can try to modify for your needs? Are you finding it frustrating when everything works on minix? No more all- nighters to get a nifty program working? Then this post might be just for you :-)
As I mentioned a month(?) ago, I'm working on a free version of a minix-lookalike for AT-386 computers. It has finally reached the stage where it's even usable (though may not be depending on what you want), and I am willing to put out the sources for wider distribution. It is just version 0.02 (+1 (very small) patch already), but I've successfully run bash/gcc/gnu-make/gnu-sed/compress etc under it.
Sources for this pet project of mine can be found at nic.funet.fi (128.214.6.100) in the directory /pub/OS/Linux. The directory also contains some README-file and a couple of binaries to work under linux (bash, update and gcc, what more can you ask for :-). Full kernel source is provided, as no minix code has been used. Library sources are only partially free, so that cannot be distributed currently. The system is able to compile "as-is" and has been known to work. Heh. Sources to the binaries (bash and gcc) can be found at the same place in /pub/gnu.

ALERT! WARNING! NOTE! These sources still need minix-386 to be compiled (and gcc-1.40, possibly 1.37.1, haven't tested), and you need minix to set it up if you want to run it, so it is not yet a standalone system for those of you without minix. I'm working on it. You also need to be something of a hacker to set it up (?), so for those hoping for an alternative to minix-386, please ignore me. It is currently meant for hackers interested in operating systems and 386's with access to minix. The system needs an AT-compatible harddisk (IDE is fine) and EGA/VGA. If you are still interested, please ftp the README/RELNOTES, and/or mail me
for additional info. I can (well, almost) hear you asking yourselves "why?". Hurd will be out in a year (or two, or next month, who knows), and I've already got
minix. This is a program for hackers by a hacker. I've enjouyed doing it, and somebody might enjoy looking at it and even modifying it for their own needs. It is still small enough to understand, use and modify, and I'm looking forward to any comments you might have. I'm also interested in hearing from anybody who has written any of the utilities/library functions for minix. If your efforts are freely distributable (under copyright or even public domain), I'd like to hear from you, so I can add them to the system. I'm using Earl Chews estdio right now (thanks for a nice and working system Earl), and similar works will be very wellcome. Your (C)'s will of course be left intact. Drop me a line if you are willing to let me use your code.

Linus

PS. to PHIL NELSON! I'm unable to get through to you, and keep getting "forward error - strawberry unknown domain" or something.