

Over een open keuze voor gesloten software

§1. Inleiding: open software voor ontwikkelingslanden

In februari 2002 maakte de Zuid-Afrikaanse president Thabo Mbeki het nieuws bekend dat Microsoft alle 32.000 openbare scholen gratis van software zou voorzien. Een maand eerder echter verklaarde de Zuid-Afrikaanse National Advisory Council on Innovation (NACI) dat de overheid het gebruik van ‘open’ software zou moeten stimuleren, omdat “it has the potential to empower people in ways that proprietary software (such as Microsoft’s) simply does not allow”.¹ Dit voorbeeld illustreert het probleem van gesloten computerprogramma’s versus open software in ontwikkelingsgebieden. Op de korte termijn bieden donaties van gesloten software door bedrijven zoals Microsoft wel degelijk een verbetering van de situatie ten aanzien van de aanwezige informatietechnologie; de vraag is echter wat de gevolgen zijn van het aannemen van dergelijke donaties voor de langere termijn. Er bestaat dan ook een levendige discussie over welk type software (open of gesloten) het beste bijdraagt aan het vinden van oplossingen voor de specifieke problemen op ICT-gebied in ontwikkelingsgebieden.

Binnen het debat over het gebruik van open broncode programma’s (die gebruik maken van open standaarden en waar de broncode bijgeleverd wordt en aangepast mag worden) versus gesloten programma’s (die gebruik maken van geheime opslagformaten en zonder broncode worden geleverd) wordt vaak beweerd dat open source / free software een ideale oplossing zou zijn voor gebruik in ontwikkelingsregio’s.² De redeneertrant dat open broncode / vrij software beter zou zijn voor ontwikkelingsgebieden komt men in de eerste plaats vooral binnen in de wereld van de pleitbezorgers van dit type software. Daarnaast dragen organisaties die zich bezig houden met ontwikkelingshulp deze visie ook steeds vaker uit. Zo biedt Unesco zelfs een volledige website aan op internet over dit thema.³

In dit essay probeer ik een overzicht te geven van de gebruikte argumenten die door voorstanders van open software worden ingezet. Daarna zal ik kritisch op reflecteren op deze argumenten om tot een conclusie te kunnen komen over de vraagstelling naar de wenselijkheid van de inzet van open software in ontwikkelingsgebieden. Alvorens ik echter

¹ Teresa Peters (2002). ‘Debate between open source and proprietary software becomes real for developing countries’. Online.available: <http://www.bridges.org/>

² Voor een kritische informatiebron over dit onderwerp, zie: Nikolai Bezroukov (2002, eerste versie 1996), ‘Internet and Open Source in Developing Countries Webliography’, Online.available: http://www.softpanorama.org/Social/oss_in_developing_countries.shtml

³ Zie Unesco’s Free Software Portal: http://www.unesco.org/webworld/portal_freesoft/index.shtml

dieper op deze discussie zelf in kan gaan, leg ik eerst kort uit wat het fundamentele verschil is tussen open software en gesloten software.

§2. Twee typen software: open en gesloten

Binnen het huidige veld van de ontwikkeling en het gebruik van computerprogramma's kunnen op velerlei manieren verschillende onderverdelingen in typen software gemaakt worden. Het onderscheid dat voor dit essay van belang is heeft te maken met de openheid dan wel de geslotenheid van computerprogramma's.⁴ De mate van openheid heeft vooral te maken met het aanwezig zijn dan wel afwezig zijn van mogelijkheden voor gebruikers om toegang te verkrijgen tot het ontwikkelingsproces en de daarvoor benodigde broncode van een computerprogramma. In feite komt deze toegankelijkheid neer op de volgende twee simpele vragen: heeft een gebruiker toegang tot de broncode (in door de auteur gebruikte programmeertaal) van een computerprogramma of niet; en wat mag de gebruiker daar dan volgens de gebruikerslicentie wel en niet mee doen?

Ruwweg kan software vervolgens worden ingedeeld in open software ('open source software' en 'free software'⁵) versus gesloten commerciële software. De onderstaande tabel vat de verschillen samen.

⁴ Deze paragraaf baseer ik op: Stefan Verhaegh (2003). 'Computerprogramma's: gevangenissen voor gebruikers of virtuele vrijplaatsen. Over openheid als concept voor het democratiseren van technologie en het faciliteren van gebruikersparticipatie.' Online.available: <http://www.globalequality.info/papers/>

⁵ Op het onderscheid tussen open source software en free software ga ik hier niet in, omdat dat voor de strekking van mijn essay niet van belang is. Voor meer informatie over open source zie: <http://www.opensource.org/docs/definition.php>. Voor meer informatie over free software zie: <http://www.gnu.org/philosophy/free-sw.html>. Het grootste verschil is dat de open source licentie zoals de naam al zegt de nadruk vooral legt op openheid, terwijl de GPL free software licentie de nadruk vooral op de vrijheden van de gebruiker legt. Binnen dit verband kan de verzameling free software dan ook als een subdomein binnen open source software gezien worden. Een andere overkoepelende naam die bezig is aan een opmars, is FLOSS (Free/Libre and Open Source Software). Zie: Rishab Ghosh et al (2002). 'Free/Libre and Open Source Software: Survey and Study'. Maastricht: Infonomics. Online.available: <http://www.infonomics.nl/floss/>

open software	gesloten software
meestal gratis (maar niet altijd)	meestal commercieel (niet gratis)
wordt meestal gemaakt door virtueel netwerk van zowel computershobbyïsten als (on)betaalde professionele programmeurs.	wordt meestal gemaakt door programmeurs in loondienst van een bedrijf dat als doel heeft om winst te maken
gebruikt meestal open standaarden, protocollen en interfaces	gebruikt meestal geheime standaarden, protocollen en interfaces
gebruiker heeft toegang tot broncode	gebruiker heeft geen toegang tot broncode
gebruikerslicentie met veel vrijheid voor gebruiker	meestal zeer restrictieve gebruikerslicentie
gebruikers mogen kopieën maken	gebruikers mogen geen kopieën maken
gebruikers mogen kopieën distribueren (meestal alleen onder dezelfde gebruikerslicentie)	gebruikers mogen geen kopieën distribueren
gebruikers mogen het programma op broncodeniveau veranderen	gebruikers mogen niets aan programma veranderen op broncodeniveau
gebruiker kan direct invloed uitoefenen op ontwikkeling van computerprogramma (door rechtstreeks contact met de makers ervan)	gebruiker kan geen directe invloed uitoefenen op ontwikkeling van het computerprogramma (door afschermen van makers van gebruikers)

Hierbij is van belang om te beseffen dat de schaal van open naar gesloten software een continue spectrum is en geen dichotomie tussen twee ideaaltypische tegenpolen. In de praktijk zijn zelfs gevallen bekend van precies dezelfde software die zowel in een open als in een gesloten versie te verkrijgen is.⁶

De aanwezigheid van de broncode van computerprogramma's is cruciaal, omdat de toegang tot de interne werking van software hier volledig van afhankelijk is. Zonder broncode kun je niets aan een computerprogramma veranderen, want de gecompileerde binaire programmacode kan een gebruiker niet geschikt maken voor een ander type computer of besturingssysteem, fouten zijn niet te herstellen en er kunnen geen nieuwe functies toegevoegd of onnodige functies verwijderd worden. Een veelgebruikte metafoor in de literatuur over open software is de vergelijking van gesloten software met een auto met een dichtgelaste motorkap. In dat geval kan een autobezitter namelijk niet kijken hoe de onder de motorkap verborgen technologie werkt, laat staan dat hij reparaties kan uitvoeren, net zomin als dat met gesloten software zonder broncode mogelijk is.

Het gevolg van het ontbreken van broncode is de afhankelijkheid van de gebruiker van gesloten software ten opzichte van de maker. Omdat alleen de maker de broncode bezit kan

⁶ Voor meer informatie over dit fenomeen, zie: Mikko Välimäki (2003) 'Dual Licensing in Open Source Software Industry', Helsinki Institute for Information Technology. Online available: http://www.hiit.fi/u/valimaki/dual_licensing.pdf

ook alleen hij veranderingen aanbrengen in de programma's.⁷ Deze situatie wordt in de literatuur op dit gebied omschreven als een 'vendor lock-in'; de gebruiker zit als het ware vastgeklonken aan de fabrikant door middel van een computerprogramma. Door een gebrek aan keuzevrijheid voor consumenten kan vervolgens ook nog eens een consumer lock-in situatie ontstaan. Hierbij kiezen consumenten door netwerkvoordelen voor één specifiek product, omdat iedereen dat nu eenmaal gebruikt en het daardoor economisch gezien op dat moment de voordeligste keuze is.

De meest zwaarwegende argumenten die de voorstanders van open software claimen voor het gebruik in ontwikkelingsregio's hangen voornamelijk samen met dit afhankelijkheidsprobleem, dat kan ontstaan bij het gebruik van gesloten software waarvoor geen redelijk alternatief voor handen is en waarvan de producent een monopoliepositie bezit. Vooral de negatieve gevolgen die deze monopoliepositie met zich meebrengt voor de toekomstige ontwikkelingen van een lokaal autonoom ICT-beleid, zijn een bron van zorg voor pleitbezorgers van open software.

§3. Twee typen argumenten

In de discussies over welk type software (open of gesloten) het beste gebruikt zou kunnen worden in ontwikkelingsgebieden, worden vaak twee soorten argumenten gebruikt voor de inzet van open software. In deze paragraaf ga ik dieper in op deze argumenten.

In de eerste plaats gebruiken de voorstanders van open software vooral economische argumenten.

A. Zo zou open software goedkoper zijn, in ieder geval wat aankoop betreft, want je kunt het programma namelijk gratis van internet 'downloaden'. De vraag is echter hoe relevant dit is voor mensen in gebieden die geen toegang hebben tot internet (zoals bijna de gehele bevolking van het continent Africa of slechts toegang via een relatief dure en absoluut langzame modemverbinding. Want wat heb je dan nog aan het feit dat je hele CD-Rom's (bijvoorbeeld een Linux-distributie) gratis van internet kan halen, als dat met een langzame verbinding feitelijk onmogelijk is? Niet zo veel natuurlijk.

Daarnaast zou open software ook goedkoper zijn in het onderhoud. Dit is echter maar de vraag, want wanneer er praktisch niemand is die verstand heeft van complexe programma's

⁷ Zogenaamde 'crackers' die gespecialiseerd zijn in het 'kraken' van kopieerbeveiligingen brengen wel rechtstreeks veranderingen aan in de binaire programmabestanden. Niettemin kunnen zij slechts zeer kleine gedeelten van een programma veranderen, en hebben zij daarvoor specialistische en zeer dure hardware nodig. Daarnaast zijn deze activiteiten wettelijk gezien illegaal en vormen zij dus geen oplossing voor het 'vendor lock-in' probleem.

die over het algemeen slechts via ingewikkelde commandolijnopdrachten of via configuratiebestanden te besturen zijn dan heb je nog niet zoveel aan het feit dat de programma's zelf gratis zijn. Omdat op negentig procent van alle computers wereldwijd Microsoft Windows geïnstalleerd is, is de kans dan ook veel groter dat er iemand bereid is en capabel genoeg om het systeembeheer van een Windows-computer te verzorgen.

Daar komt bij dat Microsoft goedkopere en soms zelfs gratis versies beschikbaar stelt aan charitatieve instellingen en ontwikkelingslanden, of grootschalige piraterij (zoals in de exUSSR-landen of in China) oogluikend toestaat, in ruil voor de voordelen die een groter marktaandeel en een grotere gebruikersbasis bieden. Het kosten-argument gaat dus niet altijd a priori op ten gunste van open source software.

B. Een andere reden waarom open source software beter geschikt zou zijn voor ontwikkelingslanden is doordat open source minder nieuwe en zwaar toegeruste computers nodig zou hebben. De vraag is in hoeverre dit geldt. De moderne Linux-distributies hebben net zoveel geheugen en processorkracht nodig als de moderne Windows-varianten (tenminste bij gebruik in een grafische gebruikers omgeving). Het enige gebied waarvoor deze stelling meestal wel op gaat is wanneer Linux gebruikt wordt om een server in te richten. Doordat een grafische gebruiksomgeving meestal overbodig is op een server, en Linux het uitschakelen van deze grafische functionaliteit mogelijk maakt, kan een minder zware computer toch dezelfde functionaliteit bieden als een zwaardere Windows-server. Bij Windows is alle grafische functionaliteit namelijk onlosmakelijk verbonden met het besturingssysteem en kan het ook niet 'uitgeschakeld' worden om systeembronnen vrij te maken.

Daarnaast is er ook het besturingssysteem MS-DOS dat weliswaar veel minder functionaliteit biedt, maar ook extreem veel minder systeemkracht en geheugen nodig heeft. Op een tien jaar oude computer draait MS-DOS probleemloos. Een ander voorbeeld is NewDeal Office pakket (gebaseerd op het GEM besturingssysteem) dat ook op verouderde hardware (80286 processor) en één megabyte werkgeheugen dezelfde functionaliteit kan bieden als het gemiddelde 'Office'-pakket op een moderne 'Pentium 4' computer. Er zijn dus besturingssystemen die met nog veel oudere hardware kunnen werken als het open source besturingssysteem Linux, en deze besturingssystemen zijn lang niet allemaal open source.

C. Het laatste argument dat wordt ingezet is het feit dat open source software de lokale economie zou stimuleren en zou kunnen leiden tot de opbloei van een lokale ICT-industrie. Ook hier dringt zich de vraag op hoe dat in zijn werk zou moeten gaan als er in de eerste plaats bijna niemand is die met computers weet om te gaan, en de weinige mensen die dat wel weten alleen thuis zijn op de veel meer algemeen aanwezige Windows-omgeving.

II. Een tweede soort argumenten die worden ingezet zijn de ‘vrijheid versus afhankelijkheid’-argumenten. In deze beargumentering staat de visie centraal dat gesloten software bijdraagt aan een passieve afhankelijke gebruiker, en dat open software gebruik automatisch leidt tot een meer actieve en autonome computergebruiker.

A. Open source software zou beter aanpasbaar zijn aan lokale talen en karaktersymbolen dan gesloten software. Bij gesloten software ben je hiervoor afhankelijk van de goede wil van de producent. In het geval van open source software kan de lokale bevolking desnoods zelf gelocaliseerde versies produceren van een bepaald computerprogramma, omdat de broncode aanwezig is en dus aangepast kan worden. Ook al wordt hier een theoretisch geldig punt gemaakt, toch vraag ik me af in hoeverre deze stelling ook daadwerkelijk in de praktijk opgaat. Er zijn namelijk geen harde cijfers beschikbaar over de verhouding tussen de beschikbaarheid van gelocaliseerde versies van gesloten software ten opzichte van open software. Daarnaast is ook hier de vraag relevant in hoeverre het van belang is dat er een versie van een open source programma in de eigen taal bestaat, als vervolgens niemand weet hoe het programma werkt en geconfigureerd dient te worden.

B. Daarnaast is er het privacy-punt. In hoeverre weet je in het geval van een gesloten computerprogramma wat het programma in kwestie precies doet, hoe het dit doet en hoe veilig gebruikersdata is opgeslagen en beschermd tegen ongeautoriseerde toegang? Deze vraag kun je nooit met zekerheid beantwoorden als je geen blik kunt werpen op de broncode, en deze broncode vervolgens ook zelf kunt gebruiken om je eigen binaire programma's mee te compileren. Een tegenwerping die zich ook bij dit theoretisch valide punt opwerpt, is in hoeverre dit punt van belang is voor simpele gebruikersapplicaties zoals bijvoorbeeld een tekstverwerkingsprogramma. Natuurlijk is het van belang, dat software die door overheden wordt gebruikt om data over een gehele bevolking met nauwkeurig gescreende software op te slaan, te beheren en te beveiligen. Maar of dit punt op het niveau van gebruikersapplicaties ook valide is, dient nog maar te worden gezien. In dit geval zijn zaken als gebruikersvriendelijkheid van een veel groter belang.

C. Als een fabrikant van een gesloten computerprogramma failliet gaat of besluit een product niet langer te continueren kan een klant niets doen, en wordt hij gedwongen om een product te gebruiken dat niet langer officieel wordt ondersteund, of hij dient over te schakelen op een nieuwer of concurrerend product. Dit soort situaties komen in de praktijk voor, en een voorbeeld ervan is de gang van zaken rondom het besturingssysteem OS/2 van IBM. Jarenlang hebben gebruikers met dit besturingssysteem gebruikt, totdat IBM vorig jaar om

strategische redenen besloot de stekker uit dit project te trekken, waarbij tevreden OS/2-gebruikers verbouwereerd achterbleven. De vraag is echter opnieuw hoe valide dit punt is voor gebruik in ontwikkelingsregio's waar mensen blij zijn als ze überhaupt een computersysteem hebben dat werkt, ook al ontbreekt officiële en commerciële ondersteuning. Daarnaast willen mensen in ontwikkelingsgebieden uiteindelijk dezelfde mooie software-oplossingen kunnen gebruiken als in ontwikkelde gebieden.

Ook is de ondersteuningspraktijk in de open software wereld niet altijd zaligmakend. Meestal is deze inderdaad erg goed, ook al vindt zij voornamelijk informeel in gebruikersgroepen op internet plaats. Niettemin komt het wel voor dat de makers van een open source computerprogramma stoppen met hun (vrijwilligers)werk, waardoor een project 'onbemand' en dus ook zonder support achterblijft. Vaak nemen de meest actieve gebruikers de verdere ontwikkeling dan over, maar dit gebeurt lang niet altijd. Het gebruik van open software betekent dus in theorie wel meer flexibiliteit en autonomie, de vraag is echter of gebruikers in ontwikkelingsgebieden deze voordelen in de praktijk echter wel kunnen verzilveren.

§ 4. Conclusie: pleidooi voor open opties en vrije keuzes

Wat mij in deze discussie vooral tegen de borst stuit, is het gemak waarmee wordt uitgegaan van een soort universele superioriteit van open source software ten opzichte van gesloten software. Hierbij worden zaken als het gemak van kant-en-klare oplossingen en het hogere gebruiksgemak van gesloten software stiekem onder het tapijt geschoven en niet meegenomen in het totaalbeeld. In plaats van dat er daadwerkelijk gekeken wordt naar waar in de praktijk behoefte aan is, worden ontwikkelingsgebieden door voorstanders van open source software gebruikt als een uitbreiding van het strijdtoneel voor het pleiten voor open source software ten koste van de belangen van lokale gebruikers. Deze instelling is zowel naïef, als kortzichtig en zelfzuchtig.

Voorstanders van open source software verkondigen zowel impliciet als expliciet de mening dat open source software altijd beter en goedkoper zou zijn, voor elke situatie beter aanpasbaar en flexibeler zou zijn en dat het ontwikkelingsmodel en de gebruikspraktijk overal democratischer zou zijn. De vraag die ontweken wordt, is of je dit soort zaken wel zo algemeen kan stellen, zowel voor ieder uniek open computerprogramma als voor iedere unieke toepassingscontext. Deze universeel geldige superioriteit van open source software als 'beste' oplossing vind ik een ongenueanceerde en onproductieve stellingname in het debat,

die de complexe praktijk geen recht aan doet. In feite zou je de voorstanders van open source software in dit verband welhaast een soort religieus dogmatisme en een blikveldvernavuwend technisch-imperialisme kunnen verwijten.

Ook de redenering dat open source software mensen minder afhankelijk zou maken is niet altijd hard te maken. Wanneer computerexperts voor de implementatie van een ingewikkelde open source software oplossing kiezen, zonder dat de lokale bevolking wordt getraind om daarmee te kunnen werken, blijft de vraag hoe effectief de gekozen oplossing is, zodra de experts weer uit beeld zijn verdwenen. De kwestie die hierbij centraal staat is dat mensen zonder de benodigde vaardigheden en kennis zelfs van open technologie afhankelijk gemaakt kunnen worden. Een mooi motto in deze is misschien dan ook wel: 'people first, technology second'.

Een ander punt dat ik wil maken ten aanzien van deze discussie is dat het gebruik van open standaarden misschien wel belangrijker is dan het gebruik van open programma's. Wanneer er namelijk gebruik wordt gemaakt van gesloten programma's die echter wel zodanig gemaakt zijn dat ze data kunnen uitwisselen met andere programma's via publiekelijk beschikbare standaarden en protocollen kunnen gebruikers uiteindelijk voor andere oplossingen kiezen op het moment dat daar behoefte aan is, zonder geconfronteerd te worden met hoge omschakelingskosten. Een mooi voorbeeld van een categorie programma's die prima met open standaarden werken zijn webbrowsers. Zo kan een gesloten Windows Internet Explorer 'webbrowser' bijna geheel naadloos samenwerken met een open source Apache 'webserver'. Dit alles dankzij de beschikbaarheid van goede standaarden ten aanzien van informatieuitwisseling op het internet.

Hierbij sluit aan dat het niveau waarop bepaalde software werkzaam is niet uit het oog verloren dient te worden. Zo is er een fundamenteel verschil tussen applicatieprogrammatuur en software die werkt op een meer fundamenteel en infrastructureel niveau zoals op het niveau van protocollen, interfaces en standaarden. Zo kan voor applicaties prima voor gesloten software is worden gekozen. Op dit niveau zijn namelijk vooral aspecten als aanschaf- en onderhoudskosten, en nog belangrijker trainingskosten voor gebruikers en beheerders van belang. Er is dan ook niet zoveel mis om gebruik te maken van de reeds aanwezige kennis (op het gebied van de Windows-omgeving) en (gesloten) programmatuur. Op het applicatieniveau is de gebruiksvriendelijkheid dus wel degelijk een kwestie van groot belang, niet alleen voor het gemak van de gebruikers, maar ook qua kosten en tijd die anders aan extra training besteed zou moeten worden.

Op het infrastructureel niveau is het gebruik van op zijn minst open standaarden, maar liefst ook open source van een groter belang. In de eerste plaats omdat juist hier het belang van de mogelijkheid tot adaptatie en modificatie van en toegang tot de interne werking zeer groot is. Daarnaast betekent een afhankelijkheidspositie op dit niveau een groter gevaar voor gebruikers dan een afhankelijkheidsrelatie op het applicatieniveau, omdat het omschakelen naar andere systemen op fundamenteel niveau veel hogere kosten en grotere problemen met zich meebrengt. Hoe belangrijker een programma is voor het functioneren van een organisatie, hoe belangrijker het is dat een gebruiker potentieel een hoge inclusie kan verwerven (en daarmee zijn eigen belangen veilig kan stellen), waarvoor toegang tot de broncode een eerste vereiste is.

Mensen zouden gestimuleerd moeten worden om te kunnen komen tot een weloverwogen keuze voor ‘the right tool for the right job’. Wanneer gesloten software in een bepaalde locale context een betere oplossing is, dan is het onzin om daar vanwege principiële redenen van af te zien, alleen vanwege het dogma dat open source nu eenmaal ‘beter’ is. Ontwikkelingsregio’s zijn niet de aangewezen plek voor het uitvechten van dogmatische gevechten over pietluttige licentie-details, zoals op dit moment vaak gebeurt. In de plaats daarvan is het maken van een goede analyse van de verschillende voor- en nadelen, afhankelijk van lokaal beschikbare expertise, van veel grotere importantie. Wat de beste oplossing is, is een lokaal gesitueerde vraagstelling, afhankelijk van een veelheid aan complexe sociale en technologische factoren. Hierbij is het veel belangrijker om mensen te trainen om zelf keuzes te kunnen maken, dan mensen te indoctrineren met subjectieve waardeoordelen over bepaalde technologieën.

Ik denk dat het debat over het type gebruikte software afgebogen zou dienen te worden in de richting van een ander en belangrijker debat, namelijk over het belang van computeronderwijs. Het beter om mensen de hulpmiddelen te geven om zelf oplossingen te leren bouwen, in plaats van ze afhankelijk te maken van niet-lokaal aanwezige kennis en vaardigheden. Of om een metafoor te gebruiken: geef iemand een hengel, ipv een vis, of nog beter leer hem om zelf een hengel te maken en deze kennis te delen met anderen. Niettemin wil ik eindigen met een pleidooi voor een realistisch pragmatisme in plaats van een dogmatisch idealisme. Want open software past uiteindelijk vanwege haar openheid en de bijbehorende voordelen mooi in het beeld van lokale zelfvoorzienendheid op ICT-gebied, maar niet altijd, niet overal, niet voor iedereen, en zeker niet ten koste van kant-en-klare oplossingen voor concrete problemen.