Computerprogramma's: gevangenissen voor gebruikers of virtuele vrijplaatsen?

Over openheid als concept voor het democratiseren van technologie en het faciliteren van gebruikersparticipatie.

Stefan Verhaegh

Computerprogramma's: gevangenissen voor gebruikers of virtuele vrijplaatsen?
Over openheid als concept voor het democratiseren van technologie en het faciliteren van gebruikersparticipatie.

Stefan Verhaegh, ID-nummer 952007
Maastricht, april 2003

Tracé Technologische Cultuur

Begeleider: dr. G. Wackers (Faculteit der Cultuurwetenschappen, Universiteit Maastricht)
Tweede beoordelaar: dr. A. Hommels (Faculteit der Cultuurwetenschappen, Universiteit Maastricht)

Omslag en productie: Jan Eggen / Unigraphic

Inhoudsopgave

Voorwoord

Eens heel lang geleden, in het jaar 1988 in het Stefan-universum, bestond er nog geen World Wide Web, konden beeldschermen alleen groene lettertjes of grijze plaatjes in grove blokken weergeven, waren diskettes nog echt 'floppy', bezat een computer standaard nog geen muis, CD-Romspeler of geluidskaart en werkten alle programma's in MS-DOS. Zelf was ik elf, zat ik op de basisschool en schreef ik samen met mijn klasgenoot Martin voor het eerst een computerprogramma.

Dat deden we in de programmeertaal GW-BASIC, standaard aanwezig bij MS-DOS 3.33. Het mooie van Basic was, dat je de broncode nodig had om programma's te kunnen 'runnen'. Daardoor kon je bij elk Basic-programma kijken hoe het in elkaar zat. Vaak begrepen we niets van alle geheimzinnige woord- en cijferreeksen. Maar door dingen te veranderen of weg te halen, kwamen we er na veel gepruts toch achter hoe iets werkte.

Ons eerste programma dat we schreven, knutselden we in elkaar door stukken broncode uit Basic-programma's te knippen en vervolgens achter elkaar te plakken. Het resultaat noemden we 'MS-Menu', naar de eerste letters van onze voornamen. Met dit menuprogramma kon je programma's starten zonder daarvoor opdrachten in Dos in te typen. Bovendien kon je programma's aan functietoetsen op het toetsenbord toewijzen die zelfs na het afsluiten van MS-Menu gewoon in Dos werkzaam bleven. Zo kon je door op F5 te drukken bijvoorbeeld 'Mijnenveger' starten. Dit spelletje zelf was knudde, maar wij vonden het geweldig dat je het kon starten met één druk op een functietoets. Op deze manier kon je ook MS-Menu zelf rechtstreeks vanuit Dos starten. Hierdoor leek het net een 'echt' Dos-programma, in plaats van een 'simpel' Basicprogramma.

Ook al hadden we de afzonderlijke componenten van dit programma uit andere programma's 'gejat', toch was onze combinatie van onderdelen een uniek en nieuw programma. Wij, twee basisschooljongens, konden de computer laten doen wat wij wilden. Dat euforische gevoel zal ik niet snel vergeten. Helaas eindigde mijn programmeerhobby met de komst van Microsoft Windows, omdat daar geen gratis programmeeromgeving meer bij zat. Ook in de broncode van programma's kijken om te zien hoe alles werkte, was er niet meer bij.

Pas in het nieuwe millennium, tijdens mijn stage bij het populair-wetenschappelijke tijdschrift 'Natuur & Techniek', werd mijn interesse in computerprogramma's opnieuw gewekt. Voor een recensie bestelde ik boeken over 'Linux' en 'open source software'. In

‘The Cathedral and the Bazar’ (Raymond 1999), ‘Free for All’ (Wayner 2000) en ‘Open Sources’ (DiBona et al 1999) las ik over een zelfde soort enthousiasme en spelend leren als ik ooit zelf had ervaren tijdens mijn gehobby met Basic. Mijn interesse was gewekt. Tot een boekenrecensie heeft het uiteindelijk niet meer geleid; wel tot een scriptie.

De opkomst van ‘open’ en ‘vrije’ software, waaronder het GNU-project en Linux, maakt het opnieuw mogelijk om zelf te experimenteren met computerprogramma’s, net als toen ik tien was. Daardoor maakt vrije en open software een leukere, actievere en creatievere omgang met computers mogelijk, niet alleen voor experts, maar ook voor ‘leken’. Alle hulde en dank daarom voor programmeurs die hun werk met anderen delen, zoals bijvoorbeeld Bram Moolenaar dat doet met zijn programma Vim.

Daarnaast wil ik in dit voorwoord de mensen bedanken zonder wie deze scriptie niet tot stand had kunnen komen: mijn vrienden Antoon, Charlotte, Ester, Hemmo, Michiel en Raymond, mijn vriendin Katja, mijn ouders Jan en Maria, mijn zus Mirjam en mijn huisgenoten in Amsterdam. ‘Last, but not least’ bedank ik Ger Wackers voor zijn deskundige scriptiebegeleiding, kritische commentaar en waardevolle suggesties en ideeën.

Stefan Verhaegh, Amsterdam, 15 april 2003

1. Inleiding

§ 1.1 Computerprogramma's: gevangenis of vrijplaats?

> "Funny thing is I've never downloaded nor viewed the source code. But the source availability is an escrow guaranteeing I'll always have this software. Did I mention that a large portion of my business runs on Vim[1]?"[2]

Het bovenstaande citaat is afkomstig van één van de leden van de grote en levendige gemeenschap die pleit voor een vrij en democratisch gebruik van computerprogramma's. De belangrijkste voorwaarde hiervoor is de aanwezigheid van de originele broncode, die garandeert dat een programma altijd vrij toegankelijk blijft voor gebruikers.

Lijnrecht tegenover deze idealistische computergemeenschap die openheid en vrijheid in computergebruik voor iedereen nastreeft, staat de hedendaagse praktijk van de softwareindustrie, waar het verdienen van geld centraal staat, zowel letterlijk als figuurlijk ten koste van gebruikers. Om de winst te maximaliseren en de concurrentie uit te schakelen proberen bedrijven klanten aan zich te binden door hen volledig afhankelijk te maken van hun producten. Dit doen ze door het gebruikers onmogelijk te maken om inzicht te krijgen in de werking van de door hen gebruikte programmatuur.

Tegenwoordig zijn commerciële programma's hermetisch afgesloten 'zwarte dozen' waartoe de klant op geen enkele manier toegang kan krijgen. Voor het herstellen van fouten of het toevoegen van nieuwe functionaliteit is een gebruiker volledig aangewezen op de bereidwilligheid van de fabrikant. De fabrikant van een computerprogramma heeft door zijn exclusieve controle over de toegang tot de interne werking van een computerprogramma voor zichzelf een monopoliepositie gecreëerd, waardoor andere bedrijven van concurrentie uitgesloten zijn en gebruikers volledig overgeleverd zijn aan één bedrijf.

Het bekendste voorbeeld van een bedrijf dat deze tactiek hanteert is het Amerikaanse Microsoft, dat al talloze malen in de rechtszaal moest verschijnen voor vermeend misbruik van haar monopoliepositie als producent van computerprogramma's. Klanten die eenmaal met mooie beloftes en visueel snoepgoed door Microsoft waren binnengehaald, kwamen er pas later achter wat het betekende om voor alle IT-oplossingen van slechts één bedrijf afhankelijk te zijn in plaats van te kunnen kiezen uit een open markt van vraag en aanbod vol

[1] Vim is een open source teksteditor om tekstbestanden mee te bewerken. Hoofdstuk vijf is gewijd aan mijn onderzoek naar Vim als voorbeeld van een gemeenschap rondom een open computerprogramma.
[2] Bron: e-mailcorrespondentie met Steve Litt, onderdeel van mijn empirische onderzoek.

concurrentie.

Ook al garandeert deze slinkse handelwijze bedrijven zoals Microsoft een voortdurende bron van inkomsten, voor de gebruiker zijn de gevolgen van deze 'vendor-lock-in'-politiek minder positief. Enkele voorbeelden van het ontbreken van een gezonde concurrentie op het gebied van de software-industrie zijn bijvoorbeeld:

- onstabiele programma's die voortdurend vastlopen met dataverlies als gevolg,
- talloze veiligheidsproblemen en grootschalige schade door computervirussen,
- totale afhankelijkheid van één leverancier voor de reparatie van fouten en de implementatie van nieuwe functies in programma's,
- moeizame uitwisseling van informatie tussen computergebruikers door het gebruik van geheime en gesloten bestandsformaten die onderling niet compatibel zijn,
- computerprogramma's die privacy-gevoelige informatie van de gebruiker zonder zijn medeweten via internet naar centrale databases versturen,
- softwarelicenties die gebruikers steeds meer beperkingen opleggen,
- computerprogramma's die slechts een beperkt aantal talen en schriften ondersteunen en daardoor een bedreiging voor de culturele diversiteit vormen,
- stijgende kosten van software door steeds duurder wordende 'upgrades',
- doordat nieuwe versies van computerprogramma's niet langer op oude hardware werken is de gebruiker gedwongen om extra geld aan nieuwere en snellere hardware uit te geven.

Een beangstigend gevolg van de toenemende machtsconcentratie van softwaremakers ten opzichte van softwaregebruikers is het feit dat makers van computerprogramma's steeds meer invloed krijgen op de toegang tot digitale informatie in het algemeen. Een klein voorbeeld van machtsmisbruik vinden we bij sommige websites van Microsoft die niet of slecht toegankelijk zijn met webbrowsers van concurrenten[3]. Een ander voorbeeld is het feit dat Microsoft drie plekken bezet op de Nederlandse top tien van meest bezochte websites. Zo ontstaat niet alleen een softwaremonopolie, maar ook een informatiemonopolie. Een ander beangstigend idee is het feit dat Microsoft tegenwoordig niet alleen meer software voor personal computers maakt, maar voor een heel scala aan digitale apparatuur van zakagenda's, set top boxen, mobiele telefoons, spelcomputers en medische apparatuur tot zelfs hele oorlogsschepen aan toe.

Wanneer een gebruiker uiteindelijk door stijgende kosten een overstap naar een concurrent overweegt[4], durft hij die stap uiteindelijk niet te maken door de hoge

[3] Zie: http://slashdot.org/articles/03/02/06/1645229.shtml?tid=109
[4] Halverwege de jaren tachtig kostte een beetje PC voor thuisgebruik zo'n drieduizend gulden en MS-DOS

omschakelkosten die een dergelijke overstap met zich mee zou brengen. Want gebruikers en systeembeheerders moeten opnieuw getraind worden, programma's opnieuw geconfigureerd en talloze bestanden naar een ander opslagformaat omgezet. De computergebruiker zit gevangen in een virtuele gouden kooi van de softwarefabrikant. De constructie die dit alles mogelijk maakt is het feit dat de gebruiker de door hem gebruikte software niet in zijn bezit heeft, maar dat hij slechts een licentie heeft om de software te mogen gebruiken onder de voorwaarden van de fabrikant. Overigens is Microsoft niet het enige computerbedrijf dat de strategie van gesloten programma's hanteert, want dat doet tegenwoordig praktisch iedereen die software verkoopt, maar zij is wel het meest succesvol in termen van marktaandeel, winstgevendheid en het uitschakelen van concurrentie. Inmiddels staat op 450 miljoen van de wereldwijd ongeveer 500 miljoen in gebruik zijnde personal computers de software van Microsoft geïnstalleerd[5] en is oprichter Bill Gates de rijkste man op aarde.

Alhoewel de gemiddelde gebruiker afgezien van de hoge prijzen voor computerprogramma's meestal best tevreden is over de werking ervan (al dan niet uit onwetendheid over alternatieven), zijn veeleisende computergebruikers die voor hun dagelijkse werk in hoge mate afhankelijk zijn van de eigenschappen van de programma's waarmee ze werken, dat vaak absoluut niet. Veel wetenschappers, programmeurs, systeembeheerders, computerhobbyisten en andere 'power-users' stoten dagelijks hun hoofd aan de kooi-constructie van gesloten programma's die hen beperken in hun vrijheid om een computer te gebruiken op de manier zoals ze dat zelf willen.

Als reactie op deze situatie is daarom een wereldwijd, anarchistisch ongeorganiseerd netwerk van computergebruikers ontstaan. Binnen dit netwerk worden kennis, broncode en zelfgeschreven computerprogramma's uitgewisseld en helpen gebruikers elkaar bij het oplossen van problemen. Het overkoepelende principe dat al deze zeer diverse individuen verbindt is de liefde voor 'goed geschreven' broncode en programma's die 'gewoon werken'. Het delen van zowel kennis als broncode is hiervoor essentieel. Openheid staat hierbij centraal: iedereen mag de programma's uit de gemeenschap gebruiken, iedereen mag naar de broncode kijken en er zelf veranderingen in aan brengen. Hoe meer er gedeeld wordt, hoe groter de opbrengst voor de gemeenschap als geheel is. Aanvankelijk gebeurde dit delen nog

---

ongeveer 150 gulden. Tegenwoordig kun je al complete computers kopen voor minder dan 500 euro, terwijl een licentie voor het gebruik van Microsoft Windows XP inmiddels zo'n 250 euro kost. Dat betekent dat de relatieve kosten van de software (in de vorm van het besturingssysteem) ten opzichte van die van de hardware zijn gestegen van vijf naar vijftig procent. In deze calculatie houd ik geen rekening met aanvullende programma's zoals Office-pakketten, waardoor de totale kosten voor software die van de hardware zelfs ver kunnen overtreffen.

[5] Bron: Lezing Jon Hall op Fosdem (Free and Open Source Developers Meeting), Brussel, februari 2003.

via afdrukken van de broncode op papier of in tijdschriften, later via diskettes en tegenwoordig vindt de uitwisseling van dit soort programma's grotendeels via internet plaats.

Doordat de makers ook de originele broncode waarin het programma geschreven is meeleveren, is het voor anderen mogelijk om fouten in deze programma's te herstellen, nieuwe mogelijkheden in te bouwen, of ze geschikt te maken voor andere computersystemen dan waarvoor ze oorspronkelijk geschreven zijn. Op deze wijze zijn via samenwerking op internet inmiddels zelfs complete besturingssystemen geschreven, waarvan Linux het bekendste voorbeeld is.

Onlosmakelijk verbonden met deze 'open' computercultuur is het internet. Zo heeft het internet het mogelijk gemaakt om wereldwijd computergebruikers met dezelfde interesses op goedkope en efficiënte wijze met elkaar te laten communiceren. Tegelijkertijd echter bestaat het internet uit programmatuur die op open wijze tot stand is gekomen. Het fundament van het internet, namelijk het TCP/IP protocol, is een open protocol waarvan de specificaties voor iedereen beschikbaar zijn. Daarnaast was de eerste TCP/IP-implementatie in BSD Unix voor iedereen beschikbaar in de vorm van broncode, waardoor iedereen op eenvoudige wijze deze code kon integreren in zijn eigen programmatuur. Het grootste deel van de programma's die de infrastructuur vormen van het hedendaagse internet zijn op dezelfde open wijze tot stand gekomen. De bekendste voorbeelden zijn de 'domain name server' Bind, de e-mailserver Sendmail en de webpaginaserver Apache.[6]

Deze laatste soort software, die ik in mijn scriptie 'open' software noem, lijkt een veel democratischer karakter te bezitten dan de eerder genoemde 'gesloten' software. De vraag is echter of dit ook echt zo is. In hoeverre geldt, dat open gelijk staat aan democratisch en gesloten aan ondemocratisch, kan alleen door onderzoek blijken. De vraag naar de vormgeving van gevangenissen voor gebruikers of virtuele vrijplaatsen kun dus niet zomaar a priori beantwoord worden. Het is nodig om te beseffen dat technologie niet neutraal is, maar inherent politiek en sociaal geconstrueerd. Een analyse van de vormgeving van grenzen voor democratische participatie voor gebruikers van computertechnologie is dan ook noodzakelijk. De constructie van de structuur van het netwerk rondom 'open' computerprogramma's hangt namelijk nauw samen met het democratisch gehalte ervan. Juist inzicht in deze structuur kan een meer algemene bijdrage leveren aan de discussie over democratische technologie. Pas wanneer we inzicht hebben in deze processen, kunnen we een bijdrage leveren aan een visie op een hoogtechnologische informatiemaatschappij waar iedereen toegang toe heeft.

[6] Zie Wheeler 2002 voor meer voorbeelden van de succesvolle inzet van open software.

§ 1.2 Methodologie en bronnen

Mijn bijdrage aan de discussie over democratische technologie baseer ik in de eerste plaats op interdisciplinair literatuuronderzoek waarvan de hoofdstukken twee, drie en vier het resulaat vormen. Hierbij combineer ik de beschrijvende en historische literatuur over vrije en open broncode software met de theoretische literatuur afkomstig uit het interdisciplinaire onderzoeksveld van de Science and Technology Studies (STS). Daarbij moet worden opgemerkt dat open software een zodanig nieuw onderwerp binnen het STS-veld is, dat theoretische literatuur over dit onderwerp op dit moment pas in kleine hoeveelheden begint te verschijnen.[7]

Op de tweede plaats, maar zeker niet minder belangrijk, baseer ik deze scriptie op empirisch onderzoek. Om een degelijk fundament te kunnen leggen voor mijn scriptie heb ik onderzoek gedaan naar de gemeenschap rondom het open source computerprogramma Vim. Mijn keuze voor Vim als representatief onderzoeksobject baseer ik op de volgende overwegingen.

1. Vim is geen obscuur computerprogramma, maar een veelgebruikte en populaire teksteditor, die onder professionele computergebruikers een grote naamsbekendheid geniet en verschillende 'awards' toegekend kreeg.[8] Binnen de 'canon' van open source programmatuur neemt Vim een belangrijke plaats in. Daarnaast bestaat Vim al zo'n vijftien jaar en is daarmee ouder dan bijvoorbeeld het bekendere Linux-project. Deze hoge leeftijd toont aan dat Vim geen eendagsvlieg is, maar op een duurzame wijze voorziet in een reële behoefte.

2. Vim is beschikbaar voor een groot aantal computerplatformen, zoals Unix, Linux, Windows en MacOs. Selectiebias veroorzaakt door de gebruikerscultuur van slechts één specifiek besturingssysteem (bijv. het commerciële Windows of het anarchistische Linux) te benaderen is daardoor uitgesloten.

3. Vim is een overzichtelijk project om te bestuderen, en niet zo grootschalig, complex en onoverzichtelijk als bijvoorbeeld het Linuxkernel-project, waaraan tienduizenden programmeurs meehelpen. De Vim-gemeenschap is vooral gecentreerd rondom één website, een handvol officiële mailinglijsten en de hoofdontwikkelaar Bram Moolenaar. Hierdoor kan ik ondanks mijn beperkte middelen en tijd, toch een goed inzicht krijgen in de dynamiek van deze gebruikers- en ontwikkelgemeenschap.

4. De Vim-gemeenschap staat bekend om haar vriendelijke en gemoedelijke sfeer,

[7] Van Ratto verschijnt dit jaar zijn promotie-onderzoek over Linux. Ook van Von Hippel verschijnt dit jaar een interessant boek over open source software en gebruikersinnovatie. Een andere goede bron voor theoretische artikelen over open source software is het peer-reviewed e-journal First Monday.
[8] Zie: http://www.moolenaar.net/vim.html#awards

waardoor ik voor mijn onderzoek verzekerd was van actieve respons.

5. Hoofdontwikkelaar Bram Moolenaar woont in Nederland en is daardoor goed benaderbaar voor interviews.

Voor mijn onderzoek naar Vim heb ik een veelheid aan verschillende bronnen geanalyseerd:
- het programma Vim zelf (voor Windows, MacOS, Linux en FreeBSD)
- de Vim-broncode
- de Vim-licentie
- de bijgeleverde Vim-tutor (een soort introductiecursus voor een effectief gebruik van Vim)
- de zeer uitgebreide interne helpfunctie van Vim (met honderden pagina's informatie)
- de Vim 'Frequently Asked Questions' (FAQ) lijst
- interviews met en artikelen over en door de hoofdprogrammeur van Vim, Bram Moolenaar
- de Vim-mailinglijsten (vim@vim.org, vim-dev@vim.org) en het Vim-archief op internet beschikbaar op http://groups.yahoo.com/group/vim/
- de Vim-website (www.vim.org).

Daarnaast heb ik een virtueel focusgroeponderzoek uitgevoerd met leden van de Vim mailinglijst. Bij het uitvoeren van een virtuele 'focus group study' formeert de onderzoeker een groep van representatieve leden uit een bepaalde gemeenschap, om zo gedeelde normen en waarden van een bepaalde gemeenschap te kunnen ontdekken, in dit geval dus die van de Vim-gemeenschap.

Op 3 december 2002 startte ik mijn onderzoek, door het plaatsen van een oproep op de Vim e-maillijst vim@vim.org[9]. In deze oproep legde ik uit op zoek te zijn naar Vim gebruikers die willen meewerken aan een onderzoek naar de gebruiks- en ontwikkelpraktijk van Vim. Daarbij vermeldde ik duidelijk wat er met de verzamelde gegevens zou gaan gebeuren, hoe lang het onderzoek zou gaan duren, en dat ik zorgvuldig met persoonlijke gegevens om zou gaan[10]. Vervolgens heb ik achttien dagen zeer intensief met 22 respondenten gemaild. Op 20 december 2002 heb ik de eerste bevindingen van mijn onderzoek gepubliceerd op de Vim e-maillijst[11].

Het is overigens erg belangrijk om de specifieke voor- en nadelen van een virtuele focusgroep studie ten aanzien van een conventionele opzet in het oog te houden, zoals weergegeven in onderstaande tabel.

[9] Zie: http://groups.yahoo.com/group/vim/message/34898
[10] Jones (1999) stelt aandacht voor de ethische kanten van onderzoek op internet als een van de belangrijkste overwegingen centraal.

Samenvatting sterke en zwakke kanten van virtuele focus groepen (naar: Bloor 2001: 83)

Voordelen virtuele focus groepen	**Nadelen virtuele focus groepen**
- Snel en goedkoop	- Vereist een bepaald niveau van technische expertise
- Biedt makkelijk toegang voor zowel onderzoeker als participanten	
- Biedt toegang tot verspreide, immobiele en moeilijk bij elkaar te krijgen populaties	- Erft populatie-bias van internetgebruikers
- Bevordert ontboezemingen over sensitieve onderwerpen	- Het is moeilijk om bedrog of gevoelig liggende zaken op te pikken
- Reductie van het interviewer effect	- Het kan moeilijk zijn om tot een uiteindelijke terugrapportage te komen
- Data zijn al op schrift aanwezig, zonder transcriptiefouten	- Data bevatten geen non-verbale aanwijzingen of informatie
- Stemt overeen met de conventies rondom natuurlijk voorkomende interactie via computercommunicatie	

Overigens was in er dit geval geen andere mogelijkheid dan het gebruik van een virtuele focus studie, omdat de Vim-gemeenschap alleen in virtuele vorm bestaat op internet. Gebruikers en ontwikkelaars zijn over de hele wereld verspreid en communiceren alleen via e-mail op internet. Slechts een heel klein gedeelte ontmoet elkaar sporadisch in levende lijve op congressen en bijeenkomsten over open software. De kans op gebruikersbias door mijn selectie van respondenten tot internetgebruikers te beperken is daardoor uitgesloten. Deze selectie zit al in Vim zelf ingebouwd, want voor een effectief gebruik (bijvoorbeeld voor nieuwe versies of gebruikersondersteuning) is een internetaansluiting vereist.

Sociologisch onderzoek doen op internet vond ik een leerzame, stimulerende en verrijkende ervaring. In mijn rol als lid van de Vim e-maillijst heb ik tijdelijk ervaren hoe het is, om een Vimgebruiker te zijn. Het is wonderbaarlijk om te zien hoe gedurende de hele dag nieuwe e-mails met vragen over Vim je computer binnenstromen, die vervolgens binnen een half uur beantwoord worden door andere Vimgebruikers. Ditzelfde enthousiasme en deze snelle respons viel ook mijn onderzoek ten deel. Hierbij moet ik echter wel opmerken dat

[11] Zie: http://groups.yahoo.com/group/vim/message/35325

diepgravend onderzoek alleen werkt met gemotiveerde respondenten. Veel tijd investeren om antwoorden uit respondenten te trekken wanneer ze daar geen zin in hebben is niet lonend. Toen ik bijvoorbeeld probeerde om een zeer actieve e-mailbeantwoorder op de Vim e-maillijst te overtuigen om mee te doen aan mijn onderzoek, ging hij aanvankelijk schoorvoetend akkoord, om later alsnog af te haken.

Respondenten daarentegen die uit zichzelf reageerden, stuurden vaak binnen enkele uren of soms zelfs binnen een kwartier, lange uitwijdingen van meerdere a4-tjes terug. Vervolgens eindigden ze hun e-mail met de opmerking dat ze hoopten dat ze niet te langzaam reageerden en wel genoeg geschreven hadden. Intrinsieke motivatie van respondenten bij een dergelijke virtuele studie is een factor van cruciaal belang voor het slagen van het onderzoek.

Daarnaast merkte ik dat veel respondenten afhaken wanneer je ze vraagt om een vragenlijst per e-mail in te vullen en terug te sturen[12]. Daarom ben ik hier snel mee gestopt en overgestapt op een meer persoonlijke en interactieve e-mailcorrespondentie, waarin per e-mail slechts een paar verschillende onderwerpen ter sprake komen. Op deze wijze kreeg ik de meest uitgebreide reacties en bleven respondenten tot aan het einde aan mijn onderzoek meewerken. Dit had echter ook tot gevolg dat ik op een gegeven moment zo'n twintig mensen tegelijk per e-mail interviewde, hetgeen dit proces voor de onderzoeker erg arbeids- en tijdsintensief maakte.

Als laatste wil ik nog opmerken dat het grote pluspunt van de omgeving van mijn onderzoek, namelijk de informele en laagdrempelige informatie-uitwisseling via een e-maillijst, tegelijkertijd ook een groot nadeel kon vormen. Zo gemakkelijk als het was om aan persoonlijke reacties te komen, zo moeilijk was het om aan formelere informatie te komen. Op een simpele vraag over het aantal abonnees van de algemene Vim e-maillijst, kreeg ik ook na meerdere oproepen geen antwoord[13]. Niemand wist het, of wie het zou kunnen weten, of niemand wilde reageren op mijn oproep. Op zo'n moment is het ontbreken van een centraal centraal aanspreekpunt voor een verzoek om informatie erg frustrerend.

Niettemin was ik positief verrast door de kwantiteit en de kwaliteit van alle reacties. Respondenten waren erg geïnteresseerd en gemotiveerd en leverden soms zelfs spontane bijdragen in de vorm van interessante opmerkingen en observaties. Zonder hun enthousiaste medewerking had ik nooit een gedetailleerd inzicht in de Vim-gemeenschap kunnen krijgen.

[12] Zo beschrijven Franke & Hippel (2002: 12) in hun case-study naar de open source webserver Apache een emailrespons van zeventien procent.
[13] Zie: http://groups.yahoo.com/group/vim/ message/34944

§ 1.3 Structuur

Om de vraag naar een democratische computertechnologie te kunnen beantwoorden, is deze scriptie als volgt georganiseerd:

Hoofdstuk twee biedt een inleiding tot de theoretische literatuur over het democratische gehalte van technologie. Dit hoofdstuk vormt de theoretische achtergrond waarbinnen de relevantie van het onderzoek in deze scriptie naar de openheid dan wel geslotenheid van computerprogramma's voor gebruikers gezien moet worden. Daarnaast introduceer ik in dit hoofdstuk mijn theoretische gereedschapskist, met daarin de theoretische concepten waarmee ik inzicht probeer te krijgen in hoe de openheid of geslotenheid van software het gebruik ervan kan vormgeven.

In hoofdstuk drie schets ik op welke wijze een computerprogramma gesloten kan zijn aan de hand van voorbeelden uit de praktijk. Daarnaast behandel ik de negatieve gevolgen van gesloten software voor een democratisch computergebruik.

In hoofdstuk vier beschrijf ik kort de ontstaansgeschiedenis van open software in reactie op gesloten software. Daarnaast laat ik zien hoe open computerprogramma's een democratischer computergebruik faciliteren, gebruikersparticipatie in het ontwikkelproces mogelijk maken en er voor zorgen dat er een uitgebreid ondersteunend netwerk kan ontstaan van hybride gebruiker/makers.

In hoofdstuk vijf bespreek ik mijn empirische onderzoek naar het 'open' computerprogramma Vim, waarbij ik gebruik heb gemaakt van literatuuronderzoek, een retorische analyse van het computerprogramma Vim en de bijbehorende internetgemeenschap, interviews met de hoofdontwikkelaar en een virtueel focusgroeponderzoek van Vimgebruikers en -ontwikkelaars.

In het laatste hoofdstuk beantwoord ik naar aanleiding van mijn bevindingen uit de voorgaande hoofdstukken drie, vier en vijf de probleemstelling uit het eerste hoofdstuk. Daarnaast werp ik vragen op voor verder onderzoek en doe ik aanbevelingen voor een democratischer gebruik van computertechnologie waardoor een dystopische digitale dictatuur nooit werkelijkheid hoeft te worden. Zo keer ik weer terug naar het meer theoretische niveau van democratische technologie in het algemeen uit hoofdstuk twee.

2. Theoretisch kader voor een democratische technologie

§ 2.1 Inleiding

In deze scriptie pleit ik voor een wereld waarin technologie mogelijkheden schept, in plaats van beperkingen; voor een wereld waarin mensen door nieuwe technologie worden ingesloten, in plaats van buitengesloten; en voor een wereld waarin vrijheid en democratie fundamenteel belangrijker zijn dan winst en commercie. Hierbij beperk ik mij, gesteld door de grenzen van een doctoraalscriptie, tot empirisch onderzoek naar en een theoretische analyse van de technologie van computerprogramma's en hoe zij grenzen creëert waardoor ruimtes ontstaan waarbinnen gebruikersparticipatie juist wel of niet mogelijk is. Binnen de wetenschap zijn er echter al verschillende denkers uit verschillende disciplines, zoals bijvoorbeeld binnen de techniekfilosofie, techniekgeschiedenis, cultuurfilosofie en politieke filosofie, die zich hebben beziggehouden met het thema democratie en technologie. Mijn werk over democratische technologiegebruikers is dan ook niet in een vacuüm geschreven, maar dient geplaatst te worden binnen deze theoretische context, die ik in dit hoofdstuk bespreek.

§ 2.2 Denken over technologie en democratie

1. Technologie is inherent aan mens-zijn

Wat mensen zo bijzonder maakt ten opzichte van andere zoogdieren, is dat zij letterlijk een hele nieuwe wereld om hen heen hebben gecreëerd, namelijk de wereld van de technologie. Overal om hen heen is technologie, zelfs de 'natuur' van een typisch Nederlands polderlandschap is door mensen zelf gemaakt. Ook cultuur komt voort en is afhankelijk van technologische vindingen: muziek wordt gemaakt met behulp van muziekinstrumenten (van simpele trommel en fluit, tot een Stradivariusviool of Moog-synthesizer), er is geen literatuur zonder moderne drukpers of schrijfmachine (van ganzenveer tot moderne laptop) en geen kunst zonder verf (van simpel krijtje tot via ingewikkelde chemische procédés gefabriceerde hoogwaardige pigmenten). Zelfs bij de voortplanting, misschien wel de meest 'natuurlijke' activiteit van de mens, speelt technologie een belangrijke rol (van anticonceptiepil en condoom, tot in vitro fertilisatie).

De moderne mens leeft dan ook niet langer in een biotoop, maar in een technotoop. Een

kort voorbeeld hiervan geeft de techniekfilosoof Don Ihde, wanneer hij de mens-technologie interacties beschrijft van een denkbeeldige hoofdpersoon vanaf zijn wakker worden uit een ontspannen slaap: Hoe de alarmklok hem wekt met zijn gepiep, terwijl hij nog lekker geniet van de warmte van de elektrische deken of van zijn synthetische dekbed. In de badkamer voorziet een ingenieus leidingensysteem hem van water. Even later pruttelt het koffiezetapparaat, terwijl de broodrooster en de magnetron hun werk doen. Daarna brengt de auto of een ander ingewikkeld transportsysteem hem naar zijn werk, terwijl de autoradio hem ondertussen voorziet van informatie over het wereldnieuws en belangrijker, de laatste filemeldingen (Ihde, 1996). Kortom: het gebruik van technologie is een fundamenteel onderdeel van ons mens-zijn.

2. Technologie is sociaal geconstrueerd

Mensen zijn een allesomvattend verbond aangegaan met de dingen, waarbij de dingen net zo veel invloed uitoefenen op de mensen, als vice versa. Technologie is geen neutraal gereedschap, dat je voor een doel gebruikt en daarna weer weglegt, zoals veel mensen denken. Om technologie te kunnen laten functioneren zijn uitgebreide infrastructuren en ondersteunende systemen nodig, die ingrijpende gevolgen hebben voor zowel onze fysieke leefwereld als de sociale organisatie van onze samenleving.

Veel mensen beschouwen technologie echter nog steeds als iets neutraals, dat zich ‘vanzelf’ en buiten de samenleving ontwikkelt. Ondanks het feit dat technologie een grote invloed op ons dagelijks leven heeft, lijken de meeste mensen zich daar niet van bewust. Pas wanneer iets niet meer werkt - wanneer de stroom uitvalt of een computer vastloopt - beseffen we hoe afhankelijk we van technologie zijn. Zodra alles weer werkt, verdwijnt dit inzicht weer als sneeuw voor de zon. Hoe complexe technologie precies werkt of wat haar ontwikkelingsgeschiedenis is, vraagt bijna niemand zich dan nog af.

Dit dominerende ‘common sense’ standaardbeeld van technologie-ontwikkeling, wordt technologisch determinisme genoemd. In de definitie van het technologisch determinisme volgens MacKenzie en Wajcman (1985: 4-5) staan twee zaken centraal. Ten eerste ontwikkelt technologie zich autonoom, letterlijk of figuurlijk buiten de samenleving om. Ten tweede veroorzaken technologische veranderingen maatschappelijke veranderingen.[14] Voorbeelden van uitspraken die uitgaan van een technologisch deterministisch wereldbeeld zijn bijvoorbeeld “de pil is de oorzaak van de seksuele revolutie”, “de magnetron en de televisie zorgen voor een verschraling van het gezinsleven”, of “dankzij nieuwe informatie-

en communicatietechnologie ontstaat er een wereldomvattende netwerkmaatschappij."

Alhoewel het technologisch-deterministisch wereldbeeld goed samenvalt met de dagelijkse ervaringen van mensen met technologie, kleven er ook een aantal problemen aan dit conceptuele model (Wyatt 1998: 11). Behalve dat dit simplistische model geen recht doet aan de complexiteit van de ontwikkeling en invoering van nieuwe technologie, is er in dit model ook geen ruimte voor mensen om actief een keuze te maken welke technologie zij accepteren en welke zij verwerpen. Hiermee vervalt ook het afleggen van verantwoording voor de gevolgen van het gebruik van bepaalde technologieën, of dit nu kerncentrales zijn, elektrische koelkasten met cfk's als koelmiddel of Microsoft Windows 98. Binnen dit model is het niet mogelijk om vraagtekens te plaatsen of kritiek te leveren op de individuele en collectieve keuzes die we maken over de snelheid en de richting van technologische veranderingen en de bijbehorende maatschappelijke gevolgen.

Voortbouwend op het technologisch determinisme is het "neutrale technologie"-model ontstaan (ibid.: 11). In dit model ontwikkelt technologie zich buiten de maatschappij om, maar vervolgens kiezen mensen zelf of zij deze technologie accepteren en op welke manier zij haar gebruiken. De centrale gedachte hierbij is, dat de technologie zelf neutraal is. Een sprekend voorbeeld van deze redenatie zien we terug in het motto van de Amerikaanse National Rifle Association: "Guns don't kill people. People kill people." Technologie zelf is niet inherent slecht of goed, het zijn de gebruikers die een technologie voor het realiseren van goede of slechte doeleinden gebruiken. Het gevolg van deze redenatie is dat de maatschappelijke implicaties van het gebruik van technologie volledig afhankelijk zijn van onze eigen keuzes over het gebruik ervan. Hiermee verwerpt dit model het imperatieve karakter dat technologie oplegt aan de samenleving. Maar dit model morrelt niet aan de ontwikkeling van technologie zelf, omdat deze zich volgens dit model nog steeds buiten de samenleving bevindt, en dus ook niet beïnvloedbaar is door krachten of tendensen uit deze samenleving.

Toch is ook de eerste stelling van het technologisch determinisme, dat technologie zich buiten de samenleving ontwikkelt volgens een eigen technische en wetenschappelijke logica, vrij van politieke, sociale en economische invloeden, niet langer houdbaar. En hiermee komen we op het terrein van sociaal-constructivistische opvattingen over technologie-ontwikkeling. Het zijn namelijk mensen die technologie maken, en deze mensen zijn niet los te zien van de samenleving waartoe ze behoren. Het hedendaagse techniekonderzoek probeert

[14] Voor een genuanceerd overzicht over het technologisch determinisme zie Smith & Marx 1994.

het standaardbeeld van technologie-ontwikkeling te vervangen door een genuanceerder beeld, waaruit blijkt dat technologie midden in de samenleving staat en dat het zelfs niet mogelijk is om een scheiding te maken tussen de samenleving enerzijds en technologie anderzijds.

Bijker (1997) beschrijft op heldere wijze de opkomst van het moderne techniekonderzoek. Tot ongeveer het midden van de jaren tachtig werd techniek door economen en sociologen vooral als "black box" beschouwd, waarbij de inhoud van deze black box en hoe deze tot stand was gekomen buiten beschouwing bleef. Historici keken daarbij wel naar de inhoud van de black box, maar zij zaten zodanig "in" de black box, dat zij niet naar buiten keken, waardoor het niet mogelijk was om tot nieuwe casusoverstijgende methodische en theoretische inzichten te komen.

Begin jaren tachtig vinden volgens Bijker vervolgens drie elkaar versterkende ontwikkelingen plaats. In de eerste plaats kijken economen steeds meer naar de inhoud van technologie. Op de tweede plaats werpen sociologen steeds vaker een blik in de black box van technologie, geïnspireerd door ontwikkelingen in de wetenschapssociologie waarbij wordt gekeken hoe kennis en wetenschappelijke feiten worden "gemaakt". Ten derde houden historici zich steeds meer bezig met modelvorming en theoretische verklaringen. Deze ontwikkelingen binnen afzonderlijke disciplines versterken elkaar door een toenemende samenwerking tussen techniekonderzoekers afkomstig uit verschillende disciplines. Wanneer in 1984 wetenschappers uit verschillende disciplines elkaar ontmoeten op een internationale workshop in Enschede, wordt daar de basis gelegd voor het interdisciplinaire onderzoeksveld van het techniekonderzoek, met als centrale onderzoeksvraag het inzicht verkrijgen in de ontwikkeling van technologie.

> "De nieuwe techniekgeschiedenis stelt zich ten doel, een reconstructie te geven van het complex van sociale, culturele, technische en natuurlijke omstandigheden om daarmee de ontwikkeling van een technisch apparaat te kunnen verklaren. ... De drie benaderingen stemmen met elkaar overeen in het 'opblazen' van het onderscheid tussen technische en culturele, tussen wetenschappelijke en economische, tussen cognitieve en sociale factoren: het weefsel van techniek en samenleving is naadloos. Sociale interacties vormen de primaire ingang voor onderzoek en er wordt voor gewaakt, de natuur of het technisch werkend zijn als 'machina ex deo' een ontwikkeling te laten verklaren.(Bijker 1987: 23)"

Een van de belangrijkste inzichten uit het moderne techniekonderzoek is dat technologie geconstrueerd wordt, waardoor het niet mogelijk is om een onderscheid te maken tussen technische en culturele, wetenschappelijke en economische invloeden. Dit heeft als gevolg dat we daardoor verplicht zijn om na te denken over de implicaties van het ontwerp van technologie-ontwikkeling. En wie hier eigenlijk allemaal wel of geen invloed op uit kan oefenen. Zo komen we uit bij de relatie tussen politiek en technologie.

3. Technologie ís politiek

Langdon Winner waarschuwt dat technologie niet neutraal is, maar een vorm van politiek met andere middelen. Technologische artefacten kunnen politieke standpunten in zich opnemen, om die vervolgens actief en dwingend uit te dragen. Als voorbeeld van deze stelling, verwijst Winner naar de bruggen die door de New Yorkse stadsplanner Robert Moses doelbewust zo laag werden gebouwd, dat er geen bussen onder door konden rijden[15]. Dit had tot gevolg dat bepaalde delen van de stad onbereikbaar werden voor het openbaar vervoer. En daarmee dus ook voor de voornamelijk zwarte gebruikers ervan (Winner 1980). Winner stelt vervolgens de vraag:

> "What kind of world are we building here? As we develop new devices, techniques and technical systems, what qualities of social, moral and political life do we create in the process? Will this be a world friendly to human sociability or not? (Winner 1991: 284)"

Uitgaande van de redeneertrant van Winner, is het van belang om er voor te waken om niet in in een technologische dictatuur terecht te komen. Voor de negatieve gevolgen van de wisselwerking tussen de technische en sociale ontwikkeling, heeft Lewis Mumford als een van de eerste techniekfilosofen al gewaarschuwd. Mumford onderscheidt twee verschillende soorten technologie, die naast elkaar bestaan en verschillende uitwerkingen hebben op de structuur van menselijke samenlevingen (Mumford 1964: 2-3). De oudste technologievorm is de democratische technologie. Zij is mens-georiënteerd, relatief zwak, maar duurzaam en stabiel. Een relatief recentere uitvinding van een paar duizend jaar geleden is de autoritaire technologie. Zij is systeem-georiënteerd, erg krachtig, maar ook erg onstabiel. Mumford waarschuwt ons dat we op een punt in de geschiedenis staan, waarop als we niet oppassen, de democratische technologie verdwijnt door de opkomst van gecentraliseerde, autoritaire technologische systemen, waarvoor Mumford de term 'megamachine' gebruikt. Met behulp van dit concept probeert hij de ontsporing en de waanzin van veel moderne technologische ontwikkelingen te begrijpen en aan de orde te stellen (Achterhuis 1992: 234).

Volgens Mumford ontstond de megamachine grotendeels gelijktijdig in de hogere culturen in de rivierdelta's van Egypte, Mesopotamië en het verre Oosten evenals in Midden- en Zuid-Amerika. De megamachine is zowel een arbeids- als oorlogsmachine, berust op fysieke dwang en slavenarbeid en bezit het equivalent van duizenden paardenkrachten. Hierdoor maakt zij tegelijkertijd zowel massa-productie als massa-destructie op een ongekend grote schaal mogelijk. Ook maakt zij een economie van overvloed mogelijk,

[15] Joerges (1999) bekritiseert het canonische verhaal van Winner over Moses door de empirische bewijzen te problematiseren. Woolgar & Cooper (1999) bepleiten waarom dit minder erg is dan het lijkt en analyseren het succes van de casus van Moses binnen Science Technology and Society Studies.

alhoewel dit wel ten koste gaat van het leven (zowel mens als natuur) in het algemeen.

Deze 'megamachine' is in de eerste plaats maatschappelijk georganiseerd en heeft daarom in de geschiedenis geen zichtbare materiële sporen nagelaten. Opgravingen laten alleen de resten van werktuigen zien, maar de machine waarbinnen deze functioneerden blijft onzichtbaar. Mumford gebruikt de megamachine om de bouw van de wereldwonderen uit de Oudheid mee te verklaren. Deze zijn immers niet primair dankzij werktuigen ontstaan, maar door de aanwezigheid van de sociale megamachine. Verantwoordelijk ervoor waren koningen, een nieuw soort machthebbers die in vroegere, minder complexe gemeenschappen, onbekend waren. Ondersteund door priesters die over astronomische kennis beschikten, slaagden zij erin losse individuen en groepen samen te brengen in een zeer grootschalig geheel, dat perfect op bepaalde taken en functies was toegesneden (ibid.: 244-45).

Tegenwoordig wordt de megamachine ingezet voor de ontwikkeling van de atoombom of om mensen op de maan te kunnen laten landen. De hedendaagse megamachines zijn de moderne technisch-industriële complexen, die de effectiviteit en efficiëntie van de antieke voorbeelden op veel grotere schaal overtreffen. Daarbij zijn ook de destructieve krachten en de insluiting van de mens in de machine, vergeleken met de oudere voorbeelden, enorm toegenomen (ibid.: 249).

De sombere boodschap van Mumford is dat de huidige economische vrijheden een gevolg zijn van moderne autoritair-technologische systemen. In feite is de moderne vrijheid niets meer dan een veel verfijndere vorm van de oude slavernij. De opkomst van de politieke democratie gedurende de laatste eeuwen wordt tenietgedaan door de succesvolle wederopstanding van de gecentraliseerde autoritaire technologieën. Terwijl in het westen de eens almachtige koning van zijn troon werd verstoten en van zijn macht werd ontdaan, hebben we ditzelfde machtssysteem weer gerestaureerd, maar dan in een technologische en veel effectievere vorm. De uitvinders van de atoombom, de ruimteraket en de computer zijn dan ook de piramidebouwers van onze tijd. Dit autoritaire gecentraliseerde systeem streeft een dwangmatige uitbreiding na, onafhankelijk van de eventuele kosten voor het leven op aarde. Het centrum van de macht ligt in deze nieuwe systemen niet langer bij een zichtbare persoonlijkheid, zoals een almachtige koning, dictator of 'democratisch' gekozen president, maar is alom aanwezig in het systeem zelf (Mumford 1964).

4. De rol van computertechnologie binnen de samenleving

Informatie- en communicatietechnologie (ICT) neemt een steeds belangrijkere plaats in onze samenleving in. Sommigen beweren zelfs dat onze samenleving dankzij nieuwe computertechnologie zodanig ingrijpend zal veranderen, dat we met recht kunnen spreken van een compleet nieuw tijdperk. Zo heeft MacLuhan (1967) het over 'the Global Village'; Toffler (1980) schrijft over de 'Informatiesamenleving' en Castells (1996, 2001) introduceert de termen 'Network Society' en 'Internet Galaxy'.

Terwijl McLuhan de komst van de 'Global Village' via de telematica als één grote belofte beschouwt, en Alvin Toffler de gevolgen van de 'Informatiemaatschappij' bijna exclusief positief waardeert, wijst Mumford voortdurend op de ambivalentie van de te verwachten technologische ontwikkelingen, waarbij de nieuwe informatietechnologie alleen maar de bestaande destructieve verhoudingen zal versterken (Achterhuis 1992: 218).

De negatieve dan wel positieve invloed van nieuwe informatietechnologie op de samenleving voorspellen is onmogelijk. Niettemin is het van belang om ons af te vragen in hoeverre de samenleving iets te zeggen wil hebben over hoe en wanneer veranderingen plaats zullen vinden, wat daar de maatschappelijke gevolgen en kosten van zullen zijn, voor wie die van belang zullen zijn, en of die veranderingen überhaupt gewenst zijn. Het is inmiddels duidelijk dat computers een grote invloed op het dagelijks leven hebben en dat die invloed waarschijnlijk alleen maar zal toenemen. In steeds meer apparaten worden 'chips' en micro-electronica ingebouwd, en bijna niemand werkt meer zonder in meer of mindere mate gebruik te maken of zelfs afhankelijk te zijn van computertechnologie. De vraag is in hoeverre de samenleving op dit moment zelf actief haar toekomst kan vormgeven en ervoor kan zorgen dat iedereen zich thuis voelt in de wereld van morgen, zonder dat mensen uitgesloten of 'geleefd' worden door computertechnologie waar ze zelf niet voor gekozen hebben. Een somber beeld ontstaat wanneer de megamachines na Mumford ook nog eens beschikken over moderne informatie- en communicatietechnologie om de verschillende menselijke onderdelen te kunnen controleren en commanderen op een tot daarvoor ongekend grote schaal, waarbij de controle zoals we die kennen uit het 'panopticon' van Foucault volledig verbleekt.

5. Recht op democratische technologie

Tegenwoordig is iedereen het er over eens dat mensen het recht moeten hebben om inspraak te hebben op de politiek op zowel locaal, regionaal, nationaal als internationaal niveau, door middel van het stemrecht (zowel actief als passief). Zoals ik al eerder heb laten zien is

technologie ook een vorm van politiek. Daarom zou ook iedereen het recht moeten bezitten om invloed uit te kunnen oefenen op de ontwikkeling en het gebruik van technologie. Alleen op deze manier kunnen we voorkomen om in een technologische dictatuur terecht te komen.

Met deze vraag naar een democratische technologie, houdt Sclove zich in zijn boek 'Technology and democracy' (1995) bezig. Daarin introduceert hij het begrip democratische technologie als oplossing om te voorkomen dat we terechtkomen in een gecentraliseerde, autoritaire 'megamachine' zoals Mumford die beschrijft. In de inleiding van zijn boek vat hij zijn theorie kernachtig samen:

> "Insofar as (1) citizens ought to be empowered to participate in shaping their society's basic circumstances and (2) technologies profoundly affect and partly constitute those circumstances, it follows that (3) technological design and practice should be democratized. (Sclove 1995: ix)"

Sclove wijst ons op het democratisch deficit, dat op dit moment bestaat op het gebied van het ontwerp en gebruik van technologie in onze samenleving. Als een handreiking om te laten zien hoe het beter zou kunnen, heeft Sclove een lijst met ontwerpcriteria ontwikkeld waaraan technologie moet voldoen voordat zij democratisch genoemd kan worden. Democratische technologie moet volgens Sclove aan de principes van de "strong democracy" voldoen, namelijk rechtstreekse participatie van burgers op lokaal niveau. Deze "strong democracy" staat tegenover een representatieve democratie, waarin het volk vertegenwoordigers kiest die op een hoger niveau de stem van het electoraal uitdraagt.

De aanleiding voor Sclove voor het schrijven van zijn boek is vooral zijn bezorgdheid over de (vaak onbedoelde, maar meestal ook ondoordachte) negatieve effecten die bepaalde technologieën voor de samenleving met zich mee kunnen brengen. Als paradigmatisch voorbeeld verwijst Sclove naar wat er gebeurde in het Spaanse dorpje Ibieca toen daar in de jaren zeventig waterleiding in de huizen werd geïnstalleerd, zodat dorpelingen niet langer naar de centrale fontein hoefden te lopen om water te halen (Sclove 1995: 3). Je zou denken dat deze watermodernisering alleen positieve effecten zou hebben, maar vooral op het sociale vlak bracht deze verandering ongewenste gevolgen met zich mee. Doordat de dorpelingen niet langer water buiten de deur hoefden te halen, ging de functie van de gemeenschappelijke waterfontein en (af)wasplek als sociale ontmoetingsplek verloren, waardoor de sterke sociale cohesie tussen de dorpelingen verloren ging.

In onze moderne wereld worden we overspoeld door technologische innovaties, zonder dat we van tevoren weten wat daarvan de sociale gevolgen voor de samenleving zullen zijn. Een voorbeeld is de komst van de auto met haar bijbehorende infrastructuur van snelwegen, autogarages en benzinepompen. Natuurlijk is iedereen erg blij met de toegenomen mobiliteit die auto's mogelijk maken, maar daarvoor moet wel een prijs betaald worden in de vorm van

toegenomen geluidsoverlast, zure regen, uitstoot van roet- en stofdeeltjes, milieurampen met olietankers, fileproblemen, verkeersdoden en gewonden, toegenomen afhankelijkheid van olieproducerende landen en de verpaupering van stadscentra door suburbanisatie. Het gebruik van nieuwe technologieën voor individueel gebruik, brengt vaak onvoorziene en ongewenste gevolgen met zich mee voor de samenleving als geheel.

Ook al controleren tegenwoordig instanties of nieuwe technologieën aan de regelgeving voldoen op het gebied van milieuverontreiniging of veiligheid in het gebruik, toch is dat volgens Sclove niet genoeg. Niemand kijkt naar het gehele bereik van psychologische, culturele en politieke effecten van technologieën (ibid.: 4). Volgens Sclove zijn de onvoorziene sociale effecten van nieuwe technologie niet zodanig complex dat er niet op geanticipeerd of zonder beleid mee omgegaan zou kunnen worden. Als centrale voorbeeld van hoe het beter zou kunnen, noemt Sclove in zijn boek de gemeenschap van de Amish in de Verenigde Staten. Deze gemeenschap, die vooral bekend is als religieuze subcultuur van mensen die in ouderwetse klederdracht per paard en wagen reizen, kiest bewust voor zowel de adaptatie van nieuwe technologie als het uitsluiten van bepaalde andere technologieën. Hierbij vraagt de Amish-gemeenschap zoals beschreven door Sclove zich af wat de effecten van het invoeren van een bepaalde technologie voor de gemeenschap als geheel zullen zijn. Technologieën die de waarden van de gemeenschap versterken, zoals de sociale cohesie, de religie en de respectvolle omgang met de natuur, worden ingevoerd en gebruikt. Wanneer een nieuwe technologie echter tegen de principes van de Amish indruist wordt zij verworpen. Dit alles wordt volgens Sclove op open en democratische wijze bediscussieerd en besloten.

Daarbij benadrukt Sclove om niet alleen naar aparte technologieën kijken, maar naar het geheel van interacties tussen verschillende technologieën. In dat geval zou de samenleving tot het inzicht komen dat veel technologieën bijdragen aan een veelheid van moderne problemen zoals eenzaamheid, egocentrisme, disempowerment, onveiligheid, stress en vervreemding. Technologie kan zelfs bijdragen aan het in stand houden van antidemocratische machtsrelaties en bij het eroderen van de sociale ruimte voor het ontwikkelen en uitdrukken van burgerschap.

Hierbij is de technologie zelf niet de oorzaak voor al deze problemen, maar zij draagt er wel aan bij. Wanneer de brede sociale dimensies van technologische effecten genegeerd blijven, zullen onze sociale problemen ook niet opgelost kunnen worden. De moderne technologieën dragen bij aan de huidige sociale orde en verhinderen zo sociale vernieuwingen. Apparaten als auto's, computers, televisies, walkmans veranderen ons gedrag als individu in een samenleving. Toch heeft de samenleving niet expliciet gekozen voor een

bepaalde technologie of haar bijkomstige effecten.

Het punt dat Sclove wil maken gaat over reflexiviteit, namelijk kritisch reflecteren op technologische innovaties, hun sociale gevolgen en de manier waarop de samenleving daar mee omgaat en mee om wil gaan. In onze hedendaagse samenleving lijkt deze reflectie inderdaad te ontbreken of veel minder sterk ontwikkeld in vergelijking met bijvoorbeeld de zelfbewuste manier waarop de Amish als gemeenschap om lijkt te gaan met technologie:

> "Finally, I explore political steps and strategies that can help us achieve citizenship in a future world of democratic technology. That world—so unlike today's and within our reach—would witness technological evolution becoming subordinate to democratic prerogatives. It would be a world made by people but also for people, acting under circumstances more favorable to fair and informed results." (Sclove 1995: xi)

"Technology is evil; let's get rid of it", is in ieder geval niet de conclusie die Sclove wil trekken. Mensen kunnen nu eenmaal niet zonder technologie. Technologie moet echter ook niet zomaar kritiekloos gebruikt worden. Er zal gekeken moeten worden welke technologieën overeenkomstig zijn met het soort gemeenschap waarin mensen willen leven (ibid.: 8). De enige oplossing volgens Sclove is dan ook om instanties en organisaties te ontwikkelen die burgers faciliteren om kritisch na te denken en te beslissen over de invoering van nieuwe technologie (ibid.: 9).[16]

§ 2.3 Theoretische gereedschapskist

Na deze vooral filosofische beschouwingen over democratische technologie streef ik in mijn scriptie naar een meer op empirisch onderzoek gebaseerde benadering. In mijn optiek biedt algemeen geldende theoretische techniekfilosofie alléén te weinig aanknopingspunten voor een gedetailleerde verklaring van de gebruikspraktijk van een specifieke technologie. Daarom pleit ik voor onderzoek naar concrete situaties, in mijn geval naar het democratische gehalte van het computergebruik.

Hierbij maak ik gebruik van theoretische concepten afkomstig uit het Actor Netwerk Theorie model[17] en het Social Construction of Technology model. Hierbij staat centraal dat als iets werkt dat niet het uitgangspunt mag zijn, maar juist verklaard dient te worden. Dat software open of juist gesloten is, is geen verklaring voor het democratische gehalte ervan, juist de totstandkoming van deze eigenschappen zelf dient verklaard te worden.

Daarom introduceer ik in deze paragraaf mijn theoretische gereedschapskist met concepten waarmee ik in mijn scriptie probeer te verklaren hoe de openheid en geslotenheid van software tot stand komt en wat daarvan de gevolgen zijn voor de ruimte waarin een

[16] Voor een kritische reflectie op het werk van Sclove zie: Wilde de 1997.
[17] Door Latour ook wel gepreciseerd als 'actant rhizome ontologie', zie: Latour 1999.

democratisch technologiegebruik mogelijk is.

Woolgar (1990) beschrijft in ‘Configuring the user’ de (sociale) constructie van de gebruiker. Met de opkomst van de personal computer bleek meer en meer onderzoek gedaan te worden naar gebruikers van computers. De gebruiker moest bepaald en gedefinieerd worden. Door in het design van hardware en software te anticiperen op mogelijke handelingen van gebruikers, wordt ‘De Gebruiker’ geconstrueerd: “(...) by setting parameters for the user's actions, the evolving machine affectively attempts to configure the user (Woolgar, 1991: 61). Vervolgens analyseert hij kritisch hoe in gebruikstesten bij een software-ontwikkelaar deze constructie plaatsvindt.” (Soogelee 1996: 9)

Akrich (1992) schrijft in ‘The De-Scription of Technological Objects’ over scripts die in artefacten “ge-inscribeerd” worden. Technische objecten staan volgens Akrich niet los van hun omgeving, maar maken altijd deel uit van een heterogeen netwerk, waarvan zowel mensen als objecten deel uitmaken. Volgens Akrich is er geen verschil tussen mensen en niet-mensen, want allebei kunnen over handelingsbekwaamheid (‘agency’) beschikken. De gemeenschappelijke term die Akrich dan ook voor zowel menselijke als niet-menselijke elementen hanteert is ‘actant’. In haar artikel onderzoekt ze vervolgens op welke manier de technische objecten het gedrag en de onderlinge relaties van andere actanten beïnvloed. De kneedbaarheid of de weerbarstigheid van objecten bij de interactie is volgens haar voorgeprogrammeerd door de ontwerpers. Door middel van het begrip script, kan dit proces vervolgens geanalyseerd worden. Akrich definieert script als volgt:

> “Designers thus define actors with specific tastes, competences, motives, aspirations, political prejudices, and the rest, and they assume that morality, technology, science and economy will evolve in particular ways. A large part of the work of innovators is that of ‘inscribing’ this vision of (or prediction about) the world in the technical content of the new object. I will call the end product of this work a 'script' or ‘scenario’ ... Thus like a film script, technical objects define a framework of action together with the actors and the space in which they are supposed to act.” (Akrich 1992: 208)

Het script bepaalt het gedrag van de gebruiker, doordat dit object het handelen voorschrijft. De ontwerper gaat bij het ontwerp uit van het beeld van de potentiële gebruiker en vertaalt zijn ideeën vervolgens in de technische vorm van het object. Maar de ontwerper gaat slechts uit van een beeld van de gebruiker, de zogenaamde ‘perceived user’. In de praktijk kan er echter een verschil optreden tussen de ‘perceived user’ en de ‘real user’, de gebruiker in de echte wereld. Analoog hieraan kan er ook een discrepantie optreden tussen het gebruik zoals de ontwerper dat voor ogen had (‘perceived use’) en de praktijksituatie (‘real use’). “The world described in the object” is dus niet gelijk aan “the world described by its displacement” (ibid.: 209). Vooral wanneer een object een bepaalde gebruiker of een bepaald gebruik

uitsluit, treden de verschillen tussen datgene wat de ontwerper zich voorstelde en de praktijksituatie, duidelijk op de voorgrond.

Overigens is niet ieder script even dwingend qua voorschrijvend handelen. Zo zijn er inscripten waarvan de makers ze zo dwingend mogelijk proberen te maken. Andere scripts daarentegen laten de gebruiker meer ruimte voor handelingsvrijheid. Zo kan een script ook een meer open gebruik bevatten, waarin de gebruiker kan kiezen uit verschillende alternatieven.

Hiermee hangt samen dat gebruikers ook een 'anti-programma' kunnen ontwikkelen. Latour (1992) beschrijft het voorbeeld van een hotelbaas die zijn sleutels niet terugkrijgt van zijn hotelgasten, ook al vraagt hij dit keer op keer netjes aan zijn gasten en heeft hij ter herinnering overal bordjes opgehangen. Toch blijven de meeste gasten de sleutels nog steeds meenemen buiten het hotel. Wanneer de hoteleigenaar vervolgens een verbond aangaat met een zwaar en onhandig loden gewicht, dat hij aan de sleutels bevestigd, ontwikkelen al veel minder hotelgasten een anti-programma. De socio-technische coalitie die de hotelgast tegenwerkt om zijn anti-programma ten uitvoer te brengen wordt steeds sterker. Niettemin verhindert het loden gewicht nog steeds niet dat er enkele hotelgasten zijn die de sleutels meenemen. Nu zou de hoteleigenaar nog extremere maatregelen kunnen treffen door het inhuren van beveilingspersoneel dat de bezoekers fouilleert en door het installeren van metaaldetectoren bij de ingang, maar de kans dat hij dan überhaupt nog hotelgasten overhoudt, is dan erg klein. Een script is dus nooit dwingend, want anti-programma's blijven altijd mogelijk. Wel kun je stellen dat het veel minder makkelijk is om een anti-programma te ontwikkelen wanneer de heterogene coalitie van actanten steeds uitgebreider en omvangrijker is.

Een notie die samenhangt met dit netwerkdenken is het concept "obligitary point of passage" (Callon 1986). De vormgeving van een heterogeen netwerk, kan namelijk bepaalde routes voorschrijven die afgelegd dienen te worden binnen het netwerk. Callon beschrijft in zijn casus over wetenschappers die onderzoek doen naar de mogelijkheden voor de kunstmatige kweek van schelpen voor consumptie, hoe zij de positie weten te verwerven van een noodzakelijk passagepunt voor iedereen die iets over deze schelpen wil beweren. De wetenschappers sluiten andere actoren in het netwerk af van directe toegang naar andere plekken binnen het netwerk. Hierdoor creëeren zij voor zichzelf een onmisbare en controlerende functie. Wanneer de schelpen zelf op een gegeven moment niet meer meewerken, doordat ze niet kunstmatig kweekbaar meer zijn, verliezen de wetenschappers echter hun exclusieve toegangscontrole en raken ze hun macht weer kwijt.

Het belangrijkste inzicht van deze studie, is het feit dat actoren door het creëeren van een 'obligitary point of passage' controle kunnen uitoefenen over andere actoren in het netwerk. Door de netwerkgeografie te veranderen door bepaalde routes af te sluiten, en nieuwe door hen zelf gecontroleerde wegen te openen, kunnen zij de bewegingsvrijheid van andere actoren controleren. Zowel actoren als artefacten zijn dus altijd gelokaliseerd binnen een netwerk of meerdere netwerken. Mol (1999) werkt deze notie van de gelocaliseerdheid verder uit, wanneer zij het concept 'ontologische politiek' verder uitwerkt. Volgens Mol bestaan er door de gesitueerdheid van artefacten binnen netwerken, meerdere versies die zowel historisch, cultureel als materieel gelocaliseerd zijn. De vraag waar deze objecten gelokaliseerd zijn hangt af van de discipline waarin we op zoek gaan. Zo worden wetenschappelijke feiten geconstrueerd in een laboratorium en van daaruit naar de rest van de wereld geëxporteerd, niet zozeer in de vorm van theorie, maar vooral in de vorm van materiële artefacten (zie ook: Latour 1987).

Om de verschillende posities die actoren binnen een netwerk kunnen bekleden en hoe dit hun relatie tot technologie beïnvloed verder te kunnen verklaren, leen ik van Bijker de twee concepten 'technologisch frame' en 'inclusie in een technologisch frame' uit zijn Social Construction of Technology theorie, die hij samen met Trevor Pinch ontwikkelde. Bijker (1995a) definieert een technologisch frame als volgt (p. 123):

> "A technological frame compromises all elements that influence the interactions within relevant social groups and lead to the attribution of meanings to technical artifacts - and thus to constituting technology ... these elements include ... : goals, key problems, problem-solving strategies (heuristics), requirements to be met by problem solutions, current theories, tacit knowledge, testing procedures, and design methods and criteria."

Hiermee lijkt de notie van technologisch frame veel op het paradigmabegrip van de wetenschapshistoricus Kuhn. In 'The Structure of Scientific Revolutions' uit 1962 introduceert Kuhn dit begrip. Hoewel de notie paradigma meerdere betekenislagen in zich draagt, bedoelt Kuhn met paradigma (in de betekenis van 'disciplaire matrix') in de eerste plaats een leidend voorbeeld voor welke soort problemen wetenschappers moeten aanpakken en op welke manier ze deze problemen dienen op te lossen. Aan de hand van dit soort problemen leert een onderzoeker zijn vak. Hij krijgt standaardopgaven en moet zich de manier van kijken eigen maken die kenmerkend is voor de betreffende discipline. Via dit soort opgaven worden specifieke disciplinaire perspectieven overgedragen en verwerft een wetenschapper-in-opleiding de vaardigheden die voor een onderzoeker in een bepaald vakgebied noodzakelijk zijn (Vries de 1995: 102).

Daar waar bij Kuhn het paradigma alleen geldig was binnen de disciplaire context, dus alleen voor wetenschappers, heeft het concept technologisch frame een veel bredere

reikwijdte. Zo schrijft Bijker (1995a: 126): "... the concept of technological frame is meant to apply to all relevant social groups, not only engineers." Daarnaast bestaat het technologisch frame niet alleen uit symbolische generalisaties en verschillende sociale groepen, waaronder dus ook de gebruikers, maar ook uit de materiële artefacten zelf (ibid: 124): "The interactions in relevant social groups are never governed by cognitive and social factors alone. Artifacts, as stabilized in previous construction processes, play a crucial role too."

Het andere belangrijke concept dat ik van Bijker overneem is inclusie in het technologische frame. Verschillende actoren kunnen namelijk verschillende posities in het technologisch frame innemen. Hierdoor bezitten zij ook een verschillende relatie tot die technologie en de controle erover. Bijker (1995b) verklaart hiermee ook waarom technologie voor verschillende mensen een verschillende inclusie constitueert (p.21):

> "Voor een bepaalde persoon kan techniek zich op twee manieren als hard manifesteren—ik noem ze respectievelijk "weerbarstig" en "opsluitend." Deze twee soorten hardheid zijn afhankelijk van hoe centraal deze persoon in het betreffende technisch raam zit. Voor iemand die erg centraal zit—een hoge inclusie noem ik dat—heeft de betreffende techniek een grote hardheid in de vorm van opgeslotenheid, omdat er geen ontsnappen aan is: er zijn weliswaar veel mogelijkheden tot differentiatie binnen die techniek, maar het is ondenkbaar het zonder te doen. Voor iemand met een lage inclusie heeft techniek een heel ander soort hardheid, een weerbarstigheid: het is een slikken of stikken, geen mogelijkheid tot variatie binnen de techniek maar wel de mogelijkheid om haar helemaal te weigeren. Even een voorbeeld. Stelt u zich twee mensen voor achter een "personal computer." De eerste heeft een hoge inclusie in het technisch raam van de PC, de tweede een lage. Er gaat iets mis: de printer braakt abacadabra uit. Wat zullen zij doen? Mijn voorspelling is dat het meisje met de hoge inclusie, na een snelle controle van de verbindingssnoeren, met de printerdefinities gaat sleutelen. Zij komt niet op het idee haar brief met een balpen af te schrijven—ze is, met enige overdrijving gezegd, opgesloten in de techniek van de PC. De jongen met de lage inclusie daarentegen ziet geen uitweg, kan alleen nog nee zeggen tegen deze weerbarstige techniek, en pakt zijn vulpen."

Technologie heeft dus een verschillende hardheid voor actoren die een verschillende inclusie in het technologische frame bezitten (ibid.: pp. 19-20):

> "Naarmate een technisch raam van een relevante sociale groep zich meer stabiliseert, krijgen de daarbij horende artefacten, praktijken, rollen een vastere betekenis. Niet alles is meer mogelijk.
> [...]
> Die fixatie van betekenissen beperkt mensen in hun handelen, hij oefent macht uit. Deze structuur van gefixeerde betekenissen noem ik een semiotische machtsstructuur. 'Semiotisch' omdat het de betekenissen van machines, mensen, handelingen, beelden zijn die macht uitoefenen—doordat ze mogelijkheden scheppen en beperkingen opleggen.
> [...]
> Zo oefenen artefacten macht uit: als elementen in een semiotische machtsstructuur. Dit is ook de macht die wordt ervaren als 'technisch determinisme'; maar ze kan nu worden begrepen als een resultaat van menselijke interactie in plaats van als een aspekt van een interne logica van de techniek. Ze kan dan ook meer of minder bewust worden gecreëerd: 'Fate can be engineered. In that case, we should look at those who choose to invest in large, complex technologies, and consider that they may do so quite deliberately in order to create technological determinism'. Maar die macht heeft haar grenzen: niet iedereen is er even gevoelig voor. Alleen mensen wier handelen wordt geleid door het betreffende technisch raam zullen er de beperkingen van voelen."

Nu kunnen we ook begrijpen waarom verschillende actoren binnen een netwerk verschillende posities innemen, zich dus ook in een andere mate tot materiële artefacten binnen dat netwerk verhouden. Hiermee hangt samen dat verschillende actoren dus ook een verschillende

hoeveelheid invloed uit kunnen oefenen. Wanneer je je centraal in het netwerk bevindt, kun je veel invloed uitoefenen. Bevind je je meer naar de periferie, en ben je dus genoodzaakt via allerlei door andere(n) beheerde wegen jezelf toegang te verschaffen tot het centrum, is het veel moeilijker om invloed uit te oefenen, of zullen de gevolgen voor anderen van veel minder groot belang zijn, omdat er altijd alternatieve paden overblijven.

Vanuit dit kader van een gelokaliseerde netwerkgeografie waarbinnen gebruikers, producenten en artefacten zich bevinden, verklaar ik de invloed die alle partijen wel of juist niet op elkaar uit kunnen oefenen. Hierbij zal ik in de volgende hoofdstukken verklaren hoe de netwerkgeografie van gesloten programma's er uit ziet, en wat hierbij de verschillen zijn met die rondom open programma's. Om de verschillen tussen deze twee categorieën te kunnen begrijpen is het belangrijk om in te zien, dat zij beide een geheel andere omgeving rondom hen heen creëeren, als ware het twee verschillende universums. Alleen dan is het mogelijk om een beter inzicht te krijgen in de complexe interacties tussen alle verschillende groepen en artefacten. Law (1999: 7) vat dit samen wanneer hij schrijft: "... this sensibility for complexity is only possible to the extent that we can avoid naturalizing a single spatial form, a single topology."

3. Het gesloten software universum

§ 3.1 Inleiding

In dit hoofdstuk beschrijf ik aan de hand van theoretische noties uit het voorgaande hoofdstuk, de beperkingen van gesloten software ten aanzien van een democratisch gebruik ervan samenhangen met de vormgeving van de grenzen voor bewegingsvrijheid voor gebruikers in het netwerk waarbinnen computergebruik wordt vormgegeven. Ik laat zien hoe het netwerk van gebruikers en de leverancier van een gesloten computerprogramma er in het algemeen uitziet, waaraan dit netwerk zijn sterkte ontleent en tot welke implicaties het gesloten karakter van computerprogramma's ten aanzien van het gebruik leidt. De gebruiker van een gesloten computerprogramma bevindt zich in een netwerk waarover één partij, namelijk de softwarefabrikant, alle controle heeft verworven. De gebruiker rest slechts één keuze wat betreft het accepteren van zijn in de gesloten software ingebouwde rol: slikken of stikken.

§ 3.2 De Constructie van het gesloten computerplatform

1. De binaire zwarte doos

Computers zijn niet veel meer dan grote en dure rekenmachines. Een computer kan feitelijk gezien niet veel meer dan sommen maken. Het hart van een computer bestaat namelijk uit een zeer grote verzameling elektronische aan- en uitschakelaars. De enige twee concepten die een computer begrijpt, zijn dan ook aan (1) en uit (0).

Wat een computer zo bijzonder maakt, is dat hij gigantisch snel sommen met nullen en enen kan maken. Een computer kan namelijk miljoenen berekeningen per seconde uitvoeren en alle resultaten ook nog eens 'onthouden' in zijn 'geheugen'. Daardoor kan een apparaat dat slechts de meest simpele berekeningen kan uitvoeren, toch de meest complexe opdrachten uitvoeren. Uit zichzelf kan een computer niks zinnigs doen. Daarvoor heeft hij gedetailleerde opdrachten nodig, in een voor een computer begrijpelijke taal: enen en nullen dus.

Een computer opdrachten geven, noemen we programmeren. Een programma is dus eigenlijk niets meer dan een lange lijst met opdrachten die een computer kan uitvoeren. Deze lijst met computerinstructies noemen we de broncode (in het Engels source code) van een computerprogramma. In de begintijd van het computertijdperk was de broncode ook meteen het uiteindelijke computerprogramma zelf. Dat komt omdat de eerste computerprogramma's

direct in voor een computer begrijpelijke machinetaal geschreven werden. Computerprogrammeurs schreven hun programma's in opdrachten die uit enen en nullen bestonden, met als eindresultaat een eindeloze brij enen en nullen. Een fragment uit een programma in machinetaal ziet er zo uit:

```
0000011100111100 1010011101011110
1010110011110001 0101010110101011
```

Omdat de meeste mensen niet snel kunnen denken in enen en nullen, laat staan snel een type- of programmeerfout weten te vinden in een enorme cijferbrij, zocht men al snel naar een efficiëntere manier om computers te programmeren. De oplossing was de 'hogere programmeertaal'. Hierdoor kan een programmeur in relatief makkelijk te houden commando's die op 'normale' woorden lijken, zijn opdrachten aan de computer geven. Een computerprogramma vertaalt vervolgens deze voor mensen begrijpelijke opdrachten in voor computers begrijpelijke enen-en-nullen machinetaal. Het vertaalprogramma dat hiervoor nodig is noemt men een 'compiler'.

Een voorbeeld van een stukje broncode in de programmeertaal C zou er als volgt uit kunnen zien:

```
int i;
printf("%s (%u bytes):\n", caption, cnt);
for (i = 0; i < cnt; i++)
printf("%02X%c", ((BYTE *)ptr)[i], i % 16 == 15 ? '\n' : ' ');
putchar('\n')
```

Schematisch uitgedrukt ziet het maken van een computerprogramma in een hogere programmeertaal er als volgt uit:

1. Programmeur schrijft broncode (voor mensen leesbaar, maar niet voor een computer).
2. Compiler-programma vertaalt broncode naar binaire code ('computercode').
3. Computer voert programma uit.

Het gebruik van een 'hogere' programmeertaal heeft echter ook als gevolg dat de broncode (geschreven in een voor mensen begrijpelijke programmeertaal) niet meer gelijk is aan het uiteindelijke computerprogramma (dat bestaat uit voor een computer begrijpelijke enen en nullen). De broncode en het uiteindelijke programma zijn in hogere programmeertalen twee verschillende dingen. De broncode is alleen nodig om een programma te maken. Wanneer de compiler het binaire programma heeft gemaakt, dat uit enen en nullen bestaat, is de broncode niet meer nodig om het programma te kunnen gebruiken. Een softwaremaker kan vervolgens de binaire versie van zijn computerprogramma verkopen, en tegelijkertijd de broncode veilig in een kluis opbergen. Zo kunnen klanten wel zijn programma gebruiken, maar hebben concurrenten geen toegang tot de inhoudelijke

werking van zijn programma. Doordat de broncode ontbreekt ontstaat een dichtgelaste zwarte doos waartoe alleen de oorspronelijke auteur toegang heeft.

In het begin van het computertijdperk, ongeveer van 1943-1977, waren computers vooral heel erg duur, langzaam, log en niet talrijk. Het grote geld werd verdiend met de verkoop van de hardware en niet met de software. Applicaties werden speciaal in opdracht van de klant voor een specifieke computer geschreven. De broncode werd bij de software meegeleverd. Software, in mooie glimmende met plasticfolie dichtgesealde kartonnen dozen maar zonder broncode, ook wel 'shrink wrap' software genaamd, zoals we die tegenwoordig zelfs in supermarkten als de Albert Heijn kunnen kopen, bestond toen nog niet.

Een ander fundamenteel verschil met de huidige situatie, was dat je in het begin van het computertijdperk twee keuzes had als je een computerprogramma wilde gebruiken: of je schreef het zelf, of je huurde iemand in om voor jou een programma te schrijven, die je pas betaalde als de software aan de vooraf gestelde eisen voldeed. In beide gevallen was je ook de eigenaar van je eigen computerprogramma's. Wanneer je je programma's inclusief broncode met anderen wilde delen was dat gewoon mogelijk. Er bestond dan ook een levendige uitwisseling van programma's tussen computergebruikers.

Deze situatie veranderde volledig toen vanaf 1977 de opmars van de microcomputer begon, waardoor computers steeds goedkoper en talrijker werden. Op dat moment begonnen bedrijven te beseffen dat het veel interessanter was om geld te verdienen aan software dan aan hardware. Met de opkomst begin jaren tachtig van de eerste Apple computers en de 'Personal Computer' van IBM, verscheen tegelijkertijd de 'binary-only, shrink-wrap' software in de schappen van computerwinkels.

Bij deze nieuwe soort computerprogramma's zat ook een zogenaamde gebruikerslicentie, ook wel bekend als End User License Agreement (EULA). Hierin werd bepaald dat de koper geen eigenaar werd van de software, maar ermee akkoord ging dat hij er slechts onder strikte voorwaarden gebruik van mocht maken. Een gevolg van de softwarelicentie was ook dat het niet langer was toegestaan om programma's te delen, door middel van het maken en verspreiden van kopieën aan anderen. Dat recht had alleen de fabrikant.

In die tijd was het een logische keuze voor kleinschalige computergebruikers om een gedeelte van hun vrijheid af te staan aan de commerciële softwaremakers. Daarvoor kregen ze namelijk veel terug: relatief goedkope computerprogramma's die meteen bruikbaar waren. Het was ondenkbaar dat iedereen zijn eigen tekstverwerker of spreadsheet-programma zou moeten schrijven, laat staan een compleet besturingssysteem. Daarnaast hoefde je als gebruiker veel minder van computers af te weten om er mee te kunnen werken. Gesloten

software maakte het mogelijk om een computer te gebruiken zonder dat je bijvoorbeeld moest kunnen programmeren.

Het grote nadeel voor de klant is dat hij zich hiermee afhankelijk maakt van de softwarebedrijven waar hij zijn programmatuur koopt. Want een gesloten binair programma kun je niet veranderen, je kan het niet bekijken om te zien hoe het in elkaar zit, en je kunt het programma ook niet op een ander type computer gebruiken dan waarvoor het gecompileerd is. Een computerprogramma zonder broncode is een 'black box' die de gebruiker niet kan openen. De gebruiker van een programma zonder broncode is voor onderhoud, ontwikkeling en aanpassingen of aanvullingen aan het computerprogramma volledig afhankelijk van de maker van het programma.

Als een softwarebedrijf failliet gaat, of besluit om een bepaald computerprogramma niet meer te leveren is er geen mogelijkheid meer voor een gebruiker om aan nieuwe versies van dat programma te komen. Als er fouten in een programma zitten, waarvan de fabrikant het niet de moeite waard vindt om ze eruit te halen, kan de gebruiker daar niets aan doen. Veruit de meeste (maar niet alle) makers van commerciële software leveren tegenwoordig geen broncode bij hun programma's. En dat is hun goed recht, want niemand verplicht hen om dat te doen. Als je een auto koopt, krijg je er ook geen gedetailleerd boek bij over het fabricageproces, of als je een fles Cola koopt, krijg je het recept er ook niet bij.

Softwareproducerende bedrijven beschouwen de broncode van hun programma's als bedrijfsgeheim, om te voorkomen dat concurrenten het product na kunnen maken. Het verschil is echter dat software geen stoffelijke zaak is, zoals een auto dat wel is. Bij een auto ben je voor reparaties niet afhankelijk van één leverancier. Als je niet tevreden bent over de garage van je autodealer, ben je vrij om een andere te kiezen. Als je een andere motor in je auto wilt inbouwen, dan is dat technisch mogelijk. Autobanden van een andere fabrikant of je autoradio uit je vorige auto? Geen enkel probleem. Zelf klein onderhoud uitvoeren, is voor technisch onderlegde mensen ook geen probleem. Kortom, je kunt een auto openen om te kijken hoe hij werkt, je kunt er zelf dingen aan veranderen en je kunt zelf reparaties uitvoeren[18]. Bij software zonder broncode is dit allemaal niet mogelijk. Daardoor verkeert een gebruiker van 'software'-technologie in een fundamenteel andere positie dan een gebruiker van 'hardware'-technologie.

Een computerprogramma zonder broncode is als een auto met een dichtgelaste motorkap,

[18] Het is interessant om te zien dat ditzelfde proces van 'black boxing' steeds meer van toepassing is op de auto-industrie door het gebruik van moderne electronische systemen in auto's zoals complexe systemen voor motor- of remmanagement. Dit heeft tot gevolg dat een autogarage in steeds mindere mate service kan verlenen aan

vastgeklonken wielen en een dichtgesmolten autoradio. Gaat er iets kapot of moet er iets veranderd worden, dan ben je volledig afhankelijk van de leverancier waar je je product gekocht hebt. Een andere leverancier kan je niet verder helpen, en softwaregarages voor computerprogramma's zonder broncode zijn er ook niet. De gebruiker kan nergens anders naar toe, dan naar de originele maker van zijn computerprogramma. De softwaremakers hebben door middel van de strategie van 'black boxing' een zogenaamd 'obligitary point of passage' gecreëerd en daarmee de netwerkgeografie zodanig veranderd dat de macht exclusief bij de maker ligt in plaats van bij de gebruiker of koper.[19]

2. De configuratie van de 'domme' gebruiker

Door de toegang tot de broncode van een programma af te schermen, kunnen softwarefabrikanten ook controle uitoefenen op het gebruik ervan. In de gesloten software wereld wordt gebruikers namelijk niet alleen ontmoedigd om te kijken hoe een computerprogramma werkt; het wordt ze eenvoudigweg onmogelijk gemaakt. Het gevolg hiervan is dat een gebruiker van gesloten computerprogramma's niet veel anders kan doen dan het kritiekloos, fantasieloos indrukken van de knoppen die de softwaremakers hebben ingebouwd. Het is voor een computergebruiker niet mogelijk om aan gesloten programmatuur op fundamentele wijze iets te veranderen.

Een voorbeeld waartoe dit kan leiden is het feit dat in de succesvolle tekstverwerker Word van Microsoft tot op de dag van vandaag in haar hele bijna twintig jarige bestaan nog steeds geen handige functie is ingebouwd om snel en efficiënt wiskundige formules te schrijven.[20] Er bestaat wel een functie die 'formule-editor' heet, maar die functie is voor wiskundigen absoluut niet toereikend. Omdat wiskundigen echter maar een klein aandeel vormen van de grote groep Word-gebruikers vindt Microsoft het niet interessant om deze functie in te bouwen. Wiskundigen moeten zelf maar naar een oplossing zoeken.[21]

---

auto's van fabrikanten met wie zij geen strategisch verbond zijn aangegaan.

[19] Tegenwoordig biedt Microsoft onder de naam "Shared Source Program" zeer grote afnemers zoals overheden, de mogelijkheid om een (vluchtige) blik te werpen op de broncode van haar producten. Hiermee behoudt Microsoft echter nog steeds alle controle omdat alles onder strenge voorwaarden gebeurt, waarbij eerst een compleet contract getekend dient te worden. Individuele gebruikers, kleine bedrijven of concurrenten worden door Microsoft uitgesloten van inzage in de broncode.

[20] Daarom gebruiken wiskundigen, fysisici en andere exacte wetenschappers over het algemeen nog steeds het programma TeX, dat eind jaren zeventig al speciaal voor dit doel werd geschreven door computerwetenschapper Donald Knuth.

[21] Leslie Lamport, auteur van een belangrijk macropakket voor TeX stelt somber: "Now, technically sophisticated users are an insignificant niche market. ... So mathematicians have no place in the brave new world of computing. They will have to make do with the same flashy but technically impoverished tools that the lady in Peoria uses. So you can display video animations on the web, but there's still no good way to display a mathematical equation." In: (Ziegler, 2000).

Als we het voorgaande voorbeeld veralgemeniseren, dan kunnen we stellen dat in gesloten software vaak een script van een "zero-learning-curve" gebruiker is ingebakken, die alles meteen moet kunnen zonder ook maar iets te hoeven leren. De binnen zijn vakgebied bekende Nederlandse computerpionier Edsgar Dijkstra observeerde over deze kwestie al in de jaren tachtig het volgende:

> "The computer industry's way of speaking about its customers is very condescending: the user - sometimes strengthened as 'the end user'- is treated as a moron - e.g. 'making our systems so user-friendly that even housewives can use them'"-. (Dijkstra 1985: 3)

> "The computer 'user' isn't a real person of flesh and blood, with passions and brains, no, he is a mythical figure, and not a very pleasant one either. A kind of mongrel with money but without taste, an ugly caricature that is very uninspiring to work for. He is, as a matter of fact, such an uninspiring idiot that his stupidity alone is a sufficient explanation for the ugliness of most computer systems. And oh! Is he uneducated!" (Dijkstra 1982: 298-90)

Dit heeft als direct gevolg dat de vrijheid van computergebruikers die hun computer geavanceerde opdrachten willen geven in hun handelen beperkt worden door de grenzen die de maker in het programma heeft ingebouwd. De gebruiker kan in dit geval de werking van zijn programma op geen enkele wijze direct aanpassen, omdat hij niet beschikt over de broncode. Voor veranderingen zal hij moeten aankloppen bij de fabrikant, die misschien tegen betaling het programma wil veranderen. Of niet, wanneer hij tot een commercieel niet-interessante doelgroep behoort. Daardoor ontstaat een technische dictatuur van de meerderheid die haar wensen ten aanzien van het gebruik van computerprogramma's oplegt aan de rest die specifieke, niet veel voorkomende wensen heeft. Mogelijkheden om dan maar zelf functies in te bouwen in een programma zijn er ook niet door het gesloten karakter. De interface tussen de gebruiker en het computerprogramma is onveranderbaar uitgehard, en daaraan kan door de gebruiker niets veranderd worden.

3. Gesloten bestandsformaten en communicatieprotocollen

Nadat de gebruiker opgesloten zat binnen de grenzen van het gesloten programma zonder broncode en binnen het script de gebruiker in de rol van de 'eenvoudige' gebruiker, probeerden de makers van gesloten software hun klanten nog verder aan zich te binden. De grenzen van het gesloten software universum werden nog verder dichtgetimmerd door de invoering van gesloten bestandsformaten en communicatieprotocollen. Hierdoor worden gebruikers van gesloten programmatuur voor de toegang tot de door hen zelf geproduceerde data afhankelijk van de fabrikant. De data die zijn opgeslagen in een geheim bestandsformaat door een gesloten programma, zijn daardoor niet toegankelijk voor programmatuur van concurrenten.

Opnieuw gebruik ik Microsoft Word als voorbeeld van deze praktijk. Wanneer je een

tekst schrijft in Word en deze vervolgens opslaat onder de naam tekst.doc en je probeert dat bestand met een zogenaamde ASCII-editor (een standaardprogramma om tekst mee te kunnen lezen en schrijven op een computer, in Windows heet dit programma 'Kladblok') te openen zie je alleen maar obscure tekentjes op je scherm: ÐÏ#à¡±. Dat betekent dat zogenaamde ".doc" bestanden alleen door Microsoft Word op een zinvolle wijze in te lezen zijn, en niet door andere computerprogramma's van concurrenten. Hiermee is een Microsoft Word-bestand al net zozeer een dichtgesmolten blok plastic, waar diep van binnen ergens de tekst van de gebruiker ligt opgeslagen, zonder dat die gebruiker daar zelf ooit nog bij kan, zonder het gesloten Microsoft Word-programma als intermediair.

Niet alleen oude gebruikers worden zo gevangen gehouden binnen de grenzen van de gesloten software wereld, maar door de veranderde netwerkgeografie worden ook nieuwe gebruikers naar binnen gezogen. Wanneer een nietsvermoedende gebruiker (of misschien wel een ambtenaar van een overheidsinstantie) een Word-bestand als attachment naar andere mensen e-mailt, dan kunnen zij dit bestand alleen lezen als zij over het Microsoft Word programma beschikken. Wanneer iedereen in je omgeving gebruikt maakt van gesloten programma's om informatie in gesloten bestandsformaten op te slaan, en jij afhankelijk bent voor je informatie van uitwisseling met dit soort gebruikers, dan rest jou niet veel anders dan, of volledig te stoppen met het uitwisselen van informatie met Word-gebruikers, of zelf ook maar die Windows-computer aan te schaffen met Word erop. Let wel, we hebben het hier over een commercieel programma dat je voor veel geld in de winkel dient te kopen.

Microsoft biedt dan misschien wel een zogenaamde Word Viewer gratis aan op haar website waarmee je Word-bestanden kunt lezen zonder dat je er iets aan kunt veranderen, maar dit programma werkt alleen onder het besturingssysteem Microsoft Windows (nieuwste versie hiervan is te koop voor iets minder dan 400 euro). Bezit je een computer met daarop een ander besturingssysteem, zoals Linux, Unix (Solaris, AIX, HP-UX, SCO Unix, IRIX, Apple OS X, FreeBSD, OpenBSD, NetBSD, BSD/OS, GNU HURD, QNX), een van de verschillende versies van DOS (MS-DOS, DR-DOS, FreeDOS), Mac/OS, Amiga/OS, Atari/Os, BeOS, VMS of OS/2, dan heb je een probleem en kun je het bestand gewoon niet lezen, want voor deze besturingssystemen stelt Microsoft noch viewers, noch het Word-pakket zelf beschikbaar (uitgezonderd voor gebruikers van Apple computers, die kunnen voor 712,81 euro een office pakket kopen, waar Word inzit). Kortom, wanneer je Word-bestanden wil kunnen lezen, wordt je gedwongen om een Intel-computer met Microsoft Windows met Microsoft Word daarbij aan te schaffen.

Zo zie je dat een simpel technisch detail zoals het bestandsformaat waarin een tekst

wordt opgeslagen, toch gevolgen kan hebben voor het computergebruik en gebruikers een bepaalde richting op kan dwingen. Gebruikers die dat niet willen of kunnen, worden simpelweg uitgesloten. Technisch gezien kan er net zo makkelijk gekozen worden voor een manier van opslaan van tekstbestanden die juist zoveel mogelijk gebruikers op verschillende computerplatformen insluit. Een voorbeeld van een bestandsformaat dat in voor iedere computer leesbare ASCII-code is opgeslagen is het HTML-formaat voor internetpagina's, of een ander voorbeeld is het PDF-formaat, waarvoor de specificaties voor iedereen op te vragen zijn, en waarvoor voor veel meer verschillende computerplatformen 'viewers' bestaan.

Eenzelfde tactiek wordt toegepast met protocollen voor het uitwisselen van informatie. Zo neemt Microsoft standaarden die publiekelijk beschikbaar zijn, en verandert daar vervolgens van alles aan, zodat alleen gebruikers van Microsoft producten er nog iets aan hebben. Een voorbeeld hiervan is de HTML-specificatie voor de taal waarin webpagina's worden geschreven. Microsoft heeft hierin allerlei veranderingen doorgevoerd, waardoor pagina's die er in Microsoft Internet Explorer mooi uitzien en allerlei handige functionaliteit bezitten, in andere webbrowsers niet werken en niet leesbaar zijn.[22] Deze tactiek wordt ook wel 'embrace, extend and enhance' genoemd. Hierdoor worden computergebruikers opnieuw een ruimte binnengeloodst, waaruit ontsnappen niet of slechts zeer moeilijk mogelijk is.

4. 'Totale Integratie'

De laatste stap in het afgrenzen van de gesloten software wereld van de buitenwereld vol concurrentie en vrije keuze, is de 'totale integratie van de complete computerervaring'. Hierbij worden alle programma's met elkaar geïntegreerd (een mooi voorbeeld zijn de zogenaamde 'Office'-pakketten). Vervolgens worden deze programma's onlosmakelijk verbonden met en ingebouwd in het besturingssysteem zelf. Voor alle taken die de gebruiker met zijn computer wil uitvoeren is hij nu afhankelijk van één centrale softwarefabrikant. Andere fabrikanten kunnen nog wel applicaties leveren, maar die zijn nooit zo goed aangepast op het besturingssysteem, als programma's die door dezelfde fabrikant worden gemaakt die ook het besturingssysteem ontwikkeld.

Omgekeerd maken sommige fabrikanten hun software-applicaties alleen geschikt voor hun eigen besturingssysteem, wat in feite neerkomt op een soort gedwongen koppelverkoop. Grote voorbeelden zijn Apple en Microsoft. Wanneer je Apple-software wilt kunnen

[22] Een voorbeeld hiervan is de NS-reisplanner op internet (http://www.ns.nl) die alleen werkt in Internet Explorer. Blinde computergebruikers die de webbrowser Lynx gebruiken hebben daardoor geen toegang meer tot de NS-website.

gebruiken, dien je daarvoor zelfs je computer bij Apple te kopen. Ook gebruikers die Microsoft-applicaties willen kunnen gebruiken, dienen daarvoor eerst een exemplaar van Microsoft Windows aan te schaffen. Microsoft Word of Apple iTunes gebruiken binnen het alternatieve besturingssysteem Linux is hierdoor bijvoorbeeld niet mogelijk. Daarnaast zorgen fabrikanten als Microsoft en Apple ervoor dat de nieuwste applicaties alleen werken op de laatste versies van hun besturingssysteem, om gebruikers zo te dwingen om hun software te blijven upgraden.

§ 3.3 De gevolgen van de gebruikergevangenis

Wanneer we het totaal aan maatregelen overzien dat fabrikanten treffen om hun gebruikers aan zich te binden, ontstaat het beeld van een soort gevangenis voor gebruikers. Binnen de muren van de door de fabrikant gestelde grenzen, bezitten de gebruikers de illusie van vrijheid, maar zodra ze een wens hebben die niet is ingebouwd dan kan die niet vervuld worden. De strategie van een gesloten computertechnologie mag dan voor de softwarefabrikanten veel voordelen met zich meebrengen, voor de gebruiker veroorzaakt deze praktijk veel schadelijke gevolgen.

1. Vendor lock-in

De gebruiker van moderne computerprogramma's is inmiddels volledig afhankelijk gemaakt van de makers ervan, zoals Microsoft, Apple, Oracle en andere grote bedrijven. Daardoor betaalt hij of zij meestal te veel voor zijn computerprogramma's. Toen dit jaar voor het eerst het winstpercentage van de afdeling besturingssystemen van Microsoft bekend werd gemaakt, bleek dit cijfer zo'n tachtig procent te bedragen, een ongekend hoog winstcijfer. Daarnaast worden nog steeds nieuwe modellen ontwikkeld om de klant meer te kunnen laten betalen. De laatste strategie hierbij is het abonnementmodel, waarbij software alleen nog maar voor een jaar 'gehuurd' kan worden. In plaats van één keer een vast bedrag voor de aanschaf van software te betalen, zal een gebruiker nu ieder jaar opnieuw moeten bijbetalen. Ook werkt Microsoft aan nieuwe technologie onder de codenaam Palladium, die het onmogelijk maakt om software te installeren, die niet door Microsoft gecertificeerd is (Himelein 2002). Dat betekent dat Microsoft zelfs voor concurrerende programmatuur van andere softwaremakers een verplicht passagepunt is geworden. Dit is een zorgwekkende ontwikkeling, omdat een gebruiker in dit nieuwe model geen enkele controle meer heeft over het gebruik van zijn eigen computer, inclusief de door hem zelf gegenereerde en opgeslagen data.

2. Technologische monocultuur

Behalve dat in de gesloten software wereld alleen 'zwarte dozen' worden verkocht, komen computerprogramma's voor het grootste deel ook nog eens allemaal uit dezelfde fabriek. Vooral het programma dat de infrastructuur aanlegt voor andere computerprogramma's, namelijk het besturingssysteem van een computer. Bijna alle besturingssystemen voor personal computers komen uit dezelfde computerprogrammafabriek in de Verenigde Staten genaamd Microsoft. Of een rechter nu wel of niet vindt dat Microsoft op dit gebied een monopoliepositie bezit en die misbruikt; feit is dat op negentig procent van alle computers wereldwijd, een variant van het besturingssysteem Microsoft Windows is geïnstalleerd.[23]

In de ecologie bestaat het begrip monocultuur voor gebieden die overheerst worden door één soort. Een voorbeeld van honderden hectare landbouwgrond, waar maar één ras graan staat is hier een concreet voorbeeld van. Dat monoculturen veel nadelen hebben, zoals bijvoorbeeld een grote gevoeligheid voor plagen is ook langer bekend. Het is interessant dat deze gevoeligheid ook op het gebied van computerprogramma's bestaat. Ook daar is sprake van een monocultuur van Microsoft-programma's die allemaal gevoelig zijn voor dezelfde 'plagen', met als desastreus gevolg wereldwijde uitbraken van relatief simpele emailvirussen zoals het Melissa-virus of de I-love-you-bug die een enorme economische schade wisten aan te richten.

Een van de allergrootste problemen vormen de steeds weer opnieuw opduikende veiligheidslekken in het meest gebruikte computerplatform Microsoft Windows. Doordat niemand behalve Microsoft haar software pro-actief kan controleren op veiligheidslekken, blijven er keer op keer nieuwe 'virussen', 'wormen', 'buffer overflow exploits' en andere beveiligingsproblemen opduiken. Wanneer je creditcard-gegevens gestolen worden omdat een webwinkel je data op een Microsoft-systeem had opgeslagen is dat erg vervelend, maar hoe zit het bijvoorbeeld met gegevens die de overheid over haar burgers opslaat? Kunnen we als maatschappij erop vertrouwen dat een commercieel bedrijf deze data goed genoeg beveiligt?

De oplossing is simpel en al lang bekend: zorgen voor voldoende variëteit (Kleiner &

[23] Het is erg ironisch dat Microsoft haar monopolie heeft weten te bouwen op een open hardware standaard, namelijk die van de Personal Computer. Zie: Ceruzzi 1998: 272-280: "Neither IBM nor ayone else foresaw how successful it ['s Personal Computer] would be, or that others would copy its architecture to make it the standard for the next decade and beyond. ... MS-DOS transformed Microsoft from a company that mainly sold BASIC to one that dominated the small systems industry in operating systems. IBM found itself with an enormously successful product made up of parts designed by others, using ASCII instead of EBDIC, and with an operating system it did not have complete rights to. ... Within ten years there were over fifty millions computers installed that were variants of the original PC architecture and ran advanced versions of MS-DOS."

Graham-Rowe 2000). Landbouwgebieden met een grote soortendiversiteit zijn veel minder gevoelig voor schimmels, ongedierte en virussen. Dit principe van broodnodige diversiteit kan desastreuze gevolgen door toedoen van digitale virussen aan onze moderne informatiemaatschappij voorkomen. Hoewel we voor de bestrijding van een longontsteking niet afhankelijk willen zijn van slechts één soort antibioticum, voor onze voedselvoorziening van één voedselproducent, voor onze informatievoorziening van één nieuwsproducent of voor onze energievoorziening van slechts één energieproducent, of voor het bereiken van een bepaalde stad van één toegangweg, zo zijn we op dit moment wereldwijd voor de voorziening van de basisinfrastructuur (namelijk het besturingssysteem) van onze personal computers bijna volledig overgeleverd aan de wil en onwil van een niet-democratisch opererend bedrijf: Microsoft.

3. Problematische privacy

Omdat niemand in de broncode van gesloten programma's kan kijken, weet ook niemand wat deze programma's buiten het medeweten van de gebruiker om allemaal doen. Zo bestaat er de zogenaamde categorie software die 'spyware' heet en zonder medeweten en toestemming van de gebruiker, data over zijn computergebruik via internet naar centrale servers toestuurt.[24] Vervolgens kan dan bijvoorbeeld aan de hand van het surfgedrag op internet op maat gemaakte reclame per e-mail naar de gebruiker worden gestuurd. Ook Microsoft-producten sturen informatie over de gebruiker over internet. Zo staat bijvoorbeeld het volgende in de gebruikerslicentie van Microsoft-producten (Braeken 2002):

> *U stemt ermee in dat MS en Microsoft Corporation en aan hun gelieerde bedrijven technische informatie die op welke manier dan ook wordt verzameld als onderdeel van de (eventueel) aan u geleverde productondersteuning met betrekking tot de SOFTWARE mogen verzamelen en gebruiken.*
>
> *...*
>
> *U erkent en stemt ermee in dat MS, Microsoft Corporation of hun dochterondernemingen automatisch de versie van de SOFTWARE en/of de onderdelen die u gebruikt mogen controleren en upgrades of fixes voor de SOFTWARE mogen leveren die automatisch naar uw COMPUTER worden gedownload.*

Daarnaast kan niemand controleren of de fabrikant niet zogenaamde 'back-doors' in zijn programma's heeft ingebouwd. Ondertussen zijn er meerdere malen zulke geheime toegangspoorten in gesloten software ontdekt, onder andere in Lotus Notes en in Microsoft Windows. Zeker wanneer software door bedrijven, overheden of bijvoorbeeld mensenrechtenactivisten wordt gebruikt, is het absoluut noodzakelijk dat geheime informatie

[24] Er bestaat zelfs een hele nieuwsgroep op internet die aan dit thema gewijd is: alt.privacy.spyware/

niet kan uitlekken. Gesloten software kan deze garantie nooit overtuigend bieden.[25]

4. Bedreiging voor minderheden, kleine culturen en kritische gebruikers

Een monocultuur - de letterlijke vertaling, één cultuur - is niet goed voor het behouden van culturele diversiteit. Nu kun je je afvragen wat computerprogramma's met het behoud van cultureel erfgoed of cultureel imperialisme te maken hebben, maar hopelijk maakt de volgende casus over het gebruik van Windows in IJsland dat duidelijk (Vermeer 1999).

Omdat IJsland erg trots is op haar eigen taal en cultuur, wil het graag gebruik maken van een versie van Microsoft Windows die vertaald is in het IJslands. Daarom diende de IJslandse regering een verzoek voor een vertaalde versie Windows in bij Microsoft omdat er nog geen IJslandse versie van Windows op de markt voor handen was. Microsoft antwoordde, dat het commercieel gezien niet interessant was om een IJslandse versie uit te brengen, omdat de IJslandse bevolking te klein is. Op een aanbod van de IJslandse regering om de kosten voor het vertalen te bekostigen ging Microsoft niet in. Nu kun je je afvragen, waarom de IJslandse overheid, het vertalen vervolgens dan niet gewoon zelf regelde. Het antwoord is, dat het dat gewoon niet kon, omdat Microsoft nooit haar broncode aan derden afstaat, zodat behalve Microsoft niemand er iets aan kan veranderen. De regering van een rijk, Westers, land kan dan hoog en laag springen, maar er niets aan veranderen.

5. Bit-rot

Het opslaan van gegevens in geheime en gesloten bestandsformaten heeft nadelige gevolgen voor de archivering van data en informatie. Want hoewel voor tekstverwerking nu het Word-bestandsformaat het meest gebruikt wordt, gebruikten de meeste mensen nog geen tien jaar geleden Wordperfect. Tegenwoordig gebruikt bijna niemand meer Wordperfect. Zijn echter alle documenten die in dit formaat zijn opgeslagen eigenlijk nog wel toegankelijk voor huidige computergebruikers, en ook nog voor toekomstige generaties computergebruikers? Wanneer gebruikers tegenwoordig hun oude bestanden van concurrerende programma's proberen te openen, krijgen zij niet meer dan een verzameling nietszeggende tekens op hun beeldscherm te zien. Wanneer een gebruiker zijn data opslaat in een geheim bestandsformaat, waarvan alleen de fabrikant de specificaties kent, en de fabrikant vervolgens failliet gaat, of

[25] Zie Jansen & Janssen 1999, p. 146: "Security through obscurity is no security at all." En Anderson 2001, p. 4: "It's probably not such a great idea for all of us to run, all the time, software whose mysteries and true behavior are known only to the companies that wrote it. Jörg Wunsch, a FreeBSD developer who grew up in what was then East Germany signs his email messages with the line 'Never trust an operating system you don't have sources for.' There's some wisdom in that."

besluit geen versies van de benodigde programmatuur meer beschikbaar te stellen, heeft de gebruiker een groot probleem. Zeker wanneer deze gebruiker een overheidsonderdeel is, kunnen we ons afvragen of dit wel een wenselijke gang van zaken is.

§ 3.4 Het Microsoft-mindshare-monopolie

De meeste mensen weten tegenwoordig niet beter dan dat een computer een grijze metalen doos is met een stickertje "Made for Microsoft Windows" erop, en dat het merendeel van de voorgeïnstalleerde software ook de naam van Microsoft als maker draagt: Microsoft Windows als besturingssysteem, Microsoft Word voor het maken van tekstdocumenten, Microsoft Powerpoint voor het geven van presentaties, Microsoft Excel voor het maken van 'spreadsheets', Microsoft Access voor het aanleggen van een elektronische kaartenbak, Microsoft Internet Explorer om te surfen op het internet, de Microsoft Encarta encyclopedie om informatie in op te zoeken, Microsoft Windows Media Player voor het bekijken van filmpjes en het beluisteren van MP3's, Microsoft Hotmail als gratis e-maildienst en Microsoft MSN om mee te chatten. Het wordt gebruikers wel heel erg makkelijk gemaakt om de programma's te gebruiken die al op de computer aanwezig zijn als ze hem kopen, om zo de concurrentie uit de markt te duwen.

De Microsoft-monocultuur strekt zich echter verder uit, zelfs tot aan het denken van mensen over het gebruik van computers. Je zou kunnen zeggen dat Microsoft de netwerkgeografie rondom computergebruik zodanig in eigen voordeel heeft weten te veranderen, dat zij er in geslaagd zijn om zelf centraal te staan in het technologische frame voor hedendaags personal computer gebruik. Als je iets wil doen met een computer, dan gebruik je in de eerste plaats programma's die geschikt zijn voor Microsoft Windows. In de tweede plaats zijn dit programma's waarvoor je geld moet betalen, waarvan je niet kan kijken hoe ze werken, waaraan je dus ook niks kan veranderen, en die je niet mag kopiëren om ze aan je vrienden te geven. Dit dominante gedachtegoed is inmiddels zo sterk geworden, dat er niet meer expliciet nagedacht wordt over de verbinding personal computer - Microsoft programma's.

In dit standaardbeeld over computers, is het normaal dat er in elke computer een Intel-processor zit, dat programma's altijd onder Microsoft Windows werken, dat computers al na twee jaar verouderd zijn, dat computers langzamer worden als je er meer programma's op installeert, dat je computer minimaal een keer per dag vastloopt, dat je je bestanden door virussen kunt kwijtraken, dat je voor elk wissewasje je computer opnieuw moet opstarten, dat de beste oplossing voor problemen met onstabiele computerprogramma's het opnieuw

installeren van Windows is, en dat er voor computerprogramma's veel geld betaald moet worden (of dat programma's illegaal geïnstalleerd worden). Dat al deze zaken eerder een gevolg van het Microsoft besturingssysteem van de gemiddelde pc zijn, dan van de mogelijkheden die de moderne hardware van een computer biedt, zullen veel mensen niet weten.

Hoe zou dat ook kunnen als de meeste mensen nooit iets anders gezien hebben? Immers op hun werk, op school, in de bibliotheek, bij de belastingdienst, in het ziekenhuis, overal staan dezelfde computers, met dezelfde programma's uit dezelfde fabriek. Microsoft bezit het overgrote deel van de markt voor besturingssystemen voor personal computers in handen heeft, en zorgt ervoor dat het merendeel van informatie in geheime bestandsformaten wordt opgeslagen.

Toch bezit Microsoft feitelijk geen monopoliepositie in de traditionele betekenis van het woord. Naast Windows kan een gebruiker van een Intel-compatible computer een heleboel andere, vaak zelfs technologische superieure en veel goedkopere, besturingssystemen gebruiken, zoals BeOS, Linux of FreeBSD en hij is ook vrij om een Apple Macintosh computer te kopen. Ook al gebruiken de meeste mensen Windows op hun computer, toch heeft Microsoft dus geen absolute monopoliepositie, want gebruikers kunnen er altijd voor kiezen om een alternatief besturingssysteem te kiezen. Het monopolie van Microsoft zit vooral in het hoofd van computergebruikers. Neal Stephenson verwoordt dit mooi in zijn boek 'In the beginning ... was the commandline':

> "Here, instead, the dominance is inside the minds of people who buy software. Microsoft has power because people believe it does. This power is very real. It makes lots of money. ... But this is not the sort of power that fits any normal definition of the word 'monopoly', and it's not amenable to a legal fix. The courts may order Microsoft to do things differently. They might even split the company up. But they can't really do anything about a mindshare monopoly, short of taking every man, woman, and child in the developed world and subjecting them to a lengthy brainwashing procedure. Mindshare dominance is, in other words, a really odd sort of beast ... it looks like one of these modern, wacky chaos-theory phenomena, a complexity thing, in which a lot of independent but connected entities (the world's computer users) making decisions on their own, according to a few simple rules of thumb, generate a large phenomenon (total dominance of the market by one company) that cannot by made sense of through any kind of rational analysis." (Stephenson 1999: 143-5).

Bill Gates, in zijn functie van directeur en oprichter van Microsoft, als een soort geniaal marketinggenie beschouwen die in zijn eentje de wereld heeft veroverd, is echter te veel eer aan één persoon toedichten. Het is alleen uit de door Microsoft op listige wijze verandere netwerkgeografie van het afsluiten van uitvalswegen naar concurrenten en het insluiten van gebruikers te verklaren, dat kopers van computerprogramma's logischerwijze op individuele wijze toch in grote getalen tot dezelfde conclusie kwamen: namelijk dat het kopen en gebruiken van Microsoft-producten in een wereld waarin iedereen dit op grote schaal lijkt te

doen, uiteindelijk gewoon het makkelijkste en goedkoopste is. Hierbij zijn de grote verzameling afzonderlijke softwareconsumenten te beschouwen als elementen van een complex systeem, die op basis van simpele regels (lage prijs, makkelijk gebruik, marktstandaard) uiteindelijk wanneer alle variabelen een bepaalde drempelwaarde overschreden hebben, tot een gezamenlijk collectief besluit lijken te komen.

Zo beschrijft Johnsen (2001) dit proces van emergentie, aan de hand van een soort slijmcellen. Deze cellen bewegen zich normaal autonoom en individueel door de natuurlijke omgeving, maar onder bepaalde omstandigheden 'besluiten' ze om bij elkaar te vloeien, waardoor een macroscopisch geheel ontstaat dat een nieuw organisme lijkt te vormen met haar eigen bewustzijn en gedrag dat verschilt van dat van de losse slijmcellen. Een dergelijke verklaringswijze kan ook worden toegepast op de manier waarop Microsoft een dominante marktpositie heeft weten te veroveren. Door slim in te spelen op de significante variabelen wisten ze steeds meer gebruikers over de streep te trekken, waardoor een zichzelf versterkend emergent proces in werking trad. Het positieve van deze verklaring is dat wanneer belangrijke variabelen veranderen (te duur, te weinig gebruiksgemak, niet conform standaarden), een andere netwerkgeografie mogelijk is, waardoor Microsoft haar dominante marktpositie weer kwijt kan raken.

4. Het open software universum

§ 4.1 Open software als anti-programma

In het vorige hoofdstuk heb ik besproken hoe het netwerk van gebruikers, fabrikanten en programmatuur in het gesloten software universum is opgebouwd en welke beperkende gevolgen dat voor het gebruik met zich meebrengt. In dit hoofdstuk bespreek ik de anti-these van gesloten software, namelijk open software. Dit type software is gelocaliseerd in een geheel andere netwerkgeografie waarbij gebruikers en makers andere posities innemen, over andere en vooral meer alternatieve verplaatsingsmogelijkheden beschikken en ook de technologie zelf, in de vorm van computerprogramma's, op een andere socio-technische manier vormgegeven is. Op macro-niveau zou je de ontwikkeling van open software zelfs kunnen beschouwen als een anti-programma van een gehele gemeenschap tegen de beknellende controle door gesloten software.

Maar wat is open software eigenlijk? In de schaarse literatuur bestaat er geen eenduidige definitie voor de term 'open software'. Er bestaat wel een veelvoud aan termen voor software die de gebruiker veel vrijheden oplevert, zoals: public domain software, free software, freeware, shareware, tryware, open source software, free source software en open software. Om niet te verzanden in muggezifterij over naamgevingskwesties, zoals het geval is bij het grote schisma tussen de aanhangers van de termen 'free software' en 'open source software', kies ik voor het gebruik van de term open software. Dat doe ik, omdat de nadruk niet ligt op het gratis zijn van de software, noch op een onvoorwaardelijke aanwezigheid van de broncode, maar op het democratische gehalte van computerprogramma's. Hierbij geldt dat naarmate het netwerk rondom een bepaald computerprogramma meer open is, zij participatie in de ontwikkeling en het gebruik ervan meer faciliteert.

Ik definieer computerprogramma's als open, wanneer zij de volgende kenmerken bezitten (in volgorde van toenemende openheid):[26]

1. *Open gebruik*: de gebruiker bezit de vrijheid om het gebruik van een programma naar eigen inzichten te configureren.

Veel gesloten commerciële programma's bieden binnen bepaalde grenzen weliswaar de

[26] Voor dit model heb ik gebruik gemaakt van de volgende literatuur: Abbate 1995 en 1999, Feller & Fitzgerald 2002, Ghosh et al 2002, Moody 2001, Raymond 1999, Young & Goldman Rohm 2000, Wayner 2000, Van Wendel de Joode, de Bruyn & van Eeten 2003, Working Group on Libre Software 2000.

mogelijkheid voor aanpassingen door de gebruiker. Maar voor radicale aanpassingen is de gebruiker nog altijd afhankelijk van de producent. Er is dan dus geen sprake van echte vrijheid voor de gebruiker.

2. *Open standaarden*: de opslag van data en de uitwisseling daarvan met andere programma's vindt plaats volgens open standaarden. Daardoor heeft de gebruiker de vrijheid om te kiezen uit alternatieve programma's die dezelfde functie vervullen, waardoor hij nooit afhankelijk kan raken van één leverancier.

Dat een programma open standaarden gedeeltelijk kan gebruiken, wil nog niet zeggen dat het hele programma ook volledig is ingericht op het gebruik ervan. Microsoft Word bijvoorbeeld kan wel heel goed informatie uit open bestandsformaten binnenhalen, maar niet goed exporteren naar andere open bestandsformaten (zoals bij de moderne XML-standaard om data uit te wisselen bijvoorbeeld). Gebruik van deze open bestandsformaten wordt in dit geval zelfs actief ontmoedigd, omdat de gebruiker dan informatie verliest bij het opslaan, zoals de opmaak van tabellen, voetnoten of illustraties. Er is dus geen sprake van een volledige, 'welwillende' ondersteuning.

3. *Open broncode*: de gebruiker kan inzicht verwerven in de werking van het programma en kan nauwkeurig controleren wat het programma wel en niet doet en hoe het dat doet.

Wanneer er inzicht is in de broncode hoeft dat nog geen toegenomen vrijheid voor de gebruiker te betekenen. Opnieuw Microsoft een voorbeeld met haar 'Shared Source Programme', dat valt samen te vatten als 'wel kijken, maar niet aanraken'. Bovendien bepaalt Microsoft de condities waaronder inzicht in de broncode mogelijk is. Microsoft blijft een noodzakelijk passagepunt wanneer je bij de broncode wil. Het encryptieprogramma PGP van Phil Zimmerman is een ander bekend voorbeeld.[27] De broncode van dit programma is voor iedereen vrijelijk te downloaden, maar je mag er verder niets mee doen, zoals veranderen of kopiëren.

4. *Open licentie*: de gebruiker bezit de vrijheid om veranderingen aan te brengen in de broncode, deze veranderde broncode te distribueren, te compileren voor andere computerplatformen, en gratis of tegen een financiële vergoeding te delen met anderen.

De openheid van de broncode en de licentie waaronder een programma is uitgebracht

[27] Zie: http://www.pgp.com/products/sourcecode.html

hangen nauw met elkaar samen, want je hebt pas echt iets aan open broncode, wanneer je daar ook vrij dingen aan mag veranderen. De ‘free software’ van het GNU-project is in dit opzicht bijvoorbeeld ook niet helemaal vrij, want je kunt veranderingen aan de broncode niet voor jezelf houden, zodat je er als bedrijf geld mee kunt verdienen. Bij ‘public domain’ programma’s mag je per definitie alles doen met het programma wat je wil, maar daar zit dan vaak weer geen broncode bij, zodat dat effectief onmogelijk is.

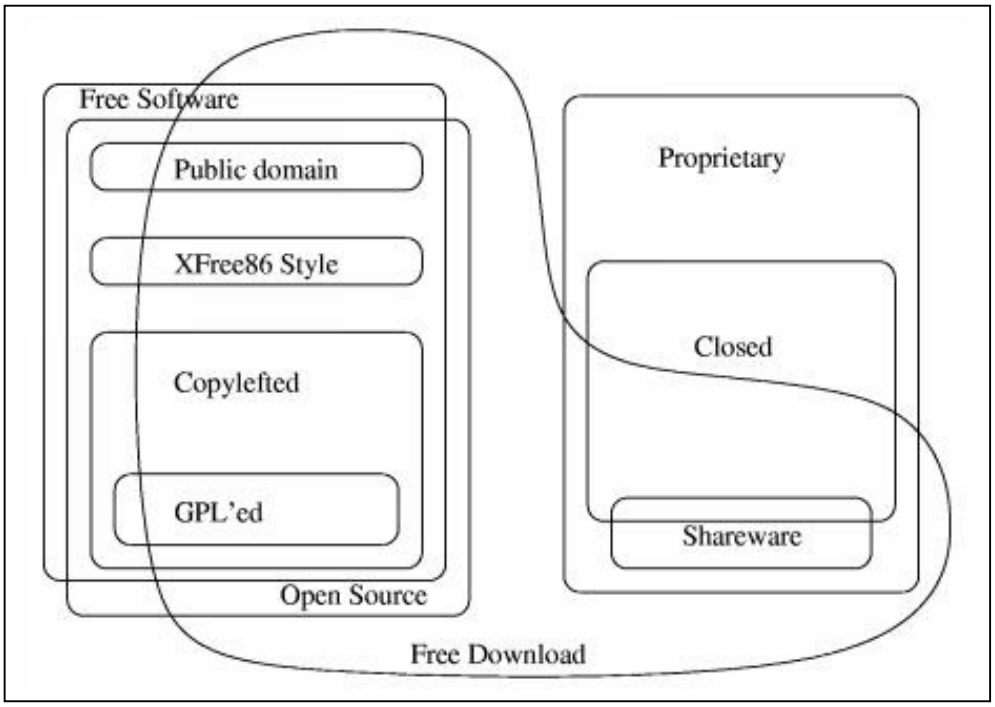
5. *Open ontwikkelingsmodel*: de gebruiker heeft toegang tot het ontwikkelingsproces van het programma en heeft daardoor invloed op toekomstige versies van de door hem gebruikte programmatuur.

Zelfs een programma dat aan de eerste vier voorwaarden voldoet, hoeft nog steeds geen open ontwikkelingsmodel te hanteren. Vaak blijft de originele programmeur een ‘obligitary point of passage’ voor de officiële distributie van dit programma. Wanneer het programma echter onder een open licentie is uitgebracht bezit je vaak wel de vrijheid om je eigen versie van het programma op de markt te brengen. Voor je eigen versie mag je dan ook zelf de openheid van je ontwikkelingsmodel bepalen. Dit punt is niet zozeer voor beginnende gebruikers van belang maar meer voor gebruikers met een hoge inclusie.

Bij deze indeling zijn een aantal aanvullende opmerkingen van belang:

In de eerste plaats is de openheid van een computerprogramma geen kwestie van een dichotomie tussen wel of niet open zijn, maar is er sprake van een continu spectrum van het ideaaltypische volledig open programma via verschillende gradaties naar het ideaaltypische volledig gesloten programma. Hoe meer van de bovenstaande kenmerken uit de opsomming een programma bezit, hoe opener het is.

Op de tweede plaats is het voor de openheid van een programma niet van belang of een programma gratis is, of dat er geld voor wordt gevraagd. Zo is Microsoft Internet Explorer gratis te downloaden, maar voor de rest volledig gesloten, terwijl je GNU software officieel kan kopen, alhoewel het bijna volledig open software is. Commercieel en open hebben dus strikt gezien niet zoveel met elkaar te maken. Met open programma’s wordt net zo goed geld verdiend, zelfs door beursgenoteerde bedrijven zoals Red Hat of IBM. Tegelijkertijd zijn gesloten programma’s soms gratis (bijvoorbeeld Microsoft Internet Explorer, Eudora, Opera, shareware en tryware in het algemeen).



Illustratie van verschillende categorieën software die je 'gratis' kunt downloaden (bron: Stallman 1996)

Op de derde plaats zou ik open ondersteuning als een zesde aanvullend kenmerk van openheid kunnen noemen. Toch doe ik dat niet, omdat open ondersteuning naar mijn mening vooral door initiatieven van gebruikers ontstaat, en niet zozeer iets is dat door de maker van een programma wordt ingebouwd. Met andere woorden, open ondersteuning hangt niet per definitie samen met de openheid van een computerprogramma. Zo zijn gebruikers vrij om op internet een eigen open ondersteuningsgroep te beginnen voor een volledig gesloten programma. Dat gebeurt ook wel, nieuwsgroepen op internet zijn hiervan een voorbeeld. Toch bestaat er meestal een correlatie tussen de totale openheid van een computerprogramma en de succesvolheid van een gebruikersinitiatief voor open ondersteuning. Vooral de toegang tot de broncode en toegang tot de programmeurs van een computerprogramma zijn hierbij van groot belang. Daarnaast is de open ondersteuning waarbij gebruikers en ontwikkelaars elkaar helpen bij het oplossen van problemen vaak een gevolg van de open cultuur binnen de gebruikersgemeenschap van open programma's.

§ 4.2 Open gebruik

In het vorige hoofdstuk heb ik beschreven hoe gesloten computerprogramma's de gebruiker van zijn creativiteit ontdoen en in het keurslijf dwingen van het script van de passieve 'gebruiker'. Het kan echter ook anders. Zo evalueert Soogelee in "Van één inscript naar

scenario's" (1996) manieren om technologie meer open te maken voor gebruikers, onder andere op het gebied van de informatietechnologie. Zo hoopt zij een oplossing aan te kunnen dragen voor de ondemocratische sturing die in computerprogramma's zit ingebouwd en waaraan de gebruiker hoegenaamd niets kan doen. Hierover schrijft Soogelee somber:

> "Naast het ontbreken van een definitie van participatie [van computergebruikers bij het ontwerpen van computerprogramma's] wordt zelden expliciet aangegeven hòe gebruikers geselecteerd worden voor dit designproces. ... Duidelijk is wel, dat de insluiting van de ene groep gebruikers tegelijkertijd uitsluiting van een andere groep inhoudt. ... Het democratische argument kan moeilijk geëvalueerd worden omdat in de 'participatory design' projekten de besluitvormingsprocedures zelden expliciet beschreven worden. ... [O]ver het algemeen hebben gebruikers weinig tot geen mogelijkheden om hun visie op een ontwerp door te drukken tegen de visie van ontwerpers in. Bij overeenkomende visies, consensus, is gebruikersinbreng in besluitvorming geen probleem. De vraag blijft in hoeverre gebruikers ook inderdaad een machtspositie kunnen gaan innemen in een ontwerpproces. ... Stel er is bij gebruikers consensus mogelijk over een ontwerp, dan is het vervolgens de vraag of deze gebruikerseisen te verenigen zijn met technische en commerciële eisen vanuit de producent van het artefact. Er kan immers sprake zijn van een welbewuste keuze van de zijde van de fabrikant. Door bij een kopieermachine bijvoorbeeld te kiezen voor een tonertechniek die gebruikers niet zelf kunnen hanteren, blijven de kopieermachinehouders afhankelijk van een onderhouds- en servicedienst van de fabrikant ... Vooralsnog kan gesteld worden, dat gebruikers in ontwerpprocessen in de informatietechnologie een marginale rol spelen. Zij worden op zijn best in een ontwerpersgroep gerepresenteerd door intermediairs, zoals: de ergonoom, de psycholoog of de antropoloog (Soogelee, 1996: 24-26)."

Voor een rechtstreekse inbreng van de gebruiker bij het ontwerpen van gesloten commerciële software is blijkbaar geen of nauwelijks plaats in het ontwerpproces. Niet omdat dat technisch niet mogelijk zou zijn, maar omdat gebruikerseisen vaak moeilijk te verenigen zijn met commerciële eisen.

Soogelee draagt echter ook een oplossing aan voor de problemen van het betrekken van gebruikers bij het ontwerpproces, namelijk gebruik maken van meerdere scenario's als vervanging voor het inbouwen van slechts één inscript van een toekomstige gebruiker, waarbij het toekomstig gebruik van een technologie open wordt gehouden en de technologie pas volledig gecomplementeerd is als de gebruiker daar zelf invulling aan geeft en daarmee onderdeel wordt van de technologie zelf. Soogelee concludeert:

> "Ook in de informatietechnologie is een ontwerpen van scenario's mogelijk. Producten worden in modules aangeboden, die door gebruikers in hun eigen omgeving geconfigureerd kunnen worden naar lokale criteria. In de structuur van een product moeten er maatregelen genomen worden om om te gaan met onbekendheid; ruimte ingebouwd voor verandering, voor interactie met gebruikers. Software moet verschillende sturingsprogramma's voor gebruikers hebben: visueel met iconen, met spraak, tekstueel of met toetsenbordformules. De machine moet leren en communiceren, moet zich laten beïnvloeden en veranderen. Het definitieve 'inscript' van de machine wordt dan pas vastgesteld in het gebruik ervan. Sterker nog de machine zou pas af zijn en gebruikt kunnen worden doordat gebruikers hun recht uitoefenen om in het gebruik door het maken van keuzes de machine te complementeren (Soogelee 1996: 72)."

Een mooie conclusie, waar ik het geheel mee eens ben. In feite pleit Soogelee voor open en flexibele, door de gebruiker zelf te configureren computerprogramma's. Helaas is deze strategie niet of moeilijk toe te passen bij de huidige gesloten programma's. Wanneer programma dichtgesealde 'zwarte dozen' blijven, dan kan de gebruiker onmogelijk zelf vorm geven aan de door hem gebruikte software.

Hierbij moet opgemerkt worden dat veel zogenaamde ‘free software’ en ‘open source software’ ook niet altijd open is in haar gebruik. Zo ontbreekt bij gesloten software weliswaar vaak een rechtstreekse toegang tot de interne werking door middel van ‘command line interface’, waardoor de veeleisende computergebruiker gefrustreerd raakt omdat hij gedwongen wordt op inefficiënte wijze zijn computerprogramma te besturen. Maar omgekeerd bezitten veel traditionele open software in de vorm van vrije en open broncode programmatuur op hun beurt vaak weer geen mooie grafische gebruikersinterface, waardoor het gebruik van deze programma’s door relatief onervaren gebruikers wordt bemoeilijkt. Overigens kan dit laatste euvel eenvoudig worden verholpen, omdat het minder werk is om een grafische schil rondom een tekstregelgestuurd programma te bouwen, dan omgekeerd.

§ 4.3 Open standaarden

Het probleem van de afhankelijkheidsrelatie van consumenten ten opzicht van de fabrikant van computerprogramma’s, wordt in de theoretische literatuur die er over dit onderwerp bestaat, het ‘vendor lock-in’ probleem genoemd. Dit probleem is geenszins een exclusieve uitvinding van de Microsoft-fabriek. Al in de jaren zeventig waren computergebruikers vaak gebonden aan fabrikanten; al was toen vooral de onderling incompatibele hardware, in plaats van de software, de veroorzaker van de afhankelijkheidsrelatie.

Een eerste oplossing voor het vendor lock-in probleem is standaardisering. Door het garanderen van meerdere alternatieve routes, wordt voorkomen dat één partij de complete controle over een gehele gebruikspraktijk kan uitoefenen. Egyedi schrijft hierover:

> “Public standards activities in the area of computers started out by addressing the problem of supplier dependency resulting from de facto standards: customers were tied to the products of their initial supplier and could not switch systems without incurring heavy costs. Soft- and hardware products were not exchangeable. ... Standardization could forestall the proliferation of ad hoc solutions. In the 1980’s this resulted in standards activities which focused on ‘open systems’. Open systems are “(...) computer environments that are based on de facto or international standards, which are publicly available and supplier independent” (Egyedi 1996: 11).

Abbate gaat dieper in op het concept van deze ‘open systemen’ in “‘Open Systems’ and the internet” (1995). De centrale stelling van haar artikel is dat het internet nooit in haar huidige vorm had kunnen ontstaan, zonder de openheid die centraal stond in de cultuur waarin de technologieën werden gemaakt, die de basis vormen voor het huidige internet (Abbate 1995). Abbate analyseert de betekenis van de term “open systemen” binnen discussies over de technische architectuur van de infrastructuur van het internet vanaf haar beginjaren tot haar huidige vorm.

Hierbij configureert ook Abatte overigens een type gebruiker, namelijk een gebruiker als consument in een vrije-markteconomie van vraag en aanbod. Abatte definieert openheid

voornamelijk in termen als keuze, vrije markt, consument en concurrentie. Keuzevrijheid en participatie om tot een democratische technologie te komen, bestaan bij Abatte dan ook niet zozeer uit een toegang hebben tot de interne werking van die technologie, maar uit het bezitten van een keuzevrijheid om tussen alternatieven te kunnen kiezen. Hierbij kan de interne werking van de technologie voor de gebruiker gesloten blijven.

Abbate beschrijft als eerste hoe het concept 'open systeem' in de jaren zeventig haar intrede deed in het vakgebied van computernetwerken. Hierbij benadrukt ze dat het concept 'openheid' niet alleen te maken heeft met technische aspecten, maar "it also tried to reshape social aspects of the system ... it explicitly links technical choices with social consequences." (ibid.: 1).

Alhoewel de term "openheid" vaak wordt gebruikt in discussies over computerprogramma's en netwerksystemen, wordt zij zelden precies gedefinieerd. Abbate beperkt zich in haar artikel tot openheid in relatie tot computernetwerken. Alhoewel ik het in mijn scriptie over computerprogramma's heb, zijn de conclusies die Abbate uit haar analyse van openheid in relatie tot computernetwerksystemen trekt ook relevant voor computerprogramma's. Zo zou je het complex van onderling afhankelijke en met elkaar informatie uitwisselende computerprogramma's die gelijktijdig op een enkele computer draaien ook als een soort netwerksysteem kunnen beschouwen.

Daarnaast is de cultuur van openheid afkomstig uit de netwerkgemeenschap van het internet, onlosmakelijk verbonden met de opkomst van open software. Het internet is gebouwd op open software en open standaarden, en omgekeerd nam de ontwikkeling van open software pas echt een grote sprong, nadat internet door commerciële ontsluiting ook echt publiekelijk toegankelijk werd.

Een van de sleutelbegrippen bij open systemen is het begrip modulariteit**.** Door een complex systeem, zoals een computernetwerk, dat uit verschillende hardware- en software-onderdelen bestaat, inclusief alle (fysieke en softwarematige) verbindingen daartussen, op te delen in verschillende componenten, kan het een stuk eenvoudiger beheerd worden. Hierbij kan elk onderdeel door andere onderdelen als een 'black box' beschouwd worden. De ene component hoeft niets te weten over de interne werking van een andere component. Zij hoeft slechts te beschikken over informatie over de taken die een andere component vervult en over de invoer en de uitvoer die zij kan verwerken. Een dergelijk modulair systeem maakt simpele verbindingen ('interfaces') mogelijk tussen de verschillende onderdelen. Modulariteit alleen is echter niet genoeg om van een open systeem te kunnen spreken.

Een tweede voorwaarde waaraan voldaan dient te worden, is het publiekelijk beschikbaar

zijn van de specificaties van de functies van en de verbindingen tussen de verschillende componenten. Want als een fabrikant de specificaties van een bepaald systeem geheim houdt, kan alleen hij en niemand anders onderdelen voor dit systeem bouwen. Pas als de specificaties voor iedereen beschikbaar zijn, kan ieder willekeurig ander bedrijf haar eigen componenten ontwerpen, die vervolgens kunnen interacteren met het groter geheel. De term "open systeem" geeft dus aan dat de componenten van een systeem zowel modulair als publiekelijk gespecificeerd dienen te zijn. De term openheid zoals de 'netwerk'-gemeenschap die gebruikt, legt vooral de nadruk op het probleem van de interoperabiliteit. Openheid is een doel dat nagestreefd dient te worden, en het systeem dient aangepast te worden om dit doel te bereiken.

Het idee van open systemen is gebaseerd op een aantal concepten uit het vakgebied van de software ontwikkeling, waarbij de nadruk ligt op het laten samenwerken van verschillende onderdelen. Deze concepten zijn portabiliteit, compatibiliteit en standaardisering.

Portabiliteit is een term die vooral betrekking heeft op computerprogramma's. Hiermee wordt bedoeld dat een bepaald onderdeel gemakkelijk van het ene systeem naar het andere systeem kan worden overgeplaatst, zonder dat daarvoor sprake is van uitvoerige aanpassingen. Je zou de term in het Nederlands kunnen vertalen met overdraagbaarheid. Portable producten zijn zowel hardware- als software-onafhankelijk.

Met compatibiliteit wordt bedoeld dat producten van de ene fabrikant samen kunnen werken met die van een andere fabrikant. Een Nederlandse vertaling van de term zou verenigbaarheid kunnen zijn. Verenigbare producten kunnen direct volgens een voorgeschreven specificatie werken. Deze werking kan ook tot stand komen doordat een fabrikant bij zijn gesloten product een toevoeging levert, waardoor het kan functioneren volgens de specificaties van een open systeem.

Standaardisering betekent dat onderdelen die aan dezelfde standaard voldoen over dezelfde verzameling functies en 'interfaces' beschikken. Twee gelijksoortige onderdelen, van twee verschillende fabrikanten, die aan dezelfde standaard voldoen, kunnen dus intern volledig anders zijn opgebouwd. Hierbij vallen twee soorten standaarden te onderscheiden.

Ten eerste de zogenoemde "de facto" of gesloten standaarden, die eigenlijk geen standaarden zijn, maar resulteren in een gebruikspraktijk waarbij een gesloten product een overheersende marktdominantie heeft weten te verkrijgen. Ten tweede zijn er de zogenoemde "de jure" of formele standaarden, die door officiële standaardiseringsorganisaties worden ontwikkeld.

Abbate beschrijft verder hoe open systemen door veel mensen als een noodzaak worden

gezien om twee belangrijke doelen te verwezenlijken, namelijk interoperabiliteit en concurrentie (ibid.: 2).

Interoperabiliteit houdt in dat onafhankelijk van elkaar ontwikkelde systemen of componenten met elkaar kunnen werken en informatie kunnen uitwisselen, zonder dat er sprake is van een verlies aan functionaliteit. Openheid als een standaard herstructureert hierbij de markt voor netwerkproducten door concurrentievermogen te definiëren als het vermogen om producten van andere fabrikanten in te sluiten, in plaats van uit te sluiten. Hierdoor bieden open systemen de mogelijkheid voor kleine producenten om producten te leveren voor een bepaalde markt, en op die markt die concurrentie aan te gaan met grote, dominante marktpartijen. Voor grote marktpartijen die hun aandeel aan gesloten systemen te danken hebben, zoals IBM of Microsoft, zijn open systemen minder aantrekkelijk, doordat zij daardoor veel meer concurrentie ondervinden. Zij zijn dan ook vaak tegen open systemen of weigeren er slechts met tegenzin aan mee te werken.[28]

Vooral voor 'gebruikers' (onder deze brede term vallen zowel individuen, als bedrijven en organisaties) bieden open systemen voordelen, namelijk een grote keuzevrijheid en gebruiksgemak. Doordat open systemen aan standaarden voldoen, kunnen ze zonder moeite vervangen worden. Zo kun je tegenwoordig iedere willekeurige telefoon op iedere telefoonwanddoos aansluiten. Daarnaast bieden open componenten een simpeler gebruik, omdat een gebruiker niets hoeft af te weten van de interne werking van een product. Als voorbeeld kun je hier een CD-speler nemen, die bij de meeste merken op vergelijkbare wijze te bedienen is.

Tenslotte maken open systemen het mogelijk voor gebruikers om producten te kopen op basis van prijs en kwaliteit, hetgeen de onderlinge concurrentie tussen fabrikanten van een gelijksoortig product bevordert. Doordat alle producten aan dezelfde standaard voldoen, en daardoor over dezelfde functionaliteit beschikken, hebben gebruikers de keuze om producten van verschillende fabrikanten te betrekken, zonder dat ze afhankelijk zijn van één enkele fabrikant. Hierdoor verschuift de macht van fabrikanten, die wanneer zij in een monopoliepositie zitten, een zelf bepaalde kwaliteit en functionaliteit kunnen leveren, tegen elke zelf bepaalde prijs, naar meer macht voor de gebruikers, die wanneer ze een product te duur vinden, kunnen overstappen op een gelijkwaardig, beter of goedkoper alternatief van een andere fabrikant. Alhoewel de functionaliteit van een bepaald 'open' onderdeel precies

[28] Inmiddels heeft IBM zich omgevormd van hardwareleverancier naar consultant voor het leveren van totale oplossingen op het gebied van ICT, en steunt het bedrijf open programma's zoals de webserver Apache of het GNU/Linux systeem met honderden miljoenen dollars aan ontwikkelingskosten per jaar.

moet voldoen aan publiekelijk beschikbare specificaties, ligt er niets vast over de interne werking van dat onderdeel. Dit kan geheim, gesloten en zeer verschillend van gelijksoortige onderdelen van andere fabrikanten zijn. Het onderscheid tussen de interne en de externe structuur bepaalt dus de grens tussen publieke en private controle op een bepaald systeem.

Het is volgens Abbate vooral aan de ontwikkeling van het ARPA-net tot het huidige internet te danken, dat het open systeem gedachtegoed tegenwoordig zo invloedrijk is. Hierbij is het concept van openheid tegenwoordig steeds belangrijker, omdat we door steeds meer technologie omringd worden. Openheid als strategie om de macht van fabrikanten in te perken heeft een grote politieke aantrekkingskracht omdat computernetwerken worden gezien als een basale infrastructuur die voor iedereen toegankelijk zou moeten kunnen zijn. Hierbij zijn de belangen van fabrikanten om controle over hun producten uit te kunnen oefenen ondergeschikt aan het algemeen belang om zodoende tot een vrije markt te komen voor compatibele netwerkproducten.

In de huidige debatten over open systemen ligt de nadruk dan ook op de vraag hoe openheid nagestreefd kan worden bij de ontwikkeling van standaarden door een bepaalde bedrijfstak. Hierbij is de nadruk op het technologische domein verschoven naar het juridische en regulatoire domein. De vraag is dan ook wanneer een door een bepaald bedrijf ontwikkelde standaard als open kan worden beschouwd en wie er invloed kan en mag uitoefenen bij het tot stand komen van deze standaarden. “Everyone supports ‘openness’, but each firm ... has its own definition of ‘openness’ that supports its narrow commercial interests”, zo citeert Abbate Jonathan Band die het probleem van het openheidconcept samenvat. In haar conclusie verwoordt Abbate hoe bij het concept van open systemen niet alleen technische, maar ook sociale factoren van invloed zijn:

> “ ... ‘openness’ points to a link between technical decisions (interface standards) and social outcomes (more competitive markets, increased freedom and control for users). Openness has become politically appealing as well as controversial precisely because it alters technical characteristics of the system in ways that may shift the balance of power and threaten entrenched interests. The enduring influence of the open systems concept illustrates the power of a model to identify aspects for a technological system as sites for political intervention. (Abbate 1995: 8)”

Abbate reikt ons met haar publicatie theoretische handvaten aan om te beoordelen in hoeverre en op welke manier een systeem, of in ons geval computerprogramma, open is. Daarmee biedt zij vooral een antwoord op de vraag van het vendor-lock-in probleem. De software zelf echter kan een gesloten black box blijven. Gebruikers hebben met ‘open’ programma’s volgens de definitie van Abbate dus nu de keuze tussen computerprogramma’s van verschillende producenten, maar kunnen de programma’s zelf niet veranderen. Want toegang tot de inhoud van de black box is geen kwestie waarmee Abbate zich in deze publicatie

uitvoerig bezig houdt. Een oplossing die wel binnen dit scenario past, is het zelf bouwen van een programma volgens de publiekelijk beschikbare specificaties en standaarden. Helaas is dit zelf bouwen niet voor iedereen weggelegd door het ontbreken van tijd of expertise.

Een succesvol voorbeeld van een systeem waarin we het gedachtengoed terugvinden van de open systemen (zowel qua standaarden als qua beschikbaarheid van broncode) is het internet. Waarom kon internet een succes worden? Juist omdat het een open technologie was, die door de opdrachtgevers, ontwerpers, bouwers en technici zo open en flexibel mogelijk werd gehouden en zo verschillende gebruiken naast elkaar toeliet. In "Inventing the Internet" over de geschiedenis van de onderliggende infrastructuur van het internet, verklaart Abbate het hedendaagse succes van het internet aan de hand van het feit dat het hier om een 'open systeem' gaat, in de definitie zoals we die eerder zijn tegengekomen (Abbate 1999). De TCP/IP technologie liet applicaties een zeer grote vrijheid in wat ze konden doen, en liet zo de deur open voor nog te ontwikkelen gebruik en nieuwe technologieën.[29]

Een voorbeeld van een gebruik van het flexibele en open Internet TCP/IP protocol is het Wereld Wijde Web, de toepassing die het Internet heeft doen doorbreken bij het grote publiek. Het Wereld Wijde Web is expliciet gebouwd vanuit de gedachte dat het produceren van www-informatie net zo gemakkelijk moet zijn als het consumeren ervan. Tim Berners-Lee, de uitvinder van het WWW, ontwierp dit systeem namelijk expliciet om het uitwisselen van informatie tussen incompatibele computersystemen mogelijk te maken.[30] Wanneer we elke individuele webpagina beschouwen als een klein computerprogramma, dat de computer de opdracht geeft voor het weergeven van informatie op een beeldscherm op een hypertekst achtige manier, dan krijgt een computerprogramma bij deze programmaatjes er ook nog eens de broncode bij, zodat hij niet alleen de beschikking heeft over de eigenlijke informatie, maar ook over de commando's die bepalen hoe deze informatie wordt afgebeeld. En daarmee beschikt elke lezer van een internetpagina ook meteen over de broncode ervan, dit keer niet in C, Pascal of Basic geschreven, maar in HTML, de programmeercode voor het WWW.

Door deze bijgeleverde broncode wordt het produceren van webpagina's bijna net zo gemakkelijk als consumeren.[31] Want aan de hand van voorbeelden uit de praktijk kan iemand leren om zelf HTML-code te programmeren. De stap van gebruiker naar maker wordt op

[29] Lessig werkt dit idee overtuigend uit in zijn boek "Code and other laws of cyberspace" (1999).
[30] Zie Berners-Lee 2000 en Gilles & Cailliau 2000.
[31] Een systeem dat dit idee technisch implementeert is het Wiki-project (http://www.wiki.org/), door de makers zelf ook wel 'open editing' genoemd:

"Allowing everyday users to create and edit any page in a Web site is exciting in that it encourages democratic use of the Web and promotes content composition by nontechnical users."

deze manier veel kleiner gemaakt. Zelfs door simpel knip en plak werk kan op deze wijze al snel een fraaie pagina in elkaar geknutseld worden door iemand die slechts over een basale kennis van HTML beschikt. Het WWW op internet kon zo'n hoge vlucht nemen, omdat iedereen meteen kon zien hoe je 'coole' trucs kon uithalen of mooie pagina's kon maken, simpelweg door naar de HTML-code van het bestand te kijken.

Daarnaast zorgde Tim Berners-Lee er ook nog eens voor, dat de HTML-specificaties een open standaard waren, die voor iedereen gratis van Internet te downloaden waren. Zoals het bij andere standaarden wel vaker gebruikelijk is, dat je eerst ergens tegen betaling lid van moet worden, of dat je geld moet betalen voor het inzien van een standaard was dit bij het WWW niet het geval.

Integendeel, Berners-Lee zorgde ervoor dat een speciale programmeerbibliotheek ('libwww') vol voorgeprogrammeerde 'Lego-stenen' met WWW-functies in de populaire programmeertaal C, die voor bijna elk denkbaar computerplatform beschikbaar is, gratis beschikbaar was om het zelf schrijven van webbrowsers te faciliteren.

§ 4.4 Gereedschapskisten voor gebruikers

De combinatie van het gebruik van open standaarden en het expliciet ontwerpen van programma's voor het faciliteren van een open gebruik, biedt uiterst krachtige mogelijkheden voor gebruikers. Zo biedt het besturingssysteem Unix de gebruiker een omgeving waarin hij relatief makkelijk zijn eigen programma's kan bouwen door het aaneenschakelen van meerdere kleine standaardprogramma's tot een groter functioneel geheel. Daardoor kan een gebruiker het gebruik van zijn eigen computer zelf vormgeven, in de betekenis zoals we die bij Soogelee zijn tegengekomen. Ook kan een gebruiker door de modulaire opbouw van zijn besturingssysteem naar eigen wens subonderdelen vervangen.

In het ontwerp van Unix zijn technische keuzes ingebouwd die verstrekkende gevolgen hebben voor het scenario van mogelijkheden die de ingebouwde gebruiker tot zijn beschikking heeft. Maar voor ik daartoe kom, zal ik eerst kort uitleggen wat Unix nu eigenlijk precies is.

Unix is een computerbesturingssysteem dat in 1969 werd ontwikkeld als opvolger voor het megalomane besturingssysteem Multics, dat nooit echt uit de fase van onderzoeksproject is gekomen door de te hoge ambities en te grote complexiteit van het project. Ook al ontstond de eerste versie van Unix meer dan dertig jaar geleden, toch is Unix nog steeds 'alive and kicking' in de computerwereld. Nog steeds bieden commerciële bedrijven zoals Compaq, IBM, Silicon Graphics en Sun hun eigen Unixvariant aan. Daarnaast zijn er inmiddels ook

meerdere vrije Unixvarianten ontwikkeld, zoals FreeBSD, NetBSD en OpenBSD en het bekendere GNU/Linux. Het nieuwste besturingssysteem van Apple, OS X is ook gebaseerd op Unix. Nog steeds worden de ontwerpprincipes achter Unix als zinnig en navolgenswaardig beschouwd. Na een dip in de jaren negentig waarin op grote schaal van Unix naar het Windows-platform werd overgestapt, mogen Unix-varianten zich de laatste jaren weer in een toenemende belangstelling verheugen. Vooral voor servers, wordt Unix als besturingssysteem als beste keuze gezien. Voor gebruik op de desktop zou het besturingssysteem te moeilijk zijn is de overheersende overtuiging echter.

Unix vormt als besturingssysteem van een computer de softwarematige infrastructuur die zorg draagt voor de interactie tussen gebruikersprogramma's en de computerhardware, zoals geheugen, harddisk, floppydrive, printer, netwerkkaart of modem, videokaart, beeldscherm en invoerapparatuur zoals muizen en toetsenborden. Hierdoor hoeft niet ieder programma zelf het wiel opnieuw uit te vinden voor de omgang met de computerhardware, doordat een heleboel functies hiervoor al in het besturingssysteem ingebouwd zitten. Wat Unix echter onderscheid van de meeste andere besturingssystemen, is dat het als een soort gereedschapskist voor computerprogrammeurs is geconstrueerd. Het probeert het schrijven van efficiënte programma's zoveel mogelijk te faciliteren. Over de achterliggende 'Unix-filosofie' schrijft Kernighan het volgende:

> "Omstreeks 1974 kregen universiteiten de licentie om [UNIX] voor onderwijsdoeleinden te gebruiken en een paar jaar later kwam het voor commercieel gebruik ter beschikking. ... Sindsdien wordt UNIX gebruikt op tienduizenden installaties over de hele wereld, van microcomputer tot de grootste machines toe. Waaraan is dit succes te danken? We kunnen verschillende redenen onderscheiden. Omdat het in C geschreven is, is het ten eerste overdraagbaar - UNIX is te gebruiken voor microcomputers èn voor de grootste rekenmonsters, wat een groot commercieel voordeel is. Ten tweede is het bronprogramma beschikbaar en is het in een hogere taal geschreven, waardoor het systeem gemakkelijk is aan te passen voor bijzondere toepassingen. Maar het belangrijkste is wel dat UNIX een goed besturingssysteem is, vooral voor programmeurs. De UNIX programmeeromgeving is ongewoon overvloedig en produktief. Hoewel het UNIX-systeem een aantal programma's en technieken kent, die blijk geven van een grote vindingrijkheid, is het niet door een bepaald programma of idee dat het zo goed werkt. Wat het in plaats daarvan zo effectief maakt, is een zekere aanpak van het programmeren, een filosofie van het computergebruik. Hoewel die filosofie niet in één enkele regel kan worden omschreven, is zij gebaseerd op de gedachte dat de kracht van een systeem meer te danken is aan de relatie tussen programma's dan aan de programma's zelf. ... (Kernighan 1986: 297)
> Laat verder een deel van het werk door anderen doen. Gebruik bestaande opdrachten als bouwstenen voor uw programma en smeed ze via de schil en de programmeerbare filters tot één geheel. ... De UNIX-werkomgeving biedt tal van hulpmiddelen die op legio manieren met elkaar gecombineerd kunnen worden; vaak hoeft u dan ook niet veel meer te doen dan de juiste combinatie te bedenken. (ibid.: vii)"

Gancarz drukt deze laatste gedachte van 'softwarebricolage' in zijn boek 'The Unix Philosophy' nog explicieter uit: "...good programmers write great software; great programmers 'steal' great software (Gancarz 1995: 2)." In plaats van een passief consumptieve houding wordt er in Unix een productieve en creatieve houding van gebruikers verwacht. Juist door het combineren van kleine en eenvoudige programma's ontstaat een

krachtig groter geheel. Het gebruik van een Unix-computer is dus als het ware een continue bezigheid van het combineren van verschillende componenten.

Wanneer we Unix in het licht van open standaarden en een open gebruik bekijken, valt het volgende op:

Unix is een open systeem. Niet alleen werd Unix geleverd met een ingebouwd documentatieprogramma waarin zelfs de fouten in programma's stonden vermeld, ook kreeg je als ultieme vorm van programmadocumentatie bij Unix de broncode van alle systeemprogramma's erbij.[32] Dit was mogelijk, omdat Unix ontwikkeld werd op het research- en developmentlaboratorium van de Amerikaanse telefoonmaatschappij AT&T, en via rechtswege geen geld mocht verdienen met het verkopen van computerprogramma's. Daarom gaf AT&T Unix en de bijbehorende C-broncode gratis weg aan andere onderzoeksinstellingen en universiteiten. Door de aanwezigheid van broncode van het door hen gebruikte besturingssysteem, raakten gebruikers gewend aan een bepaalde programmeer- en gebruiksstijl, waarbij het gebruiken van programmeerwerk van anderen en het beschikbaar stellen van de broncode van je eigen programma's de norm was

Unix is portabel. Doordat Unix in een hogere programmeertaal geschreven is, kan het naar bijna ieder computersysteem worden overgezet (wat tegenwoordig met NetBSD en Linux ook daadwerkelijk gebeurt). Moderne Unix-varianten zoals Linux en NetBSD kunnen op tientallen verschillende soorten computers werken, van een oude Amiga of Atari-computer tot de meest uitgebreide IBM mainframe en alles wat daar tussen zit. Er is geen enkel ander besturingssysteem dat op zo veel verschillende hardwareplatformen werkt. Ter vergelijking: Windows werkt slechts op één soort computer, namelijk een Intel-i386-compatible computer.

Unix is extreem flexibel. Zowel door de ontwerpgedachte als door de aanwezigheid van de broncode.

Unix faciliteert het zelf programmeren. Unix is expliciet ontworpen vanuit de gedachte om het schrijven van computerprogramma's zoveel mogelijk te faciliteren. De ingebouwde gebruiker schrijft zelf zijn eigen programma's.

Een veelgebruikte metafoor stelt Unix voor als een enorme doos met Legostukken, waarmee je zelf kan bouwen wat je wil. Met de simpele basisstenen kun je grote en complexe bouwwerken realiseren. Eerst aan de hand van bouwtekeningen, later op basis van ervaring aan de hand van eigen ontwerpen. Een besturingssysteem als Unix is helemaal ingesteld op

[32] Althans bij de BSD-versie van Unix. Hierover is later een rechtszaak gevoerd tussen de University of Berkeley en Unix System Laboratories. Zie: McKusick, 1999.

het zelf schrijven van nieuwe programma's, doordat alle componenten die hiervoor nodig zijn standaard al aanwezig zijn, en makkelijk met elkaar te combineren zijn. Dit heeft echter als nadeel, dat Unix-systemen door beginnende gebruikers als moeilijk en intimiderend worden ervaren, omdat zolang je niet over de basiskennis beschikt om met de bouwstenen om te gaan, je niet makkelijk of snel iets gerealiseerd krijgt.

Bij Unix wordt standaard een hele ontwikkelomgeving voor verschillende programmeertalen (waaronder C en C-plus-plus) meegeleverd, de taal waarin Unix zelf ook is geschreven. Vooral het idee van een besturingssysteem als blokkendoos, waarvan je zelf kunt bouwen wat je wilt, zie je terug in de open software wereld waar een grote nadruk op modulariteit (kleine bouwstenen die op verschillende manieren te combineren zijn) en het zelf schrijven of aanpassen van programma's ligt. Een groot deel van de open softwarewereld werkt tegenwoordig met een besturingssysteem (zoals Linux of FreeBSD) waar Unix en de achterliggende ontwerpfilosofie voor aan de basis hebben gelegen.

De analogie tussen een besturingssysteem zoals Unix en een gereedschapskist die het een gebruiker mogelijk maakt om zelf nieuwe constructies te bouwen die nauw bij zijn eigen behoeften aansluiten, wordt op een meer theoretisch niveau gebruikt door Eric von Hippel in zijn artikel 'Horizontal innovation networks - by and for users' waarin hij het concept "toolkit for user innovation" introduceert (Hippel & Franke 2002: 4). Von Hippel legt uit dat het bijna een economische noodzaak is dat producenten alleen aan de wensen van de "gemiddelde" gebruiker van een bepaald marktsegment kunnen voldoen, om zo een zo groot mogelijke markt tegen zo laag mogelijke kosten te kunnen bedienen. Het grote minpunt is dat een product hierdoor nooit aan alle wensen van alle gebruikers kan voldoen. Want net zomin als de 'l'homme moyen' daadwerkelijk bestaat, zullen we de 'gemiddelde consument' in fysieke vorm in een winkelstraat tegen het lijf lopen.

Een manier om zowel tegemoet te komen aan de heterogene consumentenbehoeftes, als ook te kunnen profiteren van de prijsvoordelen van een standaardproduct voor een grote groep gebruikers, is het voorzien van gebruikers van zogenaamde 'toolkits for user innovation', vrij vertaald, gereedschapskisten voor gebruikers, waarmee ze zelf producten aan hun specifieke wensen en eisen kunnen aanpassen.

Als je een beetje handig en technisch bent kun je aan objecten uit de fysieke wereld vaak nog wel je eigen aanpassingen aanbrengen. Bij software die je in een dichtgelaste vorm, als black box krijgt, valt echter helemaal niets meer te veranderen. De mogelijkheid om als gebruiker je eigen 'anti-programma' te creeëren is heel erg gering. Bij software waar je echter de broncode bijkrijgt, kun je zo veel veranderen als je zelf wilt (en waartoe je

technisch gezien in staat bent). Je zou kunnen zeggen dat software zonder broncode op een andere wijze inclusie in een technologisch frame vormgeeft, dan open software.

Wanneer je een gebruiker bent, die een hoge inclusie bezit, en de technologie op zich niet meer kan verwerpen, is het bijzonder frustrerend als je vervolgens niets kunt wijzigen aan de 'black box'. Je beschikt in zo'n geval over de vaardigheden en de kennis om het artefact te 'herprogrammeren', maar door het specifieke geval van software die volledig en onomkeerbaar af te sluiten is door hem te compileren en alleen nog maar in binaire voor mensen onleesbare vorm te distribueren, kun je als gebruiker niet meer doen dan iemand die een lage inclusie heeft: namelijk slikken of stikken. Iemand die een lage inclusie heeft zal het geen probleem vinden om computertechnologie compleet te verwerpen, wanneer je echter een programmeur van computerprogramma's bent is dit geen optie.

§ 4.5 Hardhoofdige hackers

Williams beschrijft in zijn biografie over de beroemde computerhacker[33] Richard Stallman hoe deze werd gedreven tot de missie van het schrijven van een compleet open besturingssysteem. Stallman werkte begin jaren tachtig aan het Massachusetts Institute of Technology. Hij beschikte over alle vaardigheden om een technisch artefact te verbeteren, maar dit werd hem onmogelijk gemaakt, terwijl er ook geen alternatief mogelijk was (Williams 2002: 1-8).

De werkplek van Richard Stallman, een getalenteerd computerprogrammeur aan het Artificial Intelligence Lab, had van Xerox een prototype van een nieuw type laserprinter gekregen, die via een netwerk door een hele afdeling tegelijk gebruikt kon worden. Helaas liep bij deze nieuwe printer nogal vaak het papier vast. De enige manier om het probleem op te lossen, was de printer uitschakelen, vervolgens het vastgelopen papier uit het mechaniek halen om daarna de printer opnieuw aan te zetten. Bij deze hersteloperatie raakten alle printopdrachten kwijt die in de tussentijd gegeven waren. Daarbij gaf de printer geen foutmelding, wanneer het printen niet gelukt was door vastgelopen papier of door de bijkomstige herstelwerkzaamheden. Zo kwam het regelmatig voor dat Stallman nietsvermoedend dacht zijn printjes op te halen, om er achter te komen dat de printer was vastgelopen of dat zijn printopdracht gewoonweg was verdwenen, met onnodig heen-en-weer geloop tussen zijn kantoor en de printer als gevolg. Omdat Stallman geen verstand had van printermechanismen, maar wel van computerprogramma's, besloot hij een nieuw

[33] Een 'hacker' is een programmeervirtuoos, een 'cracker' is iemand die zich specifiek richt op het verwijderen van kopieerbeveiligingen van commerciële software, eventueel vanuit het oogpunt van geldelijk gewin.

aanstuurprogramma voor de printer te schrijven, dat de gebruiker een melding zou geven wanneer het afdrukken niet gelukt was. Bovendien had Stallman al jaren eerder een programma geschreven voor een printer met een soortgelijk probleem.

Vol goede moed ging Stallman op zoek naar de broncode van het aanstuurprogramma voor de Xerox-printer, zodat hij zijn eigen aanpassingen zou kunnen maken. Bovendien was het gebruikelijk dat fabrikanten bij prototypes van computers en randapparatuur die ze aan het MIT AI-lab schonken de broncode erbij leverden. Wanneer de technisch goed onderlegde medewerkers vervolgens foutjes uit de apparatuur repareerden, konden fabrikanten die verbetering inbouwen voordat ze hun apparatuur op de markt brachten.

In dit geval kwam Stallman er na een lange zoektocht echter achter dat Xerox doelbewust geen broncode met de printer mee had geleverd. Hiermee kwam een einde aan de traditie van het delen van broncode door bedrijven met het AI-lab. Waar software vroeger een toegevoegde waarde was om meer hardware verkocht te krijgen, was het nu tot een op zich zelf staand product verworden, waar veel geld mee te verdienen viel, en dat onder geen enkel beding in handen van de concurrentie mocht vallen. In plaats van een aardig extraatje, was software in de ogen van de computerindustrie een nieuwe kip met gouden eieren geworden. Diep van binnen kookte Stallman van woede over deze nieuwe gang van zaken. Vooral omdat hij nog mee had gemaakt hoe er ooit een heel andere cultuur de norm was geweest op het AI lab, namelijk de ‘hackercultuur’, zoals die eind jaren vijftig op het MIT ontstond.

In Hackers (1984) beschrijft Levy de opkomst van fanatieke computerknutselaars vanuit de Tech Model Railroad Club (TMRC), waar studenten van het Massachussets Institute of Technology (MIT) zich bezig hielden met modelbouwtreintjes en spoorbanen. In de clubruimte van de TMRC stond een groot treinparcours, inclusief landschap van papiermaché, waarover verschillende treintjes tegelijkertijd konden rijden. De TMRC bestond uit twee afdelingen. Het ene deel bestond vooral uit leden die geabonneerd waren op treintijdschriften en reizen ondernamen naar historische treintrajecten. Deze leden hielden zich vooral bezig met het bouwen en schilderen van replica’s van treinen of versieringen voor de treinbaan. De andere afdeling, het “Signals and Power Subcommittee”, bestond vooral uit mensen die gefascineerd waren door datgene wat de treintjes aanstuurde, het zogenaamde ‘System’, dat voor een groot deel uit onderdelen bestond die door donaties van telefoonmaatschappijen waren verkregen. De leden van deze afdeling waren geobsedeerd door technologie, en hingen urenlang rond op de TMRC terwijl ze constant probeerden om het “Systeem” uit te breiden en te verbeteren. In deze groep kreeg ook het woord ‘hack’ haar speciale betekenis als “a project undertaken or a product built not solely to fulfill some

constructive goal, but with some wild pleasure taken in mere involvement" (Levy, 1984, p. 23). De term werd met respect gebruikt, want "to qualify as a hack, the feat must be imbued with innovation, style, and technical virtuosity." (ibid.). Toen er eind jaren vijftig voor het eerst computers verschenen op het MIT, waren het de leden van de "S&P-group" die het eerste op onderzoek gingen om de grote en dure machines met eigen ogen te aanschouwen. Toen MIT een van de eerste transistorcomputers, de TX-0, in langdurige bruikleen kreeg, verzamelden S&P-hackers zich al snel rond deze computer. De TX-0 was een van de eerste computers die een interactief gebruik mogelijk maakte, hetgeen voor die tijd een ware sensatie was. Al snel ontstond rond de TX-0 een nieuwe manier van omgaan met technologie. Deze impliciete "Hacker Ethic" vat Levy in de volgende regels samen:

1. "Acces to computers - and anything which might teach you something about the way it or the world works - should be unlimited and total. Always yield to the Hands-On Imperative!" (p. 40)
2. "All information should be free." (p. 40)
3. "Mistrust Authority - Promote Decentralization." (p .41)
4. "Hackers should be judged by their hacking, not bogus criteria such as degrees, age, race or position."(p. 43)
5. "You can create art and beauty on a computer." (p. 43)
6. "Computers can change your life for the better." (p. 45)

Ook al schrijft Raymond (1999)[34], dat de MIT-hackers een zeer gesloten clubje waren zonder veel verbindingen met de buitenwereld, waardoor haar invloed slechts beperkt bleef, toch is de MIT-hackerssubcultuur wel degelijk van historisch belang. Want het was deze hackercultuur met haar bijbehorende 'Hacker Ethiek' die Stallman tot een groots softwareproject zou inspireren.

Toen Stallman na het printerincident ook moest toezien hoe begin jaren tachtig bedrijven langzamerhand alle hackers 'overkochten' uit het AI-lab van het MIT waar hij werkzaam is, waarmee de vrije stroom van informatie over computerprogramma's verdween, was Stallman het beu. Hij raakte zodanig gedeprimeerd dat hij als oplossing hiervoor besloot de tegenaanval te openen, door zijn eigen vrije software te schrijven, als reactie op de gesloten commerciële programma's. Niet geheel ambitieloos, besloot hij een geheel vrij besturingssysteem te schrijven onder de naam GNU (GNU is Not Unix), als vervanging voor Unix, dat niet meer vrij was, omdat AT&T vanaf 1984 haar copyrightrechten uitoefende.

Stallman begon zijn project in januari 1984, met het programma Yacc, een vervanging voor het obscure Unix-programma Bison. Daarna publiceerde hij zijn eerder geschreven Emacs teksteditor onder een nieuwe open licentie. Naarmate steeds meer mensen GNU Emacs gebruikten, kreeg Stallman meer feedback van gebruikers. Door Emacs op tape te verkopen, wist Stallman zijn eigen inkomen te creeëren. Gesteund door de enthousiaste

ontvangst van Emacs, probeerde Stallman om zijn eigen GNU C Compiler (GCC) te schrijven, die programma's die in C geschreven zijn, kon vertalen naar binaire programma's. Mede door het commentaar van gebruikers, werd GCC een van de beste programma's in zijn soort. GCC en Emacs zorgden samen voor een steeds grotere bekendheid van het GNU-project. In oktober 1985 richtte Stallman de Free Software Foundation (FSF) op, om het GNU-project in verdere banen te leiden. Personeel van de FSF werd betaald uit de opbrengst van de verkoop van GNU-programma's.

Ongeveer in 1990 waren er nog twee belangrijke programmeerklussen geklaard, GNU Bash (Bourne Again Shell) een programma om Unix commando's te geven, en een vrije GNU "C bibliotheek", met daarin allerlei kleine stukken C-code zodat programmeurs niet steeds opnieuw het wiel uit hoeven te vinden voor relatief simpele en vaak voorkomende programmeerfuncties. Het enige dat ontbrak, voordat er sprake kon zijn van een geheel functionerend GNU Unix-alternatief was de zogenaamde kernel, het hart van een besturingssysteem. De FSF was wel bezig met HURD, de kernel van GNU, maar dit project wilde door vele technische problemen maar niet opschieten.

Behalve het GNU project, waarvan de verschillende onderdelen inmiddels op grote schaal door allerlei soorten programmeurs gebruikt worden, heeft Stallman nog een andere grote bijdrage geleverd aan de wereld van de vrije software, namelijk het 'copyleft', een soort copyright, maar dan voor vrije software. Stallman wilde ervoor zorgen dat de verschillende onderdelen van het GNU project niet alleen vrij waren, maar dat ook in de toekomst bleven. Je kan altijd een programma verspreiden in het zogenaamde 'publieke domein'. Dat betekent dat een programma geen copyright heeft en iedereen er mee kan doen wat hij wil. Ook dat iemand er een commerciële versie van maakt om te verkopen. Stallman wilde niet alleen dat zijn programma's vrij waren, maar ook dat ze dit zouden blijven. In de eigen woorden van Stallman:

> "In the GNU project, our aim is to give *all* users the freedom to redistribute and change GNU software. If middlemen could strip off the freedom, we might have many users, but those users would not have freedom. So instead of putting GNU software in the public domain, we "copyleft" it. Copyleft says that anyone who redistributes the software, with or without changes, must pass along the freedom to further copy and change it. Copyleft guarantees that every user has freedom.
> ...
> To copyleft a program, we first state that it is copyrighted; then we add distribution terms, which are a legal instrument that gives everyone the rights to use, modify, and redistribute the program's code *or any program derived from it* but only if the distribution terms are unchanged. Thus, the code and the freedoms become legally inseparable. Proprietary software developers use copyright to take away the users' freedom; we use copyright to guarantee their freedom. That's why we reverse the name, changing "copyright" into "copyleft" (bron: http://www.gnu.org/copyleft/)."

[34] Raymond schrijft veel over de hackercultuur en is hoofdeditor van 'The New Hacker's Dictionary' (1991)

Om deze vrijheid voor de gebruikers van zijn computerprogramma's te garanderen, ontwierp Stallman in 1989 een specifieke licentie onder de naam General Public License (GPL). Hierin staat in juridische taal precies vastgelegd wat gebruikers wel en niet mogen. Deze vrijheden vat Stallman als volgt samen:

> "The freedom to run the program, for any purpose (freedom 0).
> The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
> The freedom to redistribute copies so you can help your neighbor (freedom 2).
> The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. (freedom 3).
> Access to the source code is a precondition for this.[35]"

Stallman is hiermee de eerste in de geschiedenis van de computer die daadwerkelijk op een systematische manier aan een vrij besturingssysteem begint te werken. Samen met de GPL heeft Stallman daardoor een platform weten te creëeren voor de ontwikkeling van open software.

Daarnaast is Stallman de eerste die uitgebreid heeft proberen bij te dragen aan een "open software ideologie". De eerste aanzet daarvoor is het "GNU manifesto" uit 1985 (http://www.gnu.org/gnu/manifesto.html). Hiermee kan Stallman beschouwd worden als een soort grondlegger van de open software cultuur. Ook al wordt Stallman tegenwoordig door velen gezien als een radicale softwarefundamentalist die principieel weigert met commerciële programma's te werken, toch is zijn invloed nog steeds erg groot. Al was het alleen maar omdat er inmiddels tienduizenden computerprogramma's zijn uitgerust met zijn GPL. Stallman kan dan ook beschouwd worden als de antithese van Bill Gates, die computerprogramma's alleen ziet als een manier om de 'share holder value' van Microsoft-aandelen verder omhoog te stuwen.

Tegenwoordig bestaan er tientallen verschillende soorten licenties voor het vrijgeven van open programma's. De twee belangrijkste daarvan zijn de GPL-licentie en de BSD-licentie. Het fundamentele verschil daartussen zit hem in het feit, dat de GPL-licentie verplicht dat alle door gebruikers zelf aangepaste versies van een 'GPL-ed' computerprogamma onder dezelfde GPL-licentie worden verspreid. Mede door dit virusachtige karakter schrikken veel bedrijven terug voor het gebruiken van GPL-programma's. De BSD-licentie is veel vrijer, en zegt in feite: "hier is de code, doe er mee wat je wilt". Dit schrikt bedrijven veel minder af, omdat ze zelf aangebrachte veranderingen niet hoeven te delen. Op dit moment is het zelfs zo dat in Microsoft Windows hele stukken BSD-broncode zit ingebouwd, zoals de code voor de implementatie van het TCP/IP-protocol.

[35] Zie: http://www.gnu.org/philosophy/free-sw.html

§ 4.6 Doe-het-zelf computergebruikers

Ook al leverde Stallman met zijn GNU-project dus de infrastructuur voor een vrij besturingssysteem dat de gebruiker niet alleen achter het stuur plaatste, maar ook de mogelijkheid bood tot rommelen onder de motorkap, toch lukte het Stallman en zijn aanhang maar niet om een compleet werkend besturingssysteem te realiseren in de jaren negentig. Waar Richard Stallman faalde om een volledig vrij besturingssysteem voor computers te maken, slaagde de Finse informaticastudent Linus Torvalds wel, door het ontbrekende stukje software te schrijven dat nodig was om van alle losse GNU programma's een volledig besturingssysteem te maken, namelijk de zogenaamde 'kernel'.

Nadat Torvalds in 1991 een nieuwe computer had gekocht met een Intel 386 processor, wilde hij experimenteren met de mogelijkheden voor 'multitasking' (dat is het simultaan uitvoeren van meerdere computerprogramma's) die deze nieuwe processor bood. Helaas maakte MS-Dos, het besturingssysteem dat in die tijd standaard bij Personal Computers geleverd werd, geen gebruik van multitasking. Daarom begon Torvalds te experimenteren met een eigen programma dat wel gebruik maakte van de multitaskingmogelijkheden van zijn nieuwe computer. Eerst kon zijn programma niet meer dan afwisselend A's en B's op het scherm afbeelden. Later bouwde hij dit simpele programmaatje uit tot een zogenaamde terminal emulator, een programma waarmee hij kon inbellen op de computer van zijn universiteit om bijvoorbeeld e-mails uit internetnieuwsgroepen te kunnen lezen.

Ondertussen was Torvalds er ook achter gekomen dat Unix een besturingssysteem was, dat multitasking ondersteunde. Dus wilde Torvalds dat graag op zijn eigen computer installeren. Helaas was Unix voor computers met een Intelprocessor niet verkrijgbaar of te duur voor Torvalds als student. Wel beschikbaar was Minix, dat werkte op pc's met een Intelprocessor, en was geschreven door Andrew Tannenbaum, die informatica doceerde aan de Vrije Universiteit in Amsterdam. Minix was een soort mini Unix, dat alleen bedoeld was voor educatieve doeleinden om te laten zien hoe Unix werkt. Tannenbaum stelde wel de broncode van zijn programma Minix beschikbaar, maar hij stond mensen niet toe om Minix aan te passen en vervolgens de aangepaste Minix te verspreiden. Aanvankelijk gebruikte Torvalds Minix om lustig te experimenteren met de mogelijkheden van zijn nieuwe computer. Maar uiteindelijk ergerde hij zich zodanig aan de beperkingen van Minix, dat hij zelf een vervanging voor Minix wilde schrijven. Hij gaf zijn programma de naam Linux.[36]

[36] Oorspronkelijk wilde Linus Torvalds zijn besturingssysteem Freax noemen, maar Arie Lemmke die het FTP-archief beheerde stelde het onder de naam Linux beschikbaar voor de rest van de wereld. Zie: Moody (2001) en Torvalds en Diamond (2001). Omdat Linux alleen de kernel van het besturingssysteem is, en de rest vooral uit programma's van het GNU-project bestaat, pleit Riachard Stallman voor het gebruik van de naam GNU/Linux

Op 17 september 1991 stelde Torvalds Linux versie 0.01 ter download op het internet beschikbaar. De eerste versie werd nog niet door veel mensen gedownload om zelf uit te proberen, maar naarmate Torvalds verder programmeerde aan zijn eigen hobby-besturingssysteempje raakten steeds meer mensen geïnteresseerd. Torvalds kreeg steeds meer emailtjes van mensen die Linux probeerden, maar met fouten of tekortkomingen geconfronteerd werden. Dat hield Torvalds gemotiveerd om verder te bouwen aan Linux. Omdat Torvalds voor het bouwen van Linux gebruik maakte van vrije computerprogramma's van het GNU-project zoals GCC en Bash, besloot hij Linux vanaf versie 0.12 ook onder de GPL van Stallman beschikbaar te stellen. Zelf vond hij het niet meer dan logisch, omdat hij zichzelf vooral zag als iemand "die op de schouders van reuzen stond" en in zijn geval waren dat de mensen die de vrije software hadden geschreven waarmee hij zijn eigen Linux besturingssysteem had kunnen bouwen. (Torvalds & Diamond 2001: 98).

In maart 1994 verscheen Linux versie 1.0. Dit versienummer betekende dat Torvalds vond dat Linux volwassen begon te worden. Ondertussen installeerden steeds meer mensen Linux op hun computer om te gebruiken als serieus besturingssysteem. Ook ontstonden er bedrijfjes die CD-Roms verkochten met Linux erop, om een gemakkelijke installatie mogelijk te maken. Een van die bedrijfjes was het inmiddels beursgenoteerde Red Hat.

De belangrijkste vernieuwing die Torvalds toepaste was dat hij vaak (soms wel meerdere malen per dag) nieuwe versies van Linux ter download beschikbaar stelde, in plaats van lang te wachten voordat hij met een nieuwe versie van Linux uitkwam. Hierdoor konden anderen die ook Linux gebruikten op een snelle manier feedback geven. Zo wist hij een enorm netwerk van tienduizenden (sommige beweren dat dit zelfs is uitgegroeid tot honderdduizenden) computerprogrammeurs om zich heen te mobiliseren die allemaal gratis meewerkten aan het verbeteren van zijn programma.

Tegenwoordig is Linux op de servermarkt gegroeid tot een serieuze concurrent van Microsoft Windows en commerciële Unixversies. Naar schatting maken tientallen miljoenen mensen over de hele wereld gebruik van Linux. En is er een hele industrie ontstaan van bedrijfjes die zich bezig houden met het verkopen van Linuxdistributies (CD-Roms met daarop de Linux-kernel en andere handige programma's) of die service rondom de installatie of het onderhoud van Linux aanbieden. Linux als vrij en open besturingssysteem biedt tegenwoordig een volwaardig alternatief voor gesloten besturingssystemen zoals commerciële Unixvarianten of Microsoft Windows. Veel vrije software programmeurs

---

is als naam voor het complete besturingssysteem. Zie: http://www.gnu.org/.

gebruiken Linux als het platform om vrije software op te schrijven.

Eric von Hippel biedt een theoretisch kader dat zeer verhelderend werkt om in te kunnen zien, wat nu precies de revolutionaire vernieuwing van de ontwikkelpraktijk van Linux is.[37] Het interessante van de doe-het-zelf-Linuxgebruikers is dat zij een dynamisch kennisnetwerk vormen, waardoor het mogelijk is om zelfs de meest complexe taken, zoals het schrijven van een compleet besturingssysteem, voor elkaar te krijgen. Zo'n netwerk kan echter niet zomaar ontstaan, maar heeft daarvoor de volgende voorwaarden nodig:

> "User innovation networks can function entirely independently of manufacturers when
> (1) at least some users have sufficient incentive to innovate,
> (2) at least some users have an incentive to voluntarily reveal their innovations, and
> (3) diffusion of innovations by users is low cost and can compete with commercial production and distribution. (Hippel 2002b: 1)"

> "User innovation networks also may, but need not, incorporate the qualities of user 'communities' for participants, where these are defined as '... networks of interpersonal ties that provide sociability, support, information, a sense of belonging, and social identity (Hippel 2002b: 3)"

Wanneer we dit vertalen naar de computerwereld dan moeten gebruikers op de eerste plaats over een software gereedschapskist beschikken (iets dat standaard wel in Unix, maar niet in Macintosh of Windows computers is ingebouwd). Daarnaast moeten ze om programma's te kunnen verbeteren over de broncode kunnen beschikken, anders valt er niets te verbeteren. Als laatste hebben ze een internet aansluiting nodig die goedkope distributie van de aanpassingen mogelijk maakt en dienst doet als communicatiemiddel om een gemeenschap te kunnen vormen. In dit netwerk verandert de passieve gebruiker in een actieve gebruiker/doe-het-zelffabrikant:

> "... fully-functional innovation networks can be built up horizontally - with actors consisting only of innovation users (more precisely, "user/self-manufacturers"). Users participating in the network design and build innovative products for their own use - and also freely reveal their designs to others. Those others then replicate and improve the innovation that has been revealed and freely reveal their improvements in turn - or they may simply replicate the product that has been revealed and adopt it for their own, in-house use. (Hippel 2002b: 3)"

§ 4.7 Conclusie

In dit hoofdstuk heb ik laten zien welke alternatieven er bestaan voor de gesloten programma's uit de voorgaande hoofdstukken. Er is inmiddels een compleet arsenaal aan open gereedschap beschikbaar voor het ontwikkelen van software, zoals:

- meerdere vrije besturingssystemen (FreeDos, (GNU/)Linux, FreeBSD, OpenBSD, NetBSD)
- het programmeergereedschap uit het GNU-project
- meerdere vrije en 'open' licenties voor computerprogramma's

[37] Matt Ratto attendeerde mij op de relevantie van het werk van Von Hippel voor het bestuderen van open software. Ratto's promotiedissertatie over de geschiedenis van Linux verschijnt vanaf april 2003.

- het internet als communicatiemiddel, productiemiddel en distributiekanaal
- een grote internationale gemeenschap van doe-het-zelf computeraars die voor gratis en open ondersteuning zorg dragen.

Het is duidelijk dat deze ‘open’ en vrije software volwassen oplossingen biedt voor de problemen die gepaard gaan met het gebruik van gesloten software, zoals de vendor-lock-in, waardoor een ongezonde afhankelijkheidsrelatie tussen producenten en consumenten van software ontstaat; de Microsoft Monocultuur, met alle nare gevolgen zoals virussen en gesloten bestandsformaten, de teloorgang van culturele diversiteit op computerprogrammagebied, de discriminatie van gebruikers die weinig geld hebben, oude of niet-standaard computerhardware wensen te gebruiken.

Daarnaast biedt ‘open’ en vrije software de mogelijkheid tot een veel kritischer consument, die doelbewust keuzes kan maken of hij bepaalde software, wel of niet wil gebruiken en waarom. Zo kan hij voor elke taak de juiste software kiezen, soms gesloten, soms open, afhankelijk van wat er op dat moment het beste geschikt is om het probleem op te lossen. Daarmee krijgen computergebruikers eindelijk weer een keuze, die ze vanaf de ontwikkeling van de Personal Computer als wereldwijde standaard steeds meer zijn kwijtgeraakt.

Wat we echter nog niet hebben gezien is hoe de ontwikkeling van ‘open’ software er nu eigenlijk aan toe gaat. Hoe vrij zijn gebruikers en ontwikkelaars nu eigenlijk echt? En worden er ook bij ‘open’ software niet ook mensen uitgesloten, al was het maar omdat ze technisch niet capabel genoeg zijn? In mijn volgende hoofdstuk ga ik op microniveau de ontwikkel- en gebruikspraktijk rondom een populaire teksteditor, Vim genaamd, bekijken, om zo tot antwoorden op deze vragen te komen. Want ook al bestaan er standaarden voor open licenties voor het gebruik van een programma met zijn bijbehorende broncode, voor het ontwikkelingsmodel van open software bestaat nog geen standaard. Daardoor is de ontwikkeling van ieder open computerprogramma uniek, en niet altijd even open.

5. Empirisch Onderzoek: Vim

> "As a user I was pretty satisfied with [the open source program] Vim. The horror came when I discovered the source." (e-mailcorrespondentie met Philippe Fremy)

§ 5.1 Een korte geschiedenis van Vim

In theorie, zoals beschreven in het vorige hoofdstuk, lijkt 'open' software democratischer dan gesloten software. Gebruikers hebben immers direct inspraak bij de ontwikkeling en het gebruik van het door hen gebruikte computerprogramma. Pas in de praktijk blijkt of dit daadwerkelijk het geval is. Daarom behandel ik in dit hoofdstuk mijn empirische onderzoek naar de open source teksteditor Vim, om te kijken hoe vorm wordt gegeven aan deze openheid en waar de grenzen aan deze openheid liggen. De structuur van Vim en de Vimgemeenschap kan alleen verklaard worden in het licht van haar ontstaansgeschiedenis. Daarom vertel ik eerst kort iets over de voorganger van Vim, de teksteditor 'vi' die standaard in Unix aanwezig is. Daarna komt de geschiedenis van Vim zelf aan bod.

Het Unix besturingssysteem (van nature al een besturingssysteem dat de nadruk legt op openheid in de zin van open standaarden, open documentatie, open broncode, modulariteit, portabiliteit en interoperabiliteit) hanteert als een van haar basisprincipes dat alles een bestand is. De meeste van deze bestanden bestaan uit simpele teksttekens. Een e-mailbericht is een voorbeeld van zo'n tekstbestand. Maar ook de broncode voor een nieuw programma is een tekstbestand. Daarnaast kan via kleine tekstbestanden (configuratiebestanden) de werking van programma's ingesteld en aangepast worden.

Een Unix-computer bevat honderden verschillende tekstbestanden. Om iets aan deze tekstbestanden te kunnen veranderen (bijvoorbeeld om de werking van een bepaald programma te veranderen of gewoon om een e-mail te typen) heb je een zogenaamde teksteditor nodig. Omdat er op Unixsystemen veel met tekstbestanden wordt gewerkt, is een goed programma voor het bewerken van tekst een basisvereiste. Een Unixgebruiker werkt een groot deel van zijn tijd in een teksteditor.

In de begintijd van Unix waren voor dit doel slechts primitieve programma's beschikbaar, die vaak met maar één regel tekst tegelijk konden werken. Toen Bill Joy in 1976 het programma 'vi' (afkorting voor visual interface) schreef, betekende dit een ware revolutie

op teksteditinggebied.[38] De nieuwe teksteditor vi maakte optimaal gebruik van de nieuwe mogelijkheden van voor die tijd moderne terminalsystemen. Zo kon vi een heel scherm tekst tegelijk afbeelden en konden gebruikers de cursor snel heen en weer door het gehele tekstbestand verplaatsen om overal wijzigingen aan te brengen. Gebruikers vonden vi een stuk makkelijker te gebruiken dan de vorige generatie teksteditors en het programma werd al snel de standaard teksteditor voor Unixsystemen.

Niettemin was vi geen simpel programma, doordat de gebruiker een groot aantal toetscommando's uit zijn hoofd moest kennen, omdat een menusysteem ontbrak. Een voorbeeld van de vi-logica: de cursor bestuur je niet met de pijltjestoetsen (want nog lang niet alle computers in de jaren zeventig hadden zoiets als pijltjestoetsen op hun toetsenbord), maar met de toetsen H, J, K en L. Gebruikers die eenmaal met vi hadden leren werken, konden er zeer snel teksten mee bewerken en de meest ingewikkelde trucs uithalen - bijvoorbeeld met één commando alle regels in een bestand in omgekeerde volgorde plaatsen. De initiële tijdsinvestering die nodig was om goed met het programma te leren werken, werd later ruimschoots terugverdiend door de tijdswinst die bij het bewerken van bestanden te behalen viel. Uit populariteitspolls voor teksteditors blijkt dat op Unixsystemen vi (of modernere afgeleiden hiervan, zoals Vim) nog steeds op grote schaal gebruikt wordt.[39]

Bram Moolenaar was ook een programmeur die Vi dagelijks gebruikte bij zijn computerwerkzaamheden en het programma letterlijk in zijn vingers had zitten. Toen hij in 1988 een Amiga voor persoonlijk gebruik kocht, wilde hij dan ook graag een programma met dezelfde toetsencombinaties op zijn nieuwe computer kunnen gebruiken. Na een tijdje zoeken vond Moolenaar een aantal vi-klonen voor de Amiga, maar hij vond geen enkel programma dat aan al zijn eisen voldeed. Daarom besloot hij zijn eigen vi-kloon voor de Amiga te gaan schrijven.[40] Hij zou daarbij niet zomaar vanuit het niets beginnen, maar zoals dat in de Unix-traditie gebruikelijk is ("good programmers write good software, great programmers steal great software" (Gancarz 1995), zou hij zich hierbij op werk van anderen baseren.

Moolenaar zegt hier zelf over: "The vi clone that I started with was also open source. I would probably not have started working on vim without that starting point. Thus open source code being available was essential for me (Hemel 2001)." Moolenaar gebruikte als basis voor zijn Amiga-vi het programma 'Stevie' van Tony Andrews, dat voor een ander

[38] Bill Joy zou vooral bekend worden als de oprichter van computerbedrijf Sun Microsystems, dat onder andere de platformonafhankelijke programmeertaal Java heeft ontwikkeld.
[39] Op dit moment zijn onder programmeurs twee soorten teksteditors overduidelijk favoriet: GNU Emacs en al haar varianten, en vi en al haar moderne varianten, waaronder Vim.
[40] Raymond introduceert in zijn boek the Cathedral and the Bazar (1999) voor dit fenomeen binnen de open

computertype geschreven was, namelijk voor de Atari ST.

Dat Moolenaar dit kon doen, kwam omdat de auteur van Stevie dit uitdrukkelijk toestond, zoals valt te lezen in de documentatie die Andrews bij de broncode van zijn programma meeleverde: “STEVIE may be freely distributed. The source isn’t copyrighted or restricted in any way. If you pass the program along, please include all the documentation and, if practical, the source as well. I’m not fanatical about this, but I tried to make STEVIE fairly portable and I’d like to see as many people have access to the source as possible.”[41] In de ware traditie van het delen van informatie leverde Andrews zelfs een uitgebreide handleiding mee, waarmee anderen aan de slag konden gaan om Stevie geschikt te maken voor andere computerplatformen.

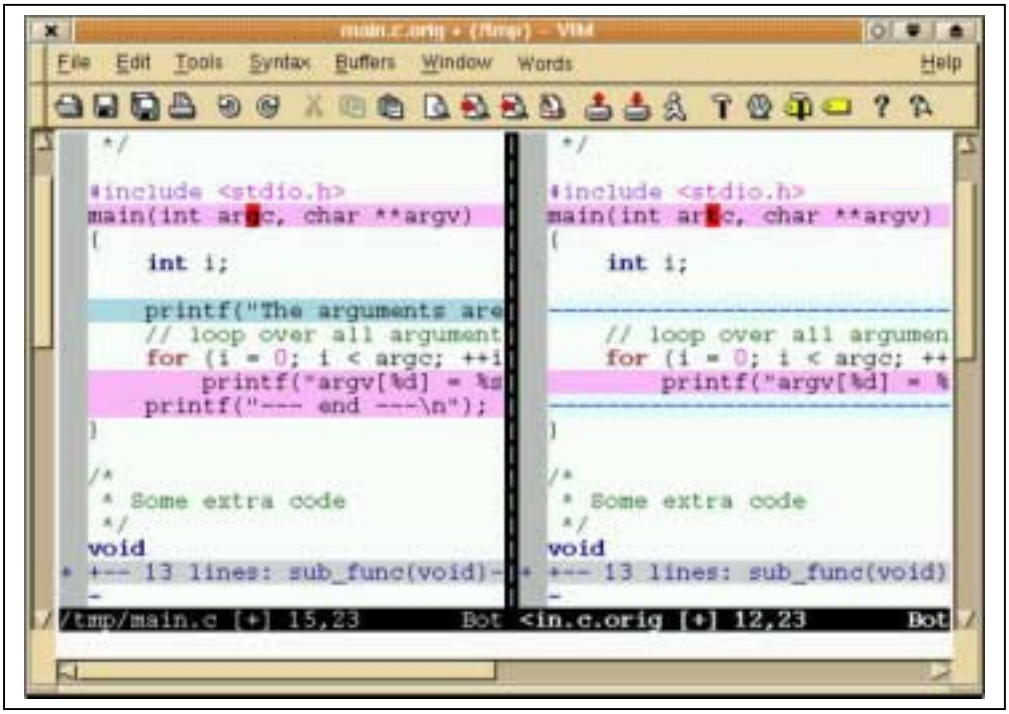
Toen Moolenaar na een groot aantal aanpassingen vond dat zijn programma goed genoeg was, besloot hij in 1991 zijn programma te verspreiden onder de naam Vim (‘Vi IMitation’) via een public domain diskettedistributie. Vanaf toen begonnen gebruikers ook aanvullingen op de broncode van het programma op te sturen. Ook waren er gebruikers die Vim geschikt maakten voor hun eigen computerplatform, zoals voor MS-Dos. Aangemoedigd door al deze enthousiaste gebruikers probeerde Moolenaar zijn programma steeds beter te maken en van meer functionaliteit te voorzien.

Nadat Vim in 1992 ook geschikt was gemaakt voor Unix veranderde Moolenaar de naam naar ‘Vi IMproved’, omdat Vim niet slechts een kopie van het originele Vi was, maar extra functionaliteit bood, die het oorspronkelijke programma niet bezat. Hiermee werd Vim een volwaardig alternatief voor het oorspronkelijke Vi. Dat mocht ook wel, want het oorspronkelijke Vi, al in 1976 geschreven, werd sinds 1985 niet meer actief verder ontwikkeld, en bood slechts mogelijkheden voor de Spartaans uitgeruste computers uit die tijd. In 2003 is Vim beland bij versienummer 6.1 en beschikt het over uitgebreide functionaliteit, zoals grafische menu’s, syntax highlighting voor bijna tweehonderd programmeertalen, ondersteuning voor niet-westerse talen (bijvoorbeeld Japans, Arabisch of Russisch) en vele andere voor programmeurs handige functies.

---

software wereld de term “scratching an itch”.

[41] Bron: stvi369g.zip, http://www.os2world.com/freeos2/editors.html#stevie



Illustratie: Beeldschermafdruk van de teksteditor Vim

Veel mensen die nog nooit met Vim hebben gewerkt, vinden het in eerste instantie vaak een vreemd programma. Ze snappen het nut niet van het besturen van de cursor met lettertoetsen in plaats van de pijltjestoetsen. En het gebruik van de escapetoets om te schakelen tussen het invoeren van tekst en het invoeren van commando's vinden ze vaak ook niet meer van deze tijd. Vim is inderdaad een programma dat gebaseerd is op de historische achtergrond waarin vi ontwikkeld werd, een tijd waarin computers nog niet zo geavanceerd waren als tegenwoordig. Niettemin is Vim toch een moderne teksteditor die op het gebied van het aanpassen van platte ASCII-teksten vaak veel meer kan dan met een tekstverwerkings-pakket zoals WordPerfect of Word mogelijk is. Misschien is het programma in eerste instantie niet zo makkelijk te gebruiken, maar als een gebruiker eenmaal gewend is aan alle toetsenbordcodes, kan hij zeer snel door teksten navigeren, en de meest ingewikkelde op reguliere expressies gebaseerde tekstvervangingstrucs uithalen, die in vele andere programma's niet mogelijk zijn.

Bovendien kan een gebruiker zijn eigen commandoreeksen opslaan onder zelf gedefinieerde toetsen. Gebruikers kunnen Vim aan bijna al hun wensen aanpassen, doordat ze 'plugins' en 'macro's' kunnen schrijven voor Vim. Ook kan een gebruiker hulp inschakelen van andere gebruikers op het Vimforum op internet, hulp zoeken in de Usenetgroep

comp.editors, gratis Howto-documenten van internet halen of zelfs een compleet boek als PDF-bestand downloaden waarin uitgebreid staat uitgelegd hoe Vim werkt. Het is dan ook belangrijk dat we een onderscheid maken tussen initiële leervriendelijkheid van een programma (zoals bijvoorbeeld het gebruik van grafische pictogrammen) en gebruiksvriendelijkheid (zoals mogelijkheden tot vergaande aanpassingen door gevorderde gebruikers) (Bruyninckx 2002). Bram Moolenaar heeft bij Vim duidelijk gekozen voor de gebruiker die complexe tekstveranderingen eenvoudig, snel en efficiënt wil kunnen uitvoeren. Niettemin is er ook gedacht aan de wensen van gebruikers die een betere initiële leervriendelijkheid willen in de vorm van een trainingsprogramma, een grafische interface en een modus waarin Vim op dezelfde wijze als Wordpad werkt (genaamd Vim-easy), zonder dat gebruik van de escapetoets nodig is.

Als Vim zo'n mooi programma is met zo veel 'gelukkige gebruikers', waarom vraagt Moolenaar dan geen geld voor zijn programma? In de eerste plaats baseerde Moolenaar Vim op het vrije computerprogramma Stevie. Het is dan ook niet meer dan fair dat Vim ook vrije software is, waar je de broncode bijkrijgt. En daar zit hem de crux. Want hoe zou je geld kunnen verdienen met iets dat je ook gratis van internet kan downloaden? Moolenaar zelf denkt niet dat nog veel mensen Vim zouden blijven gebruiken als ze er geld voor zouden moeten betalen. Er bestaan teveel goede alternatieven voor Vim die ook allemaal open source software zijn.

Bovendien is Moolenaar onbevangen begonnen aan de ontwikkeling van Vim. Hij schreef het in eerste instantie voor zichzelf, en besloot daarna zijn programmeerwerk te delen met andere computergebruikers. Aanvankelijk waren dat alleen Amiga-gebruikers, maar al snel waren dat ook gebruikers van andere computersystemen. Toen hij eind jaren tachtig zijn eigen Vi-kloon schreef had hij absoluut geen idee, dat dit programma vijftien jaar later honderdduizenden gebruikers zou hebben. Moolenaar wilde gewoon een mooi programma schrijven.

Door deze aanpak kon Moolenaar al zijn tijd besteden aan het schrijven van programmacode voor Vim en het verwerken van commentaar, tips, suggesties, foutmeldingen en verzoeken voor nieuwe functies van andere Vim-gebruikers. Zelf zegt Moolenaar hierover: "Wanneer je een computerprogramma maakt dat je wil gaan verkopen, moet je eerst een heleboel andere dingen doen voor je kunt beginnen met het schrijven van broncode. Je moet een marktonderzoek doen om te kijken of het überhaupt mogelijk is om een programma winstgevend in de markt te zetten. Zeker wanneer er een heleboel concurrerende software gratis van internet te downloaden is. Bovendien moet je je eigen bedrijf beginnen en je moet

support aan gebruikers en bedrijven kunnen leveren. Het levert een heleboel extra overheadkosten op, die ik niet heb, wanneer ik geen geld voor mijn programma vraag."[42]

Vim verkopen 'kost' Moolenaar meer, dan het gratis weggeven hem oplevert. Doordat Vim vrije software is, hoeft Moolenaar zich alleen bezig te houden met schrijven van code. Daarnaast verzorgen de gebruikers hun eigen support en dragen ze zelf bij aan de ontwikkeling van nieuwere, betere en uitgebreidere Vim-versies. Ook heeft Moolenaar dankzij Vim een grote naamsbekendheid verworven in de open source wereld. Deze reputatie heeft hem misschien wel aan zijn huidige betaalde baan bij Stichting NLnet Labs geholpen, voor wie hij het programma A-A-P schrijft.[43] Al wil Moolenaar zelf geen causaal verband leggen, hij sluit niet uit dat het misschien heeft meegespeeld.

§ 5.2 Hoe open is het gebruik van Vim?

Ik leg de nadruk op 'open' computerprogramma's die een democratisch gebruik faciliteren. De vraag die ik in dit verband wil beantwoorden, is in hoeverre Vim nu eigenlijk als een open programma beschouwd kan worden. Hierbij zal ik de nadruk in deze paragraaf vooral leggen op de gebruikspraktijk van Vim, waarbij gebruikerstoegang tot de broncode van Vim niet noodzakelijk is. In de volgende paragraaf ga ik dieper op de openheid van het netwerk voor Vim-ontwikkelaars, voor wie toegang tot de Vimbroncode een eerste voorwaarde is.[44]

In de eerste plaats werkt Vim volledig met open standaarden. Voor tekst is vooral ASCII (American Standard Code for Information Interchange) van belang. Helaas verschilt ASCII per computerplatform en hanteren Microsoft Windows, MacOS en Unix ieder haar eigen ASCII. Vim herkent deze formaten echter automatisch en kan er gewoon mee werken. Daarnaast kan Vim ook omgaan met de opvolger van ASCII, namelijk Unicode. Hierdoor kan Vim ook tekst bewerken in talen als Japans, Hebreeuws, Chinees, Russisch, Grieks of Ethiopisch. Er wordt nog steeds hard verder gewerkt aan een nog betere Unicode-ondersteuning door Vim; hiervoor bestaat een aparte Vim-e-maillijst die geheel aan dit thema is gewijd. Omdat alle tekstbestanden die in Vim gewijzigd of gemaakt worden, ook in de open standaarden ASCII of Unicode worden opgeslagen, kunnen andere programma's op andere computersystemen deze teksten vervolgens inlezen en wijzigen. Ook in de toekomst zullen besturingssystemen deze standaarden waarschijnlijk nog ondersteunen, zodat daarmee een foutloze werking in de toekomst is verzekerd.

[42] Interview met Bram Moolenaar op 02-07-2002.
[43] Voor meer informatie zie: Mallett 2002, en http://www.a-a-p.org/
[44] De algemene conclusies in deze en volgende paragrafen zijn een gecondenseerde vorm van de resultaten van

Ten tweede is Vim een voorbeeld van een programma met een grote interoperabiliteit, het kan namelijk met een groot aantal programma's samenwerken en informatie uitwisselen. Door gebruik te maken van standaardtekstbestanden kan Vim op makkelijke wijze informatie uitwisselen met andere programma's en computersystemen.

Ten derde biedt Vim een hoge portabiliteit. De broncode van Vim is vrij beschikbaar op het internet. Daardoor konden gebruikers die Vim een goed programma vonden, het vertalen naar andere computerplatformen waarvoor Vim nog niet beschikbaar was. Hoewel Vim oorspronkelijk voor de Amiga werd geschreven, is het programma inmiddels voor verschillende soorten Unix beschikbaar, voor verschillende Windowsversies, voor MS-DOS, voor OS/2, voor AmigaOS en voor verschillende versies van MacOS. Daarnaast bestaan er zowel versies van Vim voor grafische omgevingen (Graphical User Interface) als tekstversies (Command Line Interface). Doordat Vim voor een groot aantal verschillende besturingssystemen en verschillende soorten computers beschikbaar is, is het een zeer portabel programma, dat oudere, of minder voorkomende hard- en software niet discrimineert.

Ten vierde biedt Vim op het niveau van de gebruikersinterface ook een hoge interoperabiliteit. Gebruikers die gewend zijn om met het originele Vi te werken, kunnen zonder probleem overstappen op Vim. Hiervoor beschikt Vim over een speciale zogenaamde compatibiliteitsmodus, waarin Vim precies hetzelfde werkt als het originele programma Vi (daarmee voldoet Vim aan de Posix-norm van de Amerikaanse overheid die de interface van Unix-programma's omschrijft). Daarnaast kunnen gebruikers van andere Vi-klonen ook makkelijk overstappen naar Vim, doordat ook nu de toetscommando's waarmee de werking van Vim bestuurd wordt, bijna volledig gelijk zijn.

Ten vijfde beidt Vim een zeer hoge mate van configurabiliteit. Vim is namelijk in grote mate te configureren door de gebruiker: van kleur tot toetscommando's, bijna alles kan de gebruiker zelf instellen. Behalve dat er al veel voorgeprogrammeerde Vim-schema's, macro's en plugins worden meegeleverd, kan de gebruiker zelf aan de slag gaan om de werking van Vim naar eigen behoefte aan te passen met de in Vim ingebouwde scriptingtaal. Deze aanpassingen kunnen vervolgens via internet weer worden uitgewisseld met andere Vimgebruikers. De centrale plek voor de uitwisseling van al deze aanpassingen is de Vim.online website. Voor het publiceren van aanpassingen aan Vim door middel van scripts, macro's of plugins is overigens van niemand toestemming nodig.

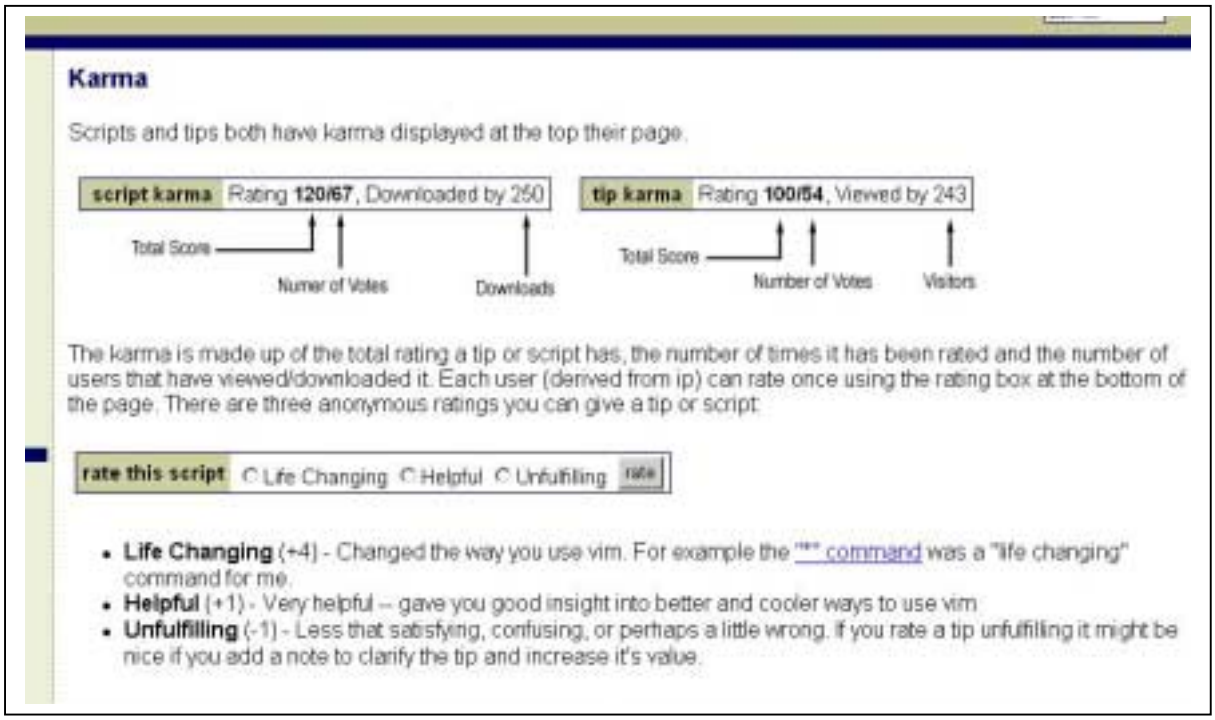
---

mijn virtueel focusgroeponderzoek.

Een metafoor die de levendige Vimgemeenschap goed omschrijft, is die van een emergente structuur, zoals een mierennest dat is. De wetenschapsjournalist Steven Johnson beschrijft in zijn boek Emergence helder hoe emergente structuren schijnbaar spontaan kunnen ontstaan en wat hun kracht is (Johnson 2002). Een centraal gegeven van emergente systemen is dat uit simpele onderdelen, die slechts een paar simpele regels volgen, een ingewikkelde structuur kan ontstaan van verbazingwekkende complexiteit en wonderbaarlijke meerwaarde die het niveau van de som der delen ver overstijgt. Bij de keuze van zijn voorbeelden beperkt Johnsen zich vooral tot mieren, hersenen, steden en gemeenschappen voor het ontwikkelen van computerprogramma's.

Zo beschrijft Johnsen in zijn boek als voorbeeld de online gemeenschap rondom de website Slashdot.org (met het motto "News for Nerds, Stuff that Matters"), waar door middel van webforums gediscussieerd wordt over het laatste nieuws op het gebied van internet, technologie en computerprogrammatuur. In principe zijn deze discussies, waarbij er per nieuwsitem vaak honderden reacties van verschillende lezers zijn, een groot ongeordend geheel vol waardevolle bijdragen, maar ook vol onzin. Door een gedeelte van de lezers van de website deze bijdragen echter een cijfer van -1 tot 5 te laten geven, kunnen andere lezers een 'quality treshold kiezen', waardoor alle slecht gewaardeerde bijdragen weggefilterd worden, en alleen de waardevolle overblijven.

Van een dergelijk systeem is ook sprake bij de website van Vim. Hier kunnen gebruikers zelfgemaakte aanvullingen voor Vim zoals, macro's, scripts en plugins of tips voor het gebruik aanbieden aan andere gebruikers. Het door de Vimwebsite gebruikte systeem biedt hierbij de mogelijkheid om de enorme berg aan informatie vervolgens door middel van verschillende kriteria te structureren. Zo kunnen tips bijvoorbeeld geselecteerd worden op eigenschappen als nieuwheid, aantal keren dat ze geraadpleegd zijn of waardering door lezers van de Vimwebsite. Een ordenend principe hiervoor is het zogenaamde 'karma' dat door gebruikers van de Vimwebsite aan tips en scripts kan worden toegekend.



Illustratie van het Karma-systeem, zoals dat wordt gebruikt op Vim online (www.vim.org)

Alhoewel ik eerder verdedigd heb dat open ondersteuning geen fundamenteel kenmerk is van een open computerprogramma, ik behandel het hier wel. De hoge mate van onderlinge hulp door gebruikers aan elkaar is namelijk een van de belangrijkste factoren voor het succes van Vim.

Veel mensen zien ondersteuning van open broncode/vrije software als een groot probleem. Als er iets misgaat of als het programma überhaupt niet blijkt te werken, wie helpt je dan uit de brand? Als je een softwarepakket in de winkel koopt, kun je tenminste terug naar de winkel, of naar de leverancier. Niet dat die er iets aan kunnen doen om je probleem op structurele wijze op te lossen, doordat ze niet over de broncode beschikken. Daarnaast dekken de grote softwarebedrijven, zoals bijvoorbeeld Microsoft, die wel iets aan deze problemen kunnen doen, doordat zij wel over de broncode beschikken, zichzelf grondig in tegen fouten in hun software en zorgen ze ervoor dat ze niet aansprakelijk voor eventuele schade gesteld kunnen worden. Niettemin vertrouwen veel mensen software die uit een winkel komt nog steeds meer dan software die ze gratis van internet kunnen halen. Want in dat laatste geval, lijkt het alsof er geen hulp bij problemen is. Niemand biedt immers een garantie op oplossingen.

Nu wordt er bij het gemiddelde vrije computerprogramma inderdaad geen

helpdesknummer meegeleverd dat je kunt bellen als je problemen hebt. En omdat je het programma niet gekocht hebt, kun je inderdaad ook niet terug naar de winkelier bij problemen. Maar dat wil niet zeggen dat er geen ondersteuning is. In de eerste plaats kun je de auteur van het programma e-mailen wanneer er iets niet werkt of je een vraag hebt. Daarnaast bestaan er meestal gebruikersgroepen rondom een bepaald programma, die je per e-mail kunt vragen of ze de oplossing voor jouw probleem kennen. Wanneer je zelf over de nodige programmeerkennis beschikt kun je in de broncode kijken wat er aan de hand is. Omdat veel mensen echter niet over deze kennis beschikken, maken niet al te veel mensen gebruik van deze optie. Op de laatste plaats kun je als je daarvoor het geld hebt, altijd nog een onafhankelijke software ingenieur inhuren, die voor jou naar de broncode kan kijken, om zo het probleem op te lossen. Zo kunnen we dan ook stellen dat ondersteuning van vrije/open broncode software (of het ontbreken daarvan) vooral een gepercipieerd probleem is, omdat er wel degelijk sprake is van hulp bij problemen, alleen niet op de traditionele manier.

Wanneer we kijken naar de ondersteuning van Vim, valt in de eerste plaats op dat Moolenaar bij het programma een zeer uitgebreide documentatie meelevert in de vorm van tekstbestanden, waarin de werking van Vim zeer uitvoerig staat beschreven. Deze documentatie bestaat uit meer dan honderd tekstbestanden (bijna 3,5 megabyte aan platte tekst). Je kan dan ook stellen dat je bij Vim gratis een compleet gebruikershandboek meegeleverd krijgt.

Daarnaast kunnen gebruikers die problemen met Vim ondervinden op zoek gaan op internet naar documentatie over de werking van hun programma en eventuele tekortkomingen daarin. Op de website www.vim.org is een grote verscheidenheid aan online en gratis documentatie te vinden. Zo is er een lijst met veel gestelde vragen (FAQ), er is een archief met tips van medegebruikers dat op verschillende manieren gesorteerd op te vragen is, er zijn verwijzingen naar allerlei online documentie over Vim op andere internetpagina's en er is zelfs een verwijzing naar een pagina waar je een volledig boek van meer dan 570 pagina's over Vim, dat ook in de winkel verkrijgbaar is, als PDF kunt downloaden zodat je het zelf kunt uitprinten.

Wanneer er dan nog steeds iets is dat niet werkt, kan een gebruiker direct hulp zoeken bij andere Vimgebruikers. Op Usenet is er een speciale nieuwsgroep volledig gewijd aan teksteditors, namelijk comp.editors. Officieel komen in deze nieuwsgroep vragen over allerlei verschillende editors aan de orde, maar een groot deel van de discussies gaat over Vim.

Naast de Vimwebsite is ook nog de Vim-e-maillijst, waar algemene vragen aan de orde kunnen komen. Naast deze algemene e-maillijst bestaan er meer gespecialiseerde e-

maillijsten voor mensen die Vim op een Macintosh gebruiken, voor aankondigingen en voor Vimprogrammeurs. De voertaal op deze e-maillijsten is Engels, daarom is voor de mensen die geen Engels kunnen ook nog een speciale e-maillijst in het Frans.

De hoofdauteur Bram Moolenaar leest de e-maillijsten ook door. Zo blijft hij in de gaten houden waar gebruikers problemen mee hebben en wat er nog steeds niet (goed genoeg) werkt. Moolenaar beantwoordt dagelijks tientallen e-mails van mensen met vragen of problemen over zijn programma. Hierbij laat hij de meer simpele vragen over aan andere Vimgebruikers om te beantwoorden.

Mocht e-mail ook geen uitkomst kunnen bieden, dan is het ook nog altijd mogelijk om informatie te vragen op een van de twee speciale IRC-kanalen (Internet Relay Chat) die speciaal aan het gebruik van Vim gewijd zijn.

Kortom, er zijn genoeg manier om aan goede informatie en documentatie over Vim te komen, zowel in statische vorm (van websites, howto's en faq's tot complete boeken) als in meer dynamische vorm (door directe interactie met medegebruikers of de hoofdauteur via e-mail). Hierbij is zowel informatie voor de beginner als voor de expert beschikbaar. De enige voorwaarde hiervoor is toegang tot internet. Maar ook mensen zonder internettoegang kunnen al heel erg veel informatie vinden in de bij het programma zelf meegeleverde documentatie.

§ 5.3 Hoe open is de ontwikkeling van Vim?

In de vorige paragraaf hebben we kunnen zien dat Vim in velerlei opzichten een zeer open programma is. Maar wanneer gebruikers op fundamentele wijze iets aan Vim willen kunnen veranderen dan is daartoe een open toegang tot de broncode van Vim noodzakelijk. In deze paragraaf analyseer ik de openheid van de broncode van Vim.

Iedereen die dat wil kan de Vim broncode vrij van internet downloaden. Hieronder volgt een fragment uit een van de vele broncodebestanden van Vim.[45]

```
/* vi:set ts=8 sts=4 sw=4:
 *
 * VIM - Vi IMproved    by Bram Moolenaar
 *
 * Do ":help uganda"  in Vim to read copying and usage conditions.
 * Do ":help credits" in Vim to see a list of people who contributed.
 */
```

Elk broncodebestand begint met deze bovenstaande regels. Deze regels tekst blijken bepalend voor de openheid van Vim:

Ten eerste zegt "by Bram Moolenaar" dat Bram Moolenaar de hoofdauteur is van Vim.

[45] Bron: 'vim.h' broncode bestand, ftp://ftp.vim.org/pub/editors/vim/unix/vim-6.1-src1.tar.gz

Ook al hoeft hij geen geld voor zijn programma, hij wil wel de erkenning van het hoofdauteurschap van Vim.

Ten tweede is Vim geen public domain software, maar valt het onder een gebruikerslicentie. Het feit dat iedereen de broncode van Vim kan downloaden van internet, wil niet zeggen dat iedereen vervolgens volledig vrij is om daar naar eigen goeddunken mee om te gaan.

Ten derde bedankt Moolenaar de mensen die mee hebben geholpen aan de ontwikkeling van Vim, door bijvoorbeeld het opsturen van foutmeldingen of suggesties voor veranderingen in de broncode, het beantwoorden van vragen op de emaillijst, het beheren van de Vimwebsite of het schrijven van documentatie voor Vim.

Hiermee is feitelijk het kader geschapen waarbinnen de vrijheid tot de omgang met de broncode en daarmee ook het gebruik van Vim wordt gereguleerd. Om te kunnen begrijpen hoe de Vimgemeenschap georganiseerd is, is het nodig de werking van de Vim-licentie te begrijpen.

Wanneer we openheid van software definiëren in termen van een ‘open’ licentie, dan is Vim een open programma: je krijgt de broncode er gratis bij, en je mag er aan veranderen wat je wil. Moolenaar definieert zijn programma Vim zelf trouwens niet als open source software, maar als ‘charityware’. Charityware zou je kunnen beschouwen als een element uit de grotere verzameling vrije software, want Vim voldoet aan alle eisen op dit gebied, zoals beschikbaarheid van broncode, vrijheid van gebruik, vrijheid van aanbrengen van veranderingen en vrijheid van het maken van kopieën en het distribueren daarvan.

Maar waar je open source software meestal gratis mag gebruiken, vraagt Moolenaar Vimgebruiker een kleine geldelijke bijdrage over te maken naar een goed doel, namelijk een stichting die een project in Uganda steunt voor kinderen wiens ouders aan Aids zijn overleden. De auteur stelt het volgende in de documentatie van zijn programma: “Vim is charityware. You can use and copy it as much as you like, but you are encouraged to make a donation for needy children in Uganda. (...) If you are happy with Vim, please express that by reading the rest of this file and consider helping needy children in Uganda.” Waarna een uitgebreide handleiding volgt over hoe vanuit verschillende landen op verschillende wijze geld overgemaakt kan worden naar de stichting ICCF Holland.

Vim gebruikt niet de veel gebruikte GNU Public License (GPL). Dit is een bewuste keuze van Moolenaar, omdat hij ook bedrijven de kans wilde gunnen, om hun eigen veranderingen in Vim aan te kunnen brengen zonder dat ze verplicht zijn om de gewijzigde versie van Vim inclusief broncode te distribueren. Dit is bij software die onder de GPL valt

niet mogelijk. Bram Moolenaar: “Ik vind het onzin dat bedrijven die Vim aanpassen zodat het met een specifiek commercieel pakket kan omgaan, verplicht zouden zijn om deze aanpassingen vrij te geven. (...) Ik vind het belangrijk dat Vim in zo veel mogelijk situaties gebruikt kan worden. De Vimlicentie is wat dat betreft vrijer dan de GPL.” [46]

Niettemin voldoet Vim aan alle eisen zoals de Free Software Foundation of het Open Source Initiative die stellen aan open software licenties. Zowel vrij kopiëren, vrij distribueren, als vrij veranderen van zowel de broncode als het gecompileerde programma is toegestaan, mits het resultaat onder dezelfde licentie valt. Vim mag onder de voorwaarden van de Vimlicentie vrij gebruikt, gekopieerd, aangepast, bestudeerd en gedistribueerd worden, waardoor Vim zoveel mogelijk mensen insluit in plaats van uitsluit. De Vimlicentie maakt gebruik in en aanpassing van Vim door bedrijven mogelijk, in tegenstelling tot de restrictievere GPL.

Het feit dat gebruikers bij Vim de broncode erbij krijgen, betekent dat ze zelf veranderingen hier in aan kunnen brengen. Zolang je alleen zelf die veranderingen wil gebruiken is er niets aan de hand. Zodra je echter deze aanpassingen met de rest van de wereld wil delen, ontstaat een complexere situatie. Want hoe voorkom je dat er honderden verschillende Vim-versies over het internet gaan zweven, die misschien onderling ook nog eens incompatibel zijn?

In principe wordt Vim door één hoofdauteur geschreven: “Bram Moolenaar is the main author of Vim. He writes the core Vim functionality and selects what code submitted by many others is included (Moolenaar 2000).” Dat is heldere taal. Deze uitsluiting van inclusie in het centrum van de Vim-ontwikkeling, wordt niet alleen juridisch door de copyrightbescherming vormgegeven, maar ook technisch. Doordat er geen open toegang is tot een centrale plek waar de meest actuele versie van de broncode van Vim bewaard wordt, maar deze broncode exclusief op de computer van Moolenaar aanwezig is, sluit hij actief anderen uit tot de distributie van de ‘officiële’ versie van de broncode.[47] Wil je broncode, dan ben je door deze socio-technische vormgevingsconstructie altijd verplicht deze via Moolenaar te verkrijgen.

Dat wil echter niet zeggen dat andere Vimgebruikers geen bijdrage kunnen leveren aan de ontwikkeling van Vim. Zij kunnen altijd foutmeldingen of verzoeken voor nieuwe functionaliteit (inclusief eventuele programmacode) opsturen naar Bram Moolenaar. Deze

[46] Interview met Bram Moolenaar, 02-07-2002.
[47] Voorbeelden van technische systemen die het samenwerken aan de ontwikkeling van broncode mogelijk maken zijn CVS en BitKeeper.

gebruikersinvoer is zelfs essentieel voor de ontwikkeling van Vim: “Being open-source is essential for Vim, otherwise it would be impossible for many people to work together on improvements and making ports to other operating systems (ibid).”

Daarnaast is er veel communicatie tussen de ontwikkelaar van Vim en de gebruikers. Zo houdt Moolenaar zich actief bezig met wat gebruikers graag aan nieuwe functionaliteit in zijn programma zouden willen zien:

> “Users ask questions on the Vim mailing lists, which indicates to me what the most common problems are. Sometimes people send me patches or requests for new features. I think that this co-operation of users and co-developers is the main strength of how Vim is being developed. Users and developers communicate directly with each other and with an open spirit. This is how open source software can grow to become better than commercial software (Moolenaar 2002).”

Vanuit deze gedachte organiseerde Moolenaar zelfs een soort gebruikersonderzoek op de Vim-mailinglijst om de belangrijkste wensen van Vimgebruikers voor een nieuwe Vimversie te inventariseren. Zo valt in de ingebouwde documentatie van Vim te lezen:

> “In November 1998 an inquiry was held to allow Vim users to vote for changes to Vim. Each person was allowed to give 10 positive and 5 negative points to a list of items. Below is the result. This indicates the desire of users for certain changes. This will be taken into account when deciding what to do next from the huge list above. (bron: Ingebouwde help functie in Vim 6.1)”

Hierbij viel tijdens mijn onderzoek op, dat Bram Moolenaar ontzettend snel reageerde op e-mails van gebruikers op de algemene Vim e-maillijst, wanneer zij denken dat er een fout in Vim zit. Ik denk niet dat commerciële producenten van computerprogramma’s zo snel en adequaat reageren op foutmeldingen van gebruikers van hun programma. Daarnaast bestaat er in de commerciële softwarewereld niet zo’n directe communicatielijn tussen de hoofdontwikkelaar van een compupterprogramma en de uiteindelijke gebruikers ervan. Hierdoor krijgt de schrijver van het computerprogramma een heel duidelijk beeld van waar gebruikers Vim voor gebruiken, hoe ze dat vervolgens doen, en wat daar bij misgaat.

Feitelijk komt het ontwikkelingsproces van Vim neer op een soort participatief ontwerpproces, waarbij gebruikers uitgebreid betrokken worden bij de ontwikkeling van een technologie. Hierdoor is Vim niet slechts een computerprogramma om teksten mee te bewerken, maar maakt het deel uit van een soort “Vimcultuur”, waarbinnen gebruikers elkaar helpen door het uitwisselen van kennis en waar binnen problemen op een bepaald soort manier worden opgelost. Je zou misschien wel kunnen spreken van een technologisch frame, waarbinnen nieuwe gebruikers gesocialiseerd worden (“The ‘vim’ way of doing things”).[48] Op deze manier kan het gebruik van Vim een bepaalde manier van omgang met computers uitdrukken. Deze aparte sfeer heeft waarschijnlijk ook te maken met het feit dat geld geen enkele rol speelt bij de ontwikkeling, het gebruik en de distributie van Vim.

[48] Analoog aan “the ‘Linux’ way of doing things”, zie: Ratto 2000

Hoe zit het echter met nieuwe Vim-code die niet door Moolenaar wordt geschreven? Hoe kan die toch in de 'officiële Vim-versie terecht komen? In feite beheert Bram Moolenaar in zijn eentje de broncode van Vim. Bram Moolenaar is het 'obligitary point of passage' om veranderingen in de broncode door te kunnen voeren. Soms neemt Moolenaar direct suggesties voor veranderingen in de broncode over, soms echter ook niet. Hieronder volgt een voorbeeld van een verzoek tot een grondige aanpassing van de Vim-broncode. Deze discussie is afkomstig van de Vim Developers e-maillijst (vim-dev@vim.org) en laat zien hoe er op open wijze wordt gediscussieerd over de ontwikkeling van vim. Deze discussie begon met een uitgebreide e-mail die op 5 november 2002 werd gepost, en zou uiteindelijk tot 18 november 2002 duren en bijna honderd openbare e-mails op de VimDevlist omspannen.[49] Niet alleen technische details spelen een rol in de discussie, maar ook het ontwikkelingsmodel van vim zelf, de 'identiteit' van Vim en hoe leden van de vimlijst met elkaar omgaan. De discussie begint met de volgende e-mail:

```
From: Philippe Fremy
Date: Tue Nov 5, 2002  8.40 am
Subject: Vim's future?

Hi all,
[...]
I think current vim's development is going on the wrong path. One stated goal of
vim is to not become like emacs: vim doesn't want to include every possible feature
one could dream of. But currently, vim is doing that. I see more and more code
added to vim, which, in my opinion, should not belong to vim itself.
[...]
Honestly, I am thinking about starting a vi clone for KDE, because it would be
easier to redevelop a vim from scratch using QT than to modify the current vim to
fit into what we need. However, I really prefer relying on a stable established
reliable codebase, from the vim project itself.
[...]
I would summarize the current vim like that:
- old codebase
- stable
- monolithic
- terminal oriented
The reason why vim has a monolithic old terminal-oriented codebase is quite
obvious: this is the way vim was born. But I think it is important to now move away
from this architecture to something more modern and versatile.
[...]
I know this requires a huge change in the way vim is currently architectured and
developed. But I think it is really worth it. It would allow a big code clean-up
and separation. The result will be wider use of vim, easier way to fix bugs, easier
way to get into vim development.
[...]
So what do you think?
```

In deze e-mail worden radicale dingen gezegd. In de eerste plaats stelt de auteur Fremy dat de ontwikkeling van Vim de verkeerde kant opgaat. Daarnaast stelt hij de identiteit van Vim ter discussie, het zou namelijk een te complex programma aan het worden zijn. Als oplossing

[49] Bron van alle e-mails: VimDevlist archief http://groups.yahoo.com/group/vimdev/

stelt hij zelf voor om Vim te ‘forken’ of om zijn eigen vi-kloon te beginnen. Vooral het semi-dreigement om met een ‘fork’ te komen is iets dat meestal niet lichthartig wordt opgenomen in de open software gemeenschap.

Een paar uur later al mengt Bram Moolenaar zich in de discussie:

```
From:  Bram Moolenaar <Bram@m...>
Date:  Tue Nov 5, 2002  11:11 am
Subject:  Re: Vim's future ?

[...]
This has been asked for many times. My reply has always been the same:
Vim is an editor. If you want to do more than edit text, use a
framework in which Vim is the editor and other tools can be plugged in
to do the other tasks.

Unfortunately, nobody has made this.
[...]
If you need this framework, please give a hand in making it!
[...]
```

In dit antwoord geeft Bram Moolenaar toe, dat wat Fremy wil er op dit moment nog niet is. Daarom nodigt Moolenaar Fremy uit om mee te helpen aan het schrijven van een dergelijk ‘framework’ waarmee Moolenaar inmiddels onder de naam A-A-P zelf gestart is. Vanuit Moolenaars perspectief is het goed voor te stellen dat hij niet veel zin heeft om veel tijd te stoppen in het grondig veranderen van vim. Zeker nu hij veel tijd nodig heeft voor zijn nieuwe open source project: A-A-P. Daarnaast biedt dit nieuwe project volgens Moolenaar precies wat Philippe Fremy nodig heeft.

Vervolgens volgt er een uitgebreide discussie, waarin Moolenaar steeds de rol op zich neemt van iemand op kritische wijze zijn eigen programma verdedigt en niet met vage ideeën geconfronteerd wil worden maar met concrete voorstellen. Een ander belangrijk punt dat wordt gemaakt, is dat een verandering van de code technisch gezien goed zou zijn, maar dat Moolenaar het niet gaat doen, want die heeft de tijd er niet voor. In de voorlaatste e-mail die Bram Moolenaar aan deze discussie wijdt vat hij zijn mening over de kwestie als volgt samen:

```
From:  Bram Moolenaar <Bram@m...>
Date:  Wed Nov 13, 2002  9:57 pm
Subject:  RE: [kvim-dev] RE: Vim's future ?

[...]
I have not seen a design or any other indication that the vague ideas
that have been mentioned would lead to a useful improvement. So there
is nothing for me to accept or turn down. I can only say that I'm not
going to work on these things, since I don't have the time available.

Considering the impact of the proposed changes, and the current
stability of Vim, I would only include something when I have confidence
that the existing functionality and quality of Vim will not suffer.
This means the new setup would have to be implemented and tested before
I would include it in the main code tree. An unstable branch could
exist for a while before we get there.
```

```
Since Vim is open source you don't need my permission to branch off and
implement all the nice ideas that you have. Only for folding the branch
back into the main code you would have to convince me.
[...]
I suggest you do a bit of architectural design and show us the results,
so that we at least know what your are talking about.
[...]
```

Bram Moolenaar bevestigt hierbij zijn identiteit als dominante ontwikkelaar van Vim. Maar tegelijkertijd laat hij de mogelijkheid open voor anderen om veranderingen aan te brengen in Vim, of zelfs een eigen Vim te maken. Maar dat zullen ze dan wel zelf moeten doen. Ook biedt hij de mogelijkheid om deze veranderingen later misschien ook door te voeren in de officiële distributie van Vim, maar de enige die daar uiteindelijk over beslist is Bram Moolenaar zelf.

Nu blijft echter de vraag over wat deze groep opstandige ontwikkelaars (georganiseerd onder de naam KVim) besluiten te gaan doen. Gaan ze zelf een nieuwe Vim schrijven? Fremy die de hele discussie begon, zegt hier zelf het volgende over:

```
Date: Thu, 12 Dec 2002 18:57:59 +0100
From: Philippe FREMY <P.FREMY@O...>
To: Stefan Verhaegh <sjs.verhaegh@s...>
Subject: RE: [kvim-dev] about kvim folding back into vim?

[...]
At the end of the discussion, I knew what I was looking for, which was:
1. Bram is not going to change vim the way I would like it to change
2. I can propose the changes myself but if they are not supported by a very
good reason, they won't be integrated into vim.
3. I don't have time to implement myself the changes I suggested, so the
whole discussion is void.
[...]
I probably did not take the right approach to expose
my view to the vim list, I was a bit too aggressive :-) But I was frustrated after
a few years of working with the big mess that is vim. I wanted to tell my point and
see how people would react.
[...]
The real question is can we change vim ? The answer is no.
[...]
Vim is certainly an interesting example. And we are confronted to a typical
problem: a program has been developed, the main developer does not want to
change it and we need a lot of changes to move forward. In all the situation that I
have seen like this, the answer was either forking or a new project from scratch.
[...]
The problem with Bram specifically is that [...] he is very
conservative. He pretends he is open to changes but he only means you can
send him changes and he may apply them if they are stable and are like what
he wants. Linus Torvalds is also exactly like that. This is not always good
and what is relevant for a core part of a system (the kernel) is not
necessarily relevant for an editor.

Even the way Bram develops vim is conservative. [...] The way he does it also means
that you will never be granted the right to modify vim. You only have he right to
send him patches and he'll apply them when he has time. This makes a big difference
for cooperation.

[...] he is alone coding, and I see why.
```

Philippe (in persoonlijke e-mailcorrespondentie met auteur) zegt dus in feite in deze e-mail dat het ontwikkelmodel van Vim helemaal niet zo open is. Ondanks dat je naar de code mag

kijken, en dat je er zelf veranderingen in aan mag brengen, is het niet zo dat deze veranderingen ook zo maar opgenomen worden in de officiële vim-distributie van Bram Moolenaar. Fremy is erg teleurgesteld dat hij niet mee mag werken aan een nieuwe, radicaal vernieuwde versie van Vim. De vraag is echter in hoeverre dit klopt. Zo schreef Moolenaar al dat Fremy volledig vrij is om eerst werkende code te schrijven om zijn ideeën concreet toe te lichten. Moolenaar sluit niemand uit van het meedoen aan de verdere ontwikkeling van Vim, maar daaraan zijn echter wel regels verbonden, die in de volgende email helder worden samengevat:

```
Date: Sat, 14 Dec 2002 12:09:37 GMT
From: Vince Negri <vince@b...>
To: Stefan Verhaegh <sjs.verhaegh@s...>
Subject: Re: [kvim-dev] about kvim folding back into vim?

[...]
The kvim developers are, as you've noticed, not happy with Bram's
"management style", shall we say. However, nothing like the same
level of discontent exists on the main Vim list. Why? Probably
because we are more used to the way Bram operates, and thus know how to
get things done.
[...]
Bram is conservative - as he should be, maintaining a program that
thousands of people depend upon and use happily. But he isn't
intransigent. He just has clear rules which you must follow to be
heard.

Rule 1) Don't request a feature unless you can practically justify it
in terms of productivity. It's no use requesting a feature which will
be rarely used if you could easily achieve it with a macro, and it's
something that would only be used by people who are power users anyway.
[...]
Rule 2) Don't request a change which achieves nothing directly.
Don't wrangle about variable names, the length of files, in
the abstract.
[...]
Rule 3) Show us the code!
Bram doesn't have time to write everything himself. A tidy,
functioning, tested patch with documentation, which meets the above
two rules, is very likely to be accepted. I know from 6 years
experience working on Vim.

Rule 4) Don't ask for a blank cheque.
This is the heart of the kvim developer's problem. They effectively
[...] asked Bram to guarantee in advance that their changes
would be included in the tree, before those changes had even been
written. Knowing Bram, he would never agree to that [...]. Bram *always*
wants to see code first.
[...]
What the group ought to have done was continue with the successful
basic Kvim model, namely: work on a separate tree until they have
something that *works* and then offer the working code to Bram to be
merged.
[...]
```

Aanpassingen in de broncode van Vim zijn dus wel degelijk mogelijk, maar niet zonder dat aan de hierboven gestelde voorwaarden wordt voldaan. En onderdeel daarvan is dat alle code eerst langs Bram Moolenaar gaat voordat de officiële Vim-versie wordt aangepast.

Zoals we hebben kunnen zien, bestaat de groep Vim programmeurs, niet uit één coherent en

eensgezind collectief, maar eerder uit een verzameling individuen, met elk hun eigen motieven en eigen belangen, die dus ook tot meningsverschillen kunnen leiden zoals we in het voorbeeld hierboven hebben kunnen zien. Ook al is Vim qua ontwikkeling misschien minder open, dan qua gebruik, toch is Vim a priori veel opener dan gesloten software. Want iedereen mag de broncode gebruiken om er zijn eigen project mee te starten. In dat geval echter zul je wel alles alleen moeten doen, of een nieuwe gemeenschap van gelijkgestemde gebruikers en ontwikkelaars om je heen weten te verzamelen, hetgeen zeer zeker geen geringe klus is.

§ 5.4 Bevindingen

Vim kan met recht een succesvol computerprogramma genoemd worden wanneer we kijken naar de tijdsduur dat het programma al bestaat, de uitgebreide functionaliteit dat het programma biedt, het grote aantal verschillende computerplatformen dat Vim ondersteunt en het grote aantal actieve gebruikers (schattingen lopen uiteen van tienduizenden tot honderdduizenden gebruikers). Hoe kunnen we dit succes verklaren van een programma dat door een hobbyist geschreven is?

Als er een ding duidelijk is geworden, is dat Vim als programma zodanig in elkaar zit, dat er zoveel mogelijk (verschillende) gebruikers, programma's en computerplatformen worden ingesloten. Hierdoor wordt een zeer sterke coalitie mogelijk van een heterogene groep, die bestaat uit computerhobbyisten, computerprogrammeurs, computers, programma's, tekstbestanden, e-maillijsten, tipsforums, Vimscripts, gebruiksaanwijzingen, boeken, en linuxdistributies. Het slagen van een technologie is niet af te meten aan technische qualificaties, maar aan de sterkte van het netwerk dat het heeft weten te creëren. Vim heeft een groot en uiterst heterogeen netwerk om zich heen weten te creëren, waardoor het uiteindelijk een succesvolle teksteditor is geworden.

De strategie van Moolenaar heeft hier actief aan bijgedragen: "Vim is niet zo'n succesvol programma omdat het technisch zo goed in elkaar zit. Nee, dat komt veel meer omdat ik er iedere dag mee bezig ben. Omdat ik luister naar de wensen en problemen van Vimgebruikers. Omdat ik dat al jaren doe, is Vim uiteindelijk succesvoller geworden dan andere vi-klonen."[50]

Zoals we hebben kunnen zien is volledige openheid een illusie. Projecten hebben een

[50] Interview met Bram Moolenaar, 02-07-2002

projectleider nodig die richting geeft aan het project en zorgt dat alles blijft draaien. Dit impliceert ook meteen dat ‘gebruikers’ die niet betrokken willen zijn bij het ontwikkelproces hun keuzevrijheid delegeren aan deze projectleider. Zij willen namelijk niet over ieder klein detail na hoeven te denken. Zij willen gewoon Vim gebruiken voor hun dagelijks werk en dat kost al genoeg tijd en energie. Hierbij dragen zij hun invloed op de ontwikkeling over aan Bram Moolenaar.

Wil je alles zelf kunnen bepalen dan zal daar werk voor verricht moeten worden. Dit blijkt heel duidelijk in het Vim-verhaal. Het programma Vim is extreem flexibel en bijna alle parameters aangaande het gebruik kunnen door de gebruiker naar eigen wensen en inzichten geconfigureerd worden. Dit heeft echter ook tot gevolg dat je je zult moeten verdiepen in hoe je dit technisch moet realiseren. Een voorbeeld hiervan is de discussie tussen Philippe Fremy en Bram Moolenaar over de toekomst van Vim. In principe kan er op open wijze gediscussieerd worden over de technische ontwikkeling van Vim. Voordat Moolenaar echter verzoeken voor technische veranderingen honoreert, wil hij eerst werkende code zien. Met andere woorden: als je wil dat Vim anders wordt, implementeer deze verandering dan eerst zelf. Dan beslist de hoofdontwikkelaar van Vim daarna, of hij die veranderingen wil doorvoeren in de officiële versie.

1. User versus Producer

Het onderscheid tussen gebruikers (users, consumers) en ontwikkelaars (developers, designers, programmers, producers) is in het geval van vim een fluïde categorie. Er bestaat geen hard onderscheid tussen gebruikers en ontwikkelaars, er is eerder sprake van een spectrum waarop mensen steeds verder opschuiven naarmate ze meer over de werking van Vim weten. Misschien zou een nieuwe term als prosumer of pro-user een betere benaming zijn voor vim-“gebruikers” (Ratto 2002).

Een begrip dat hier heel goed bij aansluit, is het concept ‘participatory use’, een begrip dat David Nichols en Michael Twidale in hun artikel ‘Usability and Open Source Software’ introduceren (Nichols & Twidale 2002). Dit artikel gaat expliciet in op de vraag of open source software niet veel te ingewikkeld is voor de ‘normale’ gebruiker en of open source software makers zich wel genoeg in kunnen leven in de ‘gewone’ gebruiker om daarvoor goede software te maken. Hun uiteindelijke conclusie is, dat software voor gewone gebruikers niet te ingewikkeld moet zijn, maar software voor ‘powerusers’ niet teveel ingebouwde beperkingen mag bevatten. Vim is een voorbeeld van hoe deze beide tegenstrijdige ontwerpvisies toch verenigd kunnen worden.

Dit heeft ook meteen als zeer belangrijke implicatie dat gebruikers, hierdoor mede onderdeel worden van het Vim programma, omdat ze van de gebruikersrol bijna ongemerkt overstappen in de rol van programmeur:

> "By looking at open source software, software usage and development turned out to be inseparable. Therefore the development process starts with the first related idea and ends with the removal of the software from the last hosting computer system. Another effect is the direct or indirect inclusion of all users in the development, though, some of them might only frequently update their software components." (Evers 2000: 91)

Doordat Vim een uitermate flexibel programma is, zou je het kunnen beschouwen als een verzameling Lego-stenen waarmee je van alles kunt bouwen. Wil je een extra functie of moet iets anders, dan kan de gebruiker dat zelf bouwen. Je bent misschien dan wel wat extra tijd in het begin kwijt, maar dat levert uiteindelijk wel precies het eindproduct op, dat je zelf in gedachten had. Wanneer we naar de conclusies van Soogelee kijken dan lijkt Vim verdacht veel op het ideale programma uit haar conclusie over hoe een democratisch programma er uit zou zien.

Om in de termen van Akrich te spreken zit er in Vim veel meer mogelijkheid voor een actieve gebruiker, die zelf bepaalt hoe zijn programma er uit ziet. Je zou in de termen van Soogelee ook kunnen stellen, dat er meerdere scenario's voor gebruik zijn ingebouwd. Vim bevat namelijk mogelijkheden voor toekomstige veranderingen van de werking ervan. Daarnaast wordt er ook uitgegaan van een capabele gebruiker die in de loop van de tijd bijleert, en niet van een gebruiker die gedwongen wordt om na een korte leercurve tegen een betonnen muur te botsen. Van een pejoratieve perceptie van de Vim 'end-user', zoals we dat bij Dijkstra hebben kunnen zien, is geen sprake.

2.Open gebruik versus open ontwikkeling

Een open licentie betekent nog geen open ontwikkelmodel. De vorm van de gekozen licentie (die rechten en restricties voor het *gebruik* omvat) zegt niets over de voorwaarden en vorm voor het *ontwikkelen* van dat programma. Dit is algemeen zo voor open source software. (Het is overigens erg interessant dat Vim hierbij heel erg veel op Linux lijkt. In beide gevallen beslist slechts één persoon over de uiteindelijke keuze voor het opnemen van nieuwe broncode in de officiële distributie en wordt er geen gebruik gemaakt van een algemeen toegankelijk broncodedepot (toegankelijk via bijvoorbeeld het daarvoor algemeen gebruikte programma CVS).

Misschien dat het voor de verdere ontwikkeling van Vim een goed idee zou zijn om een onstabiele ontwikkelversie van Vim uit te brengen waar ontwikkelaars aan kunnen werken, zodat gebruikers die een stabiel programma nodig hebben en ontwikkelaars die de laatste

nieuwe functionaliteit willen uittesten elkaar niet in de weg zitten.

3. Instapvriendelijkheid versus leervriendelijkheid

Bij het programma Vim wordt heel duidelijk de keuze gemaakt voor een zo groot mogelijke gebruikskracht. Dat betekent dat gevorderde vimgebruikers sommige taken ongelofelijk snel en efficiënt kunnen uitvoeren. Voor hun is Vim een uitermate gebruikersvriendelijk programma. Dat is mooi, want Vim wordt vooral door professionele computerprogrammeurs en systeembeheerders gebruikt.

Voor iemand die nog nooit met Vim heeft gewerkt (of zelfs een gevorderde Windowsgebruiker), is het echter een obscuur programma. In de eerste plaats is vaak niet meteen duidelijk hoe het programma überhaupt werkt (het centrale idee van een modale teksteditor begrijpen veel mensen niet meteen).

Herman Bruyninckx (Bruynickx 2002: 4, 6, 12, 29-32) stelt in dit verband twee begrippen tegenover elkaar: namelijk instapvriendelijkheid en leervriendelijkheid (de algemene term gebruiksvriendelijkheid is volgens Bruyninckx een ambigue, holle, nietszeggende term). Volgens Bruyninckx heb je aan instapvriendelijkheid niet zoveel als je al snel tegen de ingebouwde "vriendelijke" beperkingen van het programma oploopt. Vim biedt een ongekend aantal mogelijkheden tot het uitbreiden en aanpassen van de werking van het programma. Daardoor kunnen vervelende en tijdrovende klussen door een klein beetje programmeerwerk omgezet worden in bijvoorbeeld een script dat het hele werk in een toetsindruk op kan knappen. Dit laatste is een voorbeeld van het concept leervriendelijkheid. Zo schrijft Christiansen (1999) hierover:

> "Software engineers need to pay attention to both the keyboard and the mouse, irrespective of whether the program is running in a character-addressable terminal or in a pixel-addressable display. A simple change in visual resolution doesn't obsolete the entire access pattern. Developers should maximize locality of operations by hand positions in order to facilitate an eyes-free operation of the input device. Above all, they should give careful attention to programs destined for heavy use, so that they offer an upward path for all users, and experts are not hampered by the zero-learning-curve demands of non-users. Remember that a user is a novice but once and briefly, and an initiate ever after. Don't require infelicitous input combinations that would hamper efficiency in competent keyboardists. Only when the speed bumps and other awkward reminders of the physical world are removed can a computer user hope to reach that transcendent state of zenning out."

4. Anarchie versus Hiërarchie

Het ontbreken van een formele structuur in de betekenis van een formele organisatie of rechtspersoon, of een duidelijk organogram van de organisatie achter Vim, wil nog niet zeggen dat er geen hiërarchie is, of dat de ontwikkeling van Vim in volstrekte anarchie, zonder sturing op welke manier dan ook, zou plaatsvinden. In tegendeel. Zo is Bram Moolenaar de belangrijkste figuur in de ontwikkeling van Vim. Hij bepaalt welke kant Vim

opgaat en welke veranderingen er in de broncode worden opgenomen. Medevimgebruikers mogen suggesties doen en zelfs broncode aandragen, Bram Moolenaar is uiteindelijk degene die deze nieuwe broncode in de oude integreert.

Daarnaast is er op de vim mailinglijsten een soort impliciete sociale organisatie. Mensen die vaak of snel andere gebruikers met problemen helpen stijgen in achting bij de medegebruikers. Op deze manier verwerven zij dus een positieve reputatie onder hun medegebruikers. Een voorbeeld hiervan is Benji Fisher, die zelfs in het programma Vim (onder 'Credits') genoemd wordt voor zijn bijdrage aan het beantwoorden van vele gebruikersvragen.

5. "Geen geld" betekent niet "Geen economie"

Dat de meeste dingen rondom vim gratis zijn (zoals het programma zelf, de broncode, documentatie, e-mailsupport, aanvullingen op vim, etc) betekent niet dat er geen sprake is van een economisch ecosysteem rondom vim. Ghosh heeft voor de internetcultuur het zogenaamde melting pot model ontworpen, om de levendige informatieruilhandel op het internet te kunnen verklaren (Ghosh 1998). In dit model staan twee noties centraal: In de eerste plaats is er geen sprake van een duidelijke dichotomie tussen koper en verkoper, beide partijen van een transactie geven èn ontvangen: beiden doen beide, zowel consumeren als produceren. Op de tweede plaats zijn de kosten voor extra kopieën van een origineel gelijk aan nul, zonder dat de tienmiljoenste kopie duurder is dan de eerste kopie. De 'meltingpot' is daarmee een soort gigantische informatiekloneringsmachine, die tegen marginale kosten een onbeperkt aantal malen de inhoud kan vermenigvuldigen. Reputatie is een van de factoren die verklaart, waarom mensen elkaar 'gratis' helpen op internet (Raymond 1999).

6. Conclusie

"There's no such thing as a free lunch."

§ 6.1 Onderzoeksresulaten

1. Open software is niet vrij van politiek

Open software is net als gesloten software niet vrij van in de technologie ingebouwde politieke keuzes ten aanzien van het gebruik ervan. Het verschil is alleen dat er door makers van open software andere keuzes worden gemaakt dan door ontwikkelaars van gesloten software, hetgeen te verklaren valt door een verschil in belangen.

Zoals we hebben kunnen zien, kiezen softwarefabrikanten voor een volledig dichtgetimmerd gebruik van hun programma's door middel van zowel technische als juridische beperkingen ten aanzien van het gebruik ervan met als doel het uitschakelen van concurrentie en de maximalisatie van hun winst. Deze strategie heeft echter als keerzijde wel een beperking van de vrijheid van gebruikers tot gevolg. Dit is vanuit de bedrijven bezien acceptabel. Vanuit de gebruikerskant is dit niet altijd acceptabel.

Makers van open computerprogramma's maken andere keuzes, waarbij winstmaximalisatie niet centraal staat, maar een zo groot mogelijk gebruik (zowel qua menselijke gebruikers als aantal verschillende computerplatformen) of een zo groot mogelijke vrijheid in het gebruik van software. Deze keuzes vinden vervolgens hun weerslag in open technologie (in de vorm van open standaarden, open broncode en open licenties) die er voor moeten zorgen dat de technologie niet alleen nu, maar ook in de toekomst open blijft. Hierdoor wordt een zo groot mogelijke vrijplaats voor gebruikers van open software gecreëerd in tegenstelling tot het model van de gesloten software waarin gebruikers zichzelf feitelijk tot betalende knoppendrukkers gereduceerd zien.

2. Open gebruik ≠ open ontwikkeling

Het is belangrijk om op te merken dat open software in de praktijk niet noodzakelijkerwijs een open ontwikkelingsmodel hoeft te bezitten. Bij de openheid, zoals die in licenties van bijvoorbeeld de Free Software Foundation of van het Open Source Initiative wordt gedefinieerd, ligt de nadruk op openheid van gebruik (van de broncode). Hierbij staat een open ontwikkeling niet centraal, maar wordt beschouwd als positief bij-effect van open broncode. Binnen de open softwarewereld bestaat er dan ook een heel spectrum van

ontwikkelingsmodellen die het hele bereik van zeer gesloten tot zeer open omspannen.

De oorspronkelijke hoofdontwikkelaar van een open broncode programma bezit de controle op de toegang tot de distributie van de officiële versie. Daarmee is de hoofdontwikkelaar (dit kan ook een groep ontwikkelaars zijn) tot een 'obligitary point of passage' verworden dat niet omzeild kan worden. Participatie in de ontwikkeling van nieuwe versies van een open broncode programma is daardoor dus geen open, maar een gecontroleerd proces, waarbij de grenzen voor de toegang tot de broncode van de officiële versie nauwkeurig worden afgebakend.

Hierbij moet echter wel opgemerkt worden dat zelfs in het geval van een zeer gesloten ontwikkelingsmodel, de gebruiker van een open programma altijd de vrijheid bezit om op radicale wijze het roer in eigen handen te nemen. In het geval van een programma dat aan open standaarden voldoet, kan de gebruiker een concurrerend alternatief gebruiken om hetzelfde werk gedaan te krijgen. In het geval van een programma dat de broncode meelevert, bezit de gebruiker de vrijheid om deze broncode te veranderen en zijn eigen versie onder gelijkblijvende licentie en nieuwe naam te distribueren, onder eigen gekozen voorwaarden voor de openheid van het ontwikkelingsmodel. Het uitbrengen van een eigen versie van bestaand open broncode programma heet 'forken'. Ondertussen zijn er in de open software wereld meerdere voorbeelden van zogenaamde 'forks' bekend, waarbij zowel het originele programma als de afgesplitste variant naast elkaar hun eigen leven leiden.[51] Wanneer een mede-ontwikkelaar of geavanceerde gebruiker het niet eens is met de beslissingen van de oorspronkelijke hoofdontwikkelaar, bezit hij altijd de vrijheid om zijn eigen project te beginnen. Daarmee is het dus a priori onmogelijk voor een gebruiker om in een zogenaamde 'vendor lock-in' situatie terecht te komen. Ook is het in het geval van open software onmogelijk dat er een situatie ontstaat waarin een maker van computerprogramma's een ongewenste monopoliepositie bezit.

3. Openheid impliceert een open gemeenschap

Doordat in het gesloten model de fabrikant van software het gebruik reguleert ontstaat er ook een heel andere gebruikscultuur en gebruikersgemeenschap, dan het geval is bij open software. In het gesloten model probeert de fabrikant zoveel mogelijk naar zich toe te trekken om daar geld mee te verdienen, bijvoorbeeld bij het oplossen van problemen. In het open model wordt juist zoveel mogelijk gedelegeerd naar de gebruikers zelf. Zo wordt het

[51] Het bekendste voorbeeld zijn de afsplitsingen van de originele BSD-Unix broncode: 386BSD, FreeBSD, NetBSD, OpenBSD en BSD OS. Een ander bekend voorbeeld is de Emacs-fork Xemacs.

beantwoorden van vragen over gebruik van het programma bij voorkeur overgelaten aan andere, meer ervaren gebruikers. Ook voor het vinden van fouten in een open programma en het repareren daarvan, wordt bij voorkeur de hulp van gebruikers ingeroepen. Juist doordat gebruikers van een open programma over de meest uiteenlopende zaken kunnen discussiëren, ontstaat een open cultuur waarin geen blad voor de mond wordt genomen, of waarin 'hidden agenda's' verborgen blijven. Strategische kwesties worden uitgebreid bediscussieerd, waarna meestal door middel van consensus of een meerderheidsbesluit op grond van argumenten en voorbeelden die voor iedereen beschikbaar zijn een beslissing tot stand komt. Omdat openheid in de cultuur van gebruikersgemeenschappen van open software zo'n belangrijke rol speelt, zijn kritische zelfreflectie en leerprocessen mogelijk.

Door deze discussies rondom open programma's over technische details, zien gebruikers dat technologie niet neutraal is. Technische implementaties zijn altijd het gevolg van keuzes die de ontwikkelaars maken. Zodra gebruikers dat inzicht eenmaal verworven hebben, accepteren zij niet langer dat bepaalde (groepen) gebruikers worden uitgesloten, maar zullen zij insluiting eisen, of deze anders zelf implementeren. Gebruikers kunnen zelf nadenken over welke technologie ze willen gebruiken en of ze de bijbehorende consequenties aanvaardbaar vinden of niet.

In het gesloten softwaremodel is een open cultuur onmogelijk. Daar is het juist noodzakelijk dat strategische informatie geheim blijft, omdat anders de concurrentie er mee aan de haal kan gaan. De precieze motivatie van technische details en implementaties blijft daardoor altijd een raadsel voor de gebruiker van gesloten programma's. Zelfs wanneer de fabrikant informatie verschaft is het nooit zeker dat deze informatie correct of volledig is. Een gelijkwaardige discussie tussen gebruikers en de fabrikant is dan ook per definitie onmogelijk.

4. Openheid vergt inspanning

Er bestaat niet zoiets als een multiple-choice democratie waarin op inspanningsloze wijze tussen verschillende kant-en-klaar oplossingen gekozen kan worden. Democratische technologie zonder dat een gebruiker daar moeite voor hoeft te doen is een illusie. Democratische technologie waarop gebruikers invloed uit kunnen oefenen, vereist dat er werk wordt verricht door gebruikers wanneer zij ook daadwerkelijk invloed willen uitoefenen.

"Free software is only free, if your time is free"[52], is binnen de vrije software wereld een veel gebezigd adagium. Deze conclusie geldt misschien wel voor keuzevrijheid binnen een democratie in het algemeen. Die biedt namelijk de mogelijkheid voor inspraakprocedures, maar dan moet je daar wel naar toe gaan, je van te voren in de materie verdiepen en op het moment zelf ook daadwerkelijk je mond opendoen en met goede argumenten en eventuele alternatieven komen.

5. Niet willen kiezen ≠ niet kunnen kiezen

Ondertussen blijft de volgende vraag over: Bestaat er in het geval van open software niet een fundamentele ongelijkheid tussen mensen met technische expertise en die zonder? Ik heb beschreven hoe open software een open gebruik mogelijk maakt waarbij de gebruiker op meerdere niveaus kan kiezen om de werking van de door hem gebruikte computertechnologie aan te passen aan zijn eigen wensen en behoeften. In plaats van 'configuring the user' waarbij de technologie centraal staat (of de winstgevendheid van een softwareproducent) en de gebruiker op de tweede plaats komt, zou er sprake moeten zijn van 'configuring the technology' waarbij de gebruiker op de eerste plaats komt en zelf kan kiezen welke opties die de technologie hem biedt wel of niet wel gebruiken, onder het motto: "people first, technology second".

Hierbij faciliteert de fundamentele openheid van open broncode programma's de participatie in discussies over deze mogelijkheden voor configuratie door een laagdrempelige toegankelijkheid te garanderen. Door gebruik te maken van open standaarden, een open informatie-uitwisseling, open broncode en een open ontwikkelproces komt er meer keuzevrijheid bij de gebruiker te liggen in plaats van bij de maker van computerprogramma's.

Het is echter onzinnig om mensen die niet willen meebeslissen, te dwingen om zich te bemoeien met de interne werking van computerprogramma's in het bijzonder, of technologie in het algemeen. Tegelijkertijd echter mag het niet zo zijn, dat mensen a priori worden uitgesloten van het kunnen uitoefenen van invloed op de door hen gebruikte technologie. Het moet altijd mogelijk zijn om technologie voor eigen gebruik aan te kunnen passen. Of deze wijzigingen vervolgens ook moeten worden doorgevoerd in de officiële versie, is een andere kwestie.

6. Openheid impliceert een open toekomst

Open software maakt een toekomst mogelijk, waarin de weg naar een eventueel onvoorzien

[52] Jamie Zawinski: Linux is only free if your time has no value (http://www.jwz.org/doc/linux.html)

gebruik ervan nog helemaal open ligt. Doordat de broncode van open programma's beschikbaar is kan zij altijd aangepast worden aan veranderende wensen en inzichten op het moment dat dat daadwerkelijk nodig is. Dit in tegenstelling tot het gesloten model waarbij de fabrikant controleert en bepaalt welk soort gebruik als succesvol wordt beschouwd.

Om te voorkomen dat we toekomstige generaties uitsluiten van een creatief gebruik van ons digitale gemeenschapsgoed, pleit ik daarom voor een expliciete keuze voor democratische computertechnologie die mensen insluit, in plaast van uitsluit en die de flexibiliteit bezit om zich aan te passen aan de wensen van morgen, die we vandaag nog niet kunnen overzien. Alleen dan zullen we nieuwe technologieën zien ontstaan, die qua mogelijkheden misschien dezelfde impact zullen hebben als de introductie van het internet en Wereld Wijde Web, dat uiteindelijk de weg vrijmaakte naar een globale samenwerking bij de ontwikkeling van computerprogramma's.

§ 6.2 Aanbevelingen

Open software is niet per definitie beter dan gesloten software, dat hangt namelijk van de context van het gebruik af. Bovendien is het onrealistisch om te verwachten dat commerciële bedrijven hun concurrenten gaan financieren door volledig open te worden en al hun duurbetaalde innovaties 'out in the open' bekend te maken en volledig vrij te geven. Niettemin zijn er echter situaties waarin openheid noodzakelijk is. Bijvoorbeeld wanneer er geen alternatieven voor gebruikers zijn, zodat ongewenste machtconcentraties kunnen ontstaan. Daarnaast dient de overheid, die er niet is om commerciële bedrijven te subsidiëren, maar om de belangen van haar burgers te dienen, een open technologiebeleid na te streven[53]. Daarom is het gebruik van open software in de volgende situatie beter te rechtvaardigen dan het gebruik van gesloten software, namelijk:

1. Bij de overheid[54]: voor het voorkomen van een afhankelijkheidsrelatie van softwarefabrikanten dienen open software en open standaarden gebruikt te worden. Het verstrekken van electronische informatie die niet aan open standaarden voldoet aan burgers is ontoelaatbaar, net zoals het geheim zijn van de broncode van bijvoorbeeld stemmachines[55].

[53] Voor een volledig uitgewerkt voorstel voor het gebruik van open software zie het volgende rapport van Groen Links: Vendrik, Kees & Tilburg, Rens van (2002).

[54] NRC Handelsblad van 22 februari 2003: "Overheidsinstellingen moeten niet alleen meer vrijheid krijgen om zelf te kiezen welke software ze gebruiken, ook zou software die met belastinggeld is ontwikkeld voor één overheidsorganisatie vrij door andere instanties gebruikt moeten kunnen worden. Dat hebben de departementen Economische Zaken en Binnenlandse Zaken gisteren bekend gemaakt."

[55] Hegener, Michiel (2003). Op weg naar de uitslag. NRC Handelsblad, 20 janauri 2003, p. 12. In een artikel over controle op de software in electronische stemmachines voor de tweede kamerverkiezingen in Nederland, valt het volgende te lezen:

2. Op scholen: Voor computeronderwijs zou het volgende motto moeten gelden: “Teach them the fundamentals, not just the product” in plaats van “I can train a dog, but does he understand?”[56] In het onderwijs gaat het om het overdragen van inzicht, en niet om het leren werken met specifieke en snel-verouderende softwareproducten. Open technologie biedt door haar openheid per definitie de mogelijkheid om de interne werking ervan te bestuderen.[57]

3. Ter bescherming van opslag en transport van privacy-gevoelige informatie:
De methodiek van ‘security trough obscurity’ zoals die in gesloten software wordt toegepast blijkt in de praktijk niet te werken. Keer op keer worden geheime protocollen gekraakt. Alleen door gebruik te maken van open beveiligingsstandaarden kan iedereen controleren of zijn gegevens ook daadwerkelijk goed beveiligd zijn en niet misbruikt kunnen worden. Wanneer er een beveiligingslek bekend wordt, kan door de aanwezigheid van de broncode daar snel en adequaat op gereageerd worden.

4. Bij instellingen met kleine budgetten voor ICT-oplossingen: Voor non-commerciële instellingen of organisaties in ontwikkelingslanden waar weinig geld beschikbaar is voor ICT-oplossingen is open software eerste keus[58]. De open licentie maakt samenwerking en het delen van software met andere partijen mogelijk. Hierdoor blijft er meer geld over voor andere meer noodzakelijke doelen, dat anders naar softwarefabrikanten was gegaan.

5. In onderzoeksinstellingen: Op plaatsen waar software wordt ontwikkeld met overheidsgeld is het wenselijk dat het eindresultaat vrij hergebruikt kan worden op andere plekken, zodat er efficiënter wordt omgegaan met gemeenschapsgeld voor de ontwikkeling van software. Zowel overheidsinstanties, universiteiten en hogescholen, als andere door de overheid gesubsidieerde instellingen, zouden hun software moeten kunnen delen met iedereen die daar profijt bij heeft. Wanneer voor een bepaalde toepassing nog geen open programma beschikbaar is, verdient het in eerste instantie de voorkeur om die te ontwikkelen, of wanneer dat niet mogelijk is, zolang een zo open mogelijk alternatief dat gebruik maakt van open standaarden te gebruiken.

“Het mooie van de [papieren] biljetten is dat betrokken burgers kunnen posten op de stembureaus en mee kunnen kijken bij het tellen na sluiting. Dat recht hebben we allemaal. Maar de drie electronische technieken geven weinig ruimte voor waakzaamheid. Ir. Peter Knoppers, elektrotechnicus en onderzoeker aan de TU Delft stelt: ‘Bij electronisch stemmen heb je maar te geloven wat er gebeurt tussen het indrukken van de knop en de uitslag aan het eind van de dag. De broncode van de software is geheim. Als ik al niet kan controleren wat er gebeurt, kunnen gewone kiezers dat ook niet.’”

[56] Bron van beide citaten: Lezing van Jon Hall, CEO Linux International op Free and Open Source Software Developers Meeting, Brussel, 8 februari 2003 (zie: http://www.fosdem.org/)

[57] Voor een volledig uitgewerkt plan voor het gebruik van open software in het onderwijs zie: Bruyninckx (2002).

[58] Een interessant voorbeeld hiervan is de samenwerking tussen UNESCO en de Free Software Foundation die resulteerde in de UNESCO Free Software Portal (http://www.unesco.org/webworld/portal_freesoft/index.shtml)

§ 6.3 Bijdrage aan STS-onderzoek

Door middel van deze scriptie lever ik een bijdrage aan de discussie over de democratisering van de technologie. Het uitganspunt van mijn scriptie is dat een louter filosofische reflectie op de vraag naar democratische technologie te grofmazig is, om daarmee de daadwerkelijke praktijk rondom het gebruik van technologie en een eventuele invloed van gebruikers hierop te kunnen analyseren. Bovendien blijkt dat de simpele notie van democratische technologie in de praktijk fundamentele problemen opwerpt, waardoor een simpele implementatie niet mogelijk is. Een checklist zoals Sclove die in zijn boek 'Democracy and Technology' hanteert is een mooi uitgangspunt, maar ook niet meer dan dat.

Pas wanneer de constitutie van het daadwerkelijke netwerk van actanten in de praktijk in detail wordt bestudeerd, bijvoorbeeld door het gebruik van gedetailleerde gevalsstudies, kan men tot interessantere conclusies komen, die direct relevant zijn voor eventueel te implementeren wenselijk technologiebeleid. Door noties en concepten, afkomstig uit het hedendaagse veld van de Science and Technology Studies als analytische instrumenten te gebruiken kan ik in mijn scriptie een gedetailleerder en genuanceerder standpunt innemen inzake democratische technologie in de daadwerkelijke praktijk. Door het hanteren van begrippen als script, inclusie of 'obligitary point of passage' kan ik de onderlinge relaties en invloed van technische artefacten en actoren op elkaar beschrijven, zodat het mogelijk is om tot een algemene beschrijving te komen van de gebruikspraktijk van open computertechnologie. Wanneer vervolgens van het niveau van de gedetailleerde gevalsstudies wordt geabstraheerd naar een meer algemeen en theoretisch niveau, komt men tot interessantere en nuttigere conclusies.

Ik pleit dan ook voor een empirisch-filosofische benadering van het zoeken naar een oplossingsrichting voor het ontwikkelen van democratische technologie. Daarnaast wil ik er op wijzen dat het naïef is om te denken dat er voor het democratiseringsproces van technologie voor iedere technologie en iedere gebruiker op iedere plek en op iedere tijd en in iedere context één kant-en-klare universele altijd-geldige standaardoplossingen gehanteerd zou kunnen worden. Iedere keer opnieuw is er sprake van andere behoeften en belangen die het hanteren van standaardoplossingen onmogelijk maken en waardoor iedere keer opnieuw noodzaak bestaat tot een gedegen en gedetailleerd onderzoek van de desbetreffende situatie.

In mijn scriptie configureer ik, net zoals Woolgar, Abbate en Sclove dat op theoretisch niveau en Microsoft en de open software gemeenschap op praktisch niveau doen, mijn eigen technologiegebruiker. Deze gebruiker wijkt af van het standaardbeeld van de burger, die zijn democratische invloed laat gelden door eens in de vier jaar te participeren in represenatieve

democratische verkiezingssystemen, en daarmee zijn autonomie om keuzes te maken over technologieontwikkeling delegeert aan een professionele expert die hiervoor betaald krijgt. Ook wijkt mijn democratische technologiegebruiker af van de gebruiker in zijn rol als consument in het standaardbeeld waarbij hij zijn keuzes delegeert aan de producent van technologie, die specialisten door middel van gebruikersonderzoeken en in focuspanels op marketingafdelingen de keuzes van deze consument vorm geeft.

De democratische technologiegebruiker die ontstaat uit het beeld van de open software gemeenschap, is een actief participerende en reflexieve gebruiker die bewust en actief keuzes maakt en betrokken is bij besluitvormingsprocedures. Door de grote tijds- en energieinvesteringen die hiermee gepaard gaan kan deze gebruiker onmogelijk bij alle kwesties rondom technologieontwikkeling en gebruik betrokken zijn. Slechts daar waar hij in zijn dagelijks leven of voor zijn werk het meest afhankelijk is van de door hem gebruikte technologie, laat hij zijn invloed op actieve wijze door participatie gelden. Hierdoor is echter ook de democratische opbrengst het grootst. Alleen gebruikers die ook daadwerkelijk grote belangen hebben, zullen moeite en tijd investeren om zichzelf centraal vertegenwoordigd te zien op de plaats waar de beslissingen genomen worden die later de grenzen zullen bepalen van wat wel en niet mogelijk is voor gebruiker. De visie zoals dat iedereen altijd en overal in participeert is dan ook een utopie, omdat daarvan voor velen gewoonweg de tijd ontbreekt.

Tegelijkertijd echter betekent ‘echte’ gebruikersparticipatie niet een vertegenwoordigd zijn door specialisten en professionals, maar een daadwerkelijk meedoen, meebouwen, meediscussiëren, meeontwerpen en meeschrijven aan technologieontwikkeling. Door dit intensieve karakter van de participatie door gebruikers is deze vorm van democratische technologie altijd partieel en spatieel gelocaliseerd, beperkt door de socio-technische netwerkgrenzen. Door dit inzicht hoop ik dat het concept openheid als een conceptuele leidraad kan dienen voor een analyse van de socio-technische netwerkgeografie rondom technologieën. En daarmee kan bijdragen aan verder onderzoek naar ontwikkelings- en gebruiksmodellen voor moderne technologie, die een gebruiker niet alleen dwingt om een passieve rol in te nemen, maar daadwerkelijk de mogelijkheid biedt om tot een actieve en creatieve invulling van zijn gebruikersrol te kunnen komen.

§ 6.4 Suggesties voor verder onderzoek

Deze scriptie kan worden beschouwd als de eerste aanzet tot een uitgebreide studie naar openheid als concept om participatie van gebruikers in technologie-ontwikkeling te faciliteren. Een volgende stap kan een beschouwing zijn van de manier waarop virtuele

gemeenschappen van gebruikers en ontwikkelaars ontstaan, zich verder ontwikkelen en welke onderlinge relaties en principes hierbij een rol spelen.

In deze scriptie ligt de nadruk op het gebruik van open source software door gebruikers die een hoge technische expertise bezitten. Gebruik van open computertechnologie voor eenvoudige taken en dagelijks gebruik is daardoor relatief onderbelicht gebleven. Omdat juist deze groep gebruikers zich niet druk maakt over het kunnen uitoefenen van invloed op de haar gebruikte computertechnologie, opvattingen hierover niet expliciet maakt en deze gebruikers zich ook niet verenigen in gebruikersgemeenschappen, zijn zij veel moeilijker benaderbaar voor onderzoek.

Wanneer over vijf tot tien jaar het gebruik van open computerprogramma's is doorgesijpeld van de huidige 'early-adaptors' naar de grote massa eenvoudige computergebruikers, in de vorm van gebruikersapplicaties zoals de internetbrowser Mozilla en het officepakket OpenOffice, en er een nieuwe generatie gebruikers binnen het paradigma van open computerprogramma's is opgegroeid, is het zinnig om opnieuw vanuit de vraagstelling over de bijdrage die open computerprogramma's kunnen leveren aan het democratiseringsproces van informatietechnologie, hier onderzoek naar te doen.

Open software is een relatief nieuw onderwerp binnen het STS-veld, dat zeer zeker de moeite waard is om verder onderzocht te worden. Want één ding is duidelijk: open software en open standaarden zijn niet meer weg te denken als fundament van het internet en daarmee van ons in snel tempo digitaliserende samenleving. Door het insluitende karakter van open computertechnologie is het slechts een kwestie van tijd, voordat zij een zaak van ons allemaal is.

Bibliografie

Boeken & gedrukte artikelen

Abbate, Janet (1999). *Inventing the Internet*. Cambridge, Massachusetts: The MIT Press.

Achterhuis, Hans. Lewis Mumford: cultuur en techniek. In: Achterhuis, Hans (ed.) (1992). *De maat van de techniek*. Amsterdam: Ambo, pp. 205-59.

Akrich, Madeleine (1992). The De-Scription of Technical Objects. In: Bijker, W.E. & Law, J. (1992). *Shaping Technology / Building Society*. Cambridge, MA.: The MIT Press, pp. 205-24.

Anderson, Annelise (2001). *FreeBSD: An Open-Source Operating System for Your Personal Computer*. Portola Valley: The Bit Tree Press.

Berners-Lee, Tim (2000, 1e druk 1999). *De wereld van het world wide web. Over het oorspronkelijke ontwerp en de uiteindelijke ontwikkeling van het www, geschreven door de uitvinder zelf* (vert. van Weaving the Web - The original design and ultimate destiny of the World Wide Web by its inventor). Amsterdam: Nieuwezijds.

Bijker, Wiebe E., Hughes, Thomas P. & Pinch, Trevor J. (eds.) (1987). *The Social Construction of Technological Systems. New Directions in the Sociology and History of Technology.* Cambridge, MA.: MIT Press.

Bijker, Wiebe (1987). De sociale constructie van netwerken en technische systemen; nieuwe perspectieven voor de techniekgeschiedenis. *Jaarboek voor de Geschiedenis van Bedrijf en Techniek*, 4 (1987), pp. 7-24.

Bijker, Wiebe & Law, John (eds.) (1992). *Shaping technology / Building society: studies in sociotechnical change*. Cambridge, MA: The MIT Press

Bijker, Wiebe (1995a). *Of Bicycles, Bakelites, and Bulbs. Toward a Theory of Sociotechnical Change*. Cambridge, Massachusetts: The MIT Press.

Bijker, Wiebe (1995b). *Democratisering van de Technologische Cultuur*. Eijsden: Wiebe Bijker.

Bloor, Michael (et al.) (2001). *Focus groups in social research*. London: SAGE.

Callon, Michel (1986). Some elements of a sciocology of translation: domestication of the scallops and the fishermen of St Brieuc Bay. In: Law, John (ed.) (1986). *Power, action and belief: a new sociology of knowledge?* London: Routhledge & Kegan Paul, pp. 196-233.

Castells, Manuel (2000, first edition 1996). *The Information Age: Economy, Society and Culture*. Oxford: Blackwell Publishers.

Castells, Manuel (2001). *The Internet Galaxy. Reflections on the Internet, Business, and Society*. Oxford: Oxford University Press

Ceruzzi, Paul E. (1998). *A History of Modern Computing*. Cambridge, Massachusetts: The MIT Press

DiBona, Chris; Ockman, Sam & Stone, Mark (eds.) (1999). *Open Sources: Voices of the Open Source Revolution*. Sebastopol: O'Reilly & Associates.

Egyedi, Tineke (1996). *Shaping Standardization. A study of standards processes and standards policies in the field of telematic services*. Delft: Delft University Press

Feller, Joseph & Fitzgerald, Brian (2002). *Understanding Open Source Software Development*. London: Addison-Wesley.

Gancarz, Mike (1995). *The Unix Philosophy*. Newton: Butterworth-Heinemann.

Ghosh, Rishab Aiyer; Krueger, Bernhard; Glott, Ruediger & Robles, Gregorio (2002). *Free/Libre and Open Source Software: Survey and Study*. Maastricht: Infonomics.

Gilles, James & Cailliau, Robert (2000). *How the Web was Born. The Story of the World Wide Web*. Oxford: Oxford University Press.

Hegener, Michiel (2003). Op weg naar de uitslag. *NRC Handelsblad*, 20 janauri 2003, p. 12.

Himelein, Gerald (2002). Der digitale Knebel. Intel und Microsoft wollen Daten vor dem Anwender schützen. *C't*, 2002 heft 15, pp. 18-20.

Ihde, Don (1996). *Technology and the Lifeworld. From Garden to Earth*. Bloomington, Ind.: Indiana University Press.

Jansen & Janssen (1999). *Luisterrijk. Een gids over afluisteren*. Amsterdam: Papieren Tijger.

Joerges, Bernward (1999). Do Politics Have Artefacts? In: *Social Studies of Science*. Volume 29 Issue 03 - Publication Date: 1 June 1999, pp. 411-432.

Johnson, Steven (2001). *Emergence. The connected lives of ants, brains, cities, and software.* New York: Touchstone (Simon & Schuster).

Jones, Steve (ed.) (1999). *Doing Internet research: critical issues and methods for examining the Net*. Thousand Oaks, CA: Sage Publications.

Kernighan, Brian & Pike, Rob (1984). *De UNIX programmeeromgeving*. Schoonhoven: Academic Service (vert. van The Unix Programming Environment Englewood Cliffs, N.J.: Prentice-Hall, 1984)

Kleiner, Kurt & Graham-Rowe, Duncan (2000). Go forth and multiply. Only diversity will protect PC's against Love Bugs. In: *New Scientist*: 13 may 2000.

Latour, Bruno (1995, eerste druk 1987). *Wetenschap in actie. Wetenschappers en technici in de maatschappij* (vert. van Science in Action). Amsterdam: Ooievaar Pockets.

Latour, Bruno (1992). A new vocabulary of semiotics. In: Bijker, Wiebe & Law, John (eds.) (1992). *Shaping technology / Building society: studies in sociotechnical change*. Cambridge, MA: The MIT Press.

Latour, Bruno (1999). On recalling ANT. In: Law, John & Hassard, John (eds.) (1999). *Actor network theory and after*. Oxford: Blackwell Publishers, pp. 15-25.

Law, John & Hassard, John (eds.) (1999). *Actor network theory and after*. Oxford: Blackwell Publishers

Law, John (1999). After ANT: complexity, naming and topology. In: Law, John & Hassard, John (eds.) (1999). *Actor network theory and after*. Oxford: Blackwell Publishers, pp. 1-25.

Levy, Steven (1994, eerste druk 1984). *Hackers. Heroes of the computer revolution*. London: Penguin Group.

MacKenzie, Donald and Wajcman, Judy (eds.) (1985). *The Social Shaping of Technology. How the refrigerator got its hum*. Milton Keynes: Open University Press.

MacLuhan, Marshall & Fiore, Quentin (1989, 1967). *The Medium is the Message*. New York: Touchstone.

McKusick, Marshall (1999). Twenty years of Berkeley Unix: From AT&T-Owned to Freely Redistibutable. In: DiBona, Chris; Ockman, Sam & Stone, Mark (eds.) (1999). *Open Sources: Voices of the Open Source Revolution*. Sebastopol: O'Reilly & Associates.

Mol, Annemarie (1999). Ontological politics. A word and some questions. In: Law, John & Hassard, John (eds.) (1999). *Actor network theory and after*. Oxford: Blackwell Publishers, pp. 74-89.

Moody, Glyn (2001). *Rebel Code. Linux and the Open Source Revolution*. London: Penguin Books.

Mumford, Lewis (1964). Authoritarian and Democratic Technics. *Technology and Culture*, 1964, 5, pp. 1-8.

NRC Handelsblad (22-02-2003). Overheid: Vrijheid bij keuze software.

Raymond, Eric (1999). *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. Sebastopol: O'Reilly & Associates.

Raymond, Eric (1996, first edition 1991). *The New Hacker's Dictionary*. Cambridge, MA: MIT Press.

Sclove, Richard (1995). *Democracy and Technology*. New York: The Guilford Press.

Smith, M. R. and Marx, L. (eds.) (1994). *Does technology drive history? The dilemma of technological determinism*. Cambridge, MA: MIT Press.

Soogelee, Bertil (1996). *Van één inscript naar scenario's. Gebruikers betrekken bij het ontwerpen van technologie*. Maastricht: Rijksuniversiteit Limburg.

Stephenson, Neal (1999). *In the beginning ... was the command line*. New York: Avon Books.

Toffler, Alvin (1980). *The Third Wave*. New York: Morrow.

Torvalds, Linus & Diamond, David (2001). *Gewoon voor de fun* (vert. van Just for fun). Uithoorn: Karakter Uitgevers.

Young, Robert & Goldman Rohm, Wendy (2000). *De Linux Factor*. (vert. van Under the radar. How Red Hat Changed the Software Business - And Took Microsoft by Surprise, 1999). Schoonhoven: Academic Press.

Van Wendel de Joode, Ruben; Hans de Bruyn & Michel van Eeten (2003). *Protecting the virtual commons. Self-organizing open source communities and innovative interllectual property regimes*. ITER-reeks

Vries, Gerard de (1995, 3). *De ontwikkeling van wetenschap. Een inleiding in de wetenschapsfilosofie*. Groningen: Wolters-Noordhoff.

Wayner, Peter (2000). *Free for All. How Linux and the Free Software Movement Undercut the High-Tech Titans*. New York: HarperCollins.

Wilde, Rein de (1997). Sublime Futures. Reflection on the Modern Faith in the Compatibility of Community, Democracy, and Technology. In: Mykleburst, Sissel (ed.). (1997). *Technology and Democracy: Obstacles to Democratization - Productivism and Technocracy*. Oslo: TMV.

Williams, Sam (2002). *Free as in Freedom: Richard Stallman's Crusade for Free Software*. Sebastopol: O'Reilly & Associates.

Winner, Langdon (1980). Do artifacts have politics? Reprinted in: MacKenzie, Donald and Wajcman, Judy (eds.) (1985). *The Social Shaping of Technology. How the refrigerator got its hum*. Milton Keynes: Open University Press, pp. 26-38.

Winner, Langdon (1991). Artifacts/Ideas and Political Culture. Reprinted in: Teich, Albert H. (ed.) (1993, 6th edition). *Technology and the Future*. New York: St. Martins Press.

Wyatt, Sally (1998). *Technology's Arrow. Developing Information Networks for Public Understanding in Britain and the United States*. Maastricht: Universitaire Pers Maastricht.

Woolgar, Steve (1990). Configuring the user: the case of usability trials. Paper for discussion at Discourse Analysis Workshop, University of Lancaster, 25-26 September 1990. In: Law, John (1991). *A sociology of monsters: essays on power, technology and domination*. London: Routhledge, pp. 57-99.

Woolgar, Steve & Cooper, Geoff (1999). Do Artefacts Have Ambivalence? Moses' Bridges, Winner's Bridges and Other Urban Legends in S&TS. *Social Studies of Science*, Volume 29 Issue 03 - Publication Date: 1 June 1999, pp. 433-49.

Webpublicaties

Abbate, Janet (1995). *'Open Systems' and the Internet.* Paper presented at the joint Society for Social Studies of Science/Society for History of Technology conference, Charlottesville, October 19-22 1995. Online available: http://www.wam.umd.edu/~abbate/papers/4S.html.

Braeken, Johan (2002). *MS EULA Viewer*. Online available: http://www.knudde.be/index.php?page_name=ms_eula_viewer

Bruyninckx, Herman (2002). *De democratisering van de ICT. Een onafhankelijke visie op Informatie- en Communicatie-Technologie in het onderwijs* (Versie 3.7, 6 februari 2002). Online available: http://people.mech.kuleuven.ac.be/~bruyninc/ictvisie.html.

Christiansen, Tim (1999). *Zenclavier: Extreme Keyboarding.* Online availabe: http://www.oreilly.com/news/zenclavier_1299.html

Dijkstra, Edsger (1982) On Webster, users, bugs and Aristotle (EWD618). (Online available: http://www.cs.utexas.edu/users/EWD/ewd06xx/EWD618.PDF) In: Edsger W. Dijkstra (1982). *Selected Writings on Computing: A Personal Perspective*. New York: Springer-Verlag. (p. 288-91)

Dijkstra, Edsger (1985). *EWD 920: Can computing science save the computer industry?* Online available: http:/www.cs.utexas.edu/users/EWD/ewd09xx/EWD920.PDF

Evers, Steffen (2000). *An Introduction to Open Source Software Development*. (Revision of diploma thesis with the original title "Development Environments For Open Source Software" at the Technische Universität Berlin.) Online available: http://user.cs.tu-berlin.de/~tron/opensource/opensource.pdf

Franke, Nikolaus & Hippel, Eric von (2002). *Satisfying Heterogeneous User Needs via Innovation Toolkits: The Case of Apache Security Software*. MIT Sloan School of Management Working Paper No. 4341-02, January 2002. Online available: http://web.mit.edu/evhippel/www/ApacheHeteroWP.pdf.

Ghosh, Rishab (1998). Cooking pot markets: an economic model for the trade in free goods and services on the internet. *First Monday*, Vol. 3, Issue 3. (Online available: http://ww.firstmonday.dk/issues/issue3_3/ghosh/)

Hemel, Armijn (2001). Interview with Vim's Bram Moolenaar. *EUP E-zine*, 09-10-2001. Online available: http://e-zine.nluug.nl/hold.html?cid=180.

Hippel, Eric von (2002). *Horizontal innovation networks - by and for users*. MIT Sloan School of Management Working Paper, No. 4366-02, June 2002. Online available: http://web.mit.edu/evhippel/www/UserNetworksWP.pdf.

Mallett, Steve (2002). *Interview Bram Moolenaar of Vim fame on his new project A-A-P* (posted 30 october 2002). Online available: http://news.osdir.com/modules.php?op=modload&name=News&file=article&sid=27

Moolenaar, Bram (2000). *The continuing story of Vim*. Paper submitted for the Linux200.nl conference, 9-10 Oct 2000 in Ede. Online available: http://www.moolenaar.net/vimstory.txt.

Moolenaar, Bram (2002). *VIM: the popular text editor*. In: Free Software Magazine, Vol. 1, Issue No. 1, Jan 2002. Online available: http://www.rons.net.cn/english/FSM/Vim.

Nichols, David M. & Twidale, Michael B. (2002). *Usability and Open Source Software*. Online available: http://www.cs.waikato.ac.nz/~daven/docs/oss-wp.html

Ratto, Matt (2000). *Producing users and using producers*. Draft of paper from PDC 2000. Online available: http://communication.ucsd.edu/mratto/papers/index.html

Stallman, Richard (1996). *Categories of Free and Non-Free Software*. Boston: Free Software Foundation. Online available: http://www.gnu.org/philosophy/categories.html.

Vendrik, Kees & Tilburg, Rens van (2002). *Software open u! Plan van aanpak ter stimulering van open software (open standaarden en open source).* GroenLinks partijdocument. Online available: http://www.groenlinks.nl/partij/2dekamer/publikaties/Softwarenota191102.pdf

Vermeer, Martin (1999). Linux and Ethnodiversity. In: *Linux Today*, 21-01-1999. Online available: http://linuxtoday.com/news_story.php3?ltsn=1999-01-21-007-05-OP

Wheeler, David A. (2002). *Why Open Source Software / Free Software (OSS/FS)? Look at the numbers!* Online available: http://www.dwheeler.com/oss_fs_why.html.

Working Group on Libre Software (2000). *Free Software / Open Source: Information Society Opportunities for Europe?* Online available: http://eu.conecta.it/.

Ziegler, Günther M. (2000). How (La)TeX changed the face of Mathematics. An E-interview with Leslie Lamport, the author of LaTeX. *DMV-Mitteilungen* 1/2000. Online available: http://research.microsoft.com/lamport/pubs/lamport-latex-interview.pdf

Bronnen voor Vim-casus

‘Vim.Online’:
http://www.vim.org, http://vim.sourceforge.net/, ftp://ftp.vim.org/,
http://sourceforge.net/projects/vim/, http://sourceforge.net/projects/vimonline/ (website)

De vim@vim.org emaillijst:
http://groups.yahoo.com/group/vim/messages

Interview met Bram Moolenaar gehouden op 02-07-2002

Virtuele focus groep studie:
Thomas Capricelli,
Tim Chase,
Piet Delpoort,
Philippe Fremy,
Gary Johnsen,
Luid Jure,
Catherine Lina,
Steve Litt,
Moshe Kaminsky,
Mikolaj Machowski,
Leonid Mamtchenkov,
Mickael Marchand,
Vince Negri,
Dirk Schenkewitz,
Alan Schmitt,
Stanislav Sitar,
Matthias Ulrich,
Jonathan Wakely,
Robert Werner,
Derek Wyatt,
Carfield Yim en
Dirk Zimmerman.